LNCS 6151

Feng Bao
Moti Yung
Dongdai Lin
Jiwu Jing (Eds.)

Information Security and Cryptology

5th International Conference, Inscrypt 2009
Beijing, China, December 2009
Revised Selected Papers

Springer

Feng Bao Moti Yung Dongdai Lin
Jiwu Jing (Eds.)

Information Security and Cryptology

5th International Conference, Inscrypt 2009
Beijing, China, December 12-15, 2009
Revised Selected Papers

Springer

Volume Editors

Feng Bao
Institute for Infocomm Research
1 Fusionopolis Way, #19-01 Connexis, South Tower, Singapore 138632, Singapore
E-mail: baofeng@i2r.a-star.edu.sg

Moti Yung
Columbia University, Google Inc. and Computer Science Department
Room 464, S.W. Mudd Building, New York, NY 10027, USA
E-mail: moti@cs.columbia.edu

Dongdai Lin
SKLOIS, Chinese Academy of Sciences, Institute of Software
Beijing 100190, China
E-mail: ddlin@is.iscas.ac.cn

Jiwu Jing
SKLOIS, Graduate University of Chinese Academy of Sciences
19A Yuquan Road, Beijing 100049, China
E-mail: jing@is.ac.cn

Library of Congress Control Number: 2010936095

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

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN	0302-9743
ISBN-10	3-642-16341-6 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-16341-8 Springer Berlin Heidelberg New York

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

Preface

The 5th China International Conference on Information Security and Cryptology (Inscrypt 2009) was co-organized by the State Key Laboratory of Information Security and by the Chinese Association for Cryptologic Research in cooperation with the International Association for Cryptologic Research (IACR). The conference was held in Beijing, China, in the middle of December, and was further sponsored by the Institute of Software, the Graduate University of the Chinese Academy of Sciences and the National Natural Science Foundations of China. The conference is a leading annual international event in the area of cryptography and information security taking place in China. The scientific program of the conference covered all areas of current research in the field, with sessions on central areas of cryptographic research and on many important areas of information security. The conference continues to get the support of the entire international community, reflecting on the fact that the research areas covered by Inscrypt are important to modern computing, where increased security, trust, safety and reliability are required.

The international Program Committee of Inscrypt 2009 received a total of 147 submissions from more than 20 countries and regions, from which only 32 submissions were selected for presentation, 22 of which in the regular papers track and 10 submissions in the short papers track. All anonymous submissions were reviewed by experts in the relevant areas and based on their ranking, technical remarks and strict selection criteria the papers were chosen for the various tracks. The selection to both tracks was a highly competitive process. We further note that due to the conference format, many good papers were regrettably not accepted. Besides the contributed papers, the program also included three invited presentations by Xiaoyun Wang, Roberts Deng and Moti Yung. The program also hosted two additional special tracks with presentations that are not included in these proceedings: one on White-Box Cryptography and Software Protection, and one on Post-Quantum Cryptography.

Inscrypt 2009 was made possible by the joint efforts of numerous people and organizations worldwide. We take this opportunity to thank the Program Committee members and the external experts they employed for their invaluable help in producing the conference program. We further thank the conference Organizing Committee, the various sponsors and the conference attendees. Last but not least, we express our great gratitude to all the authors who submitted papers to the conference, the invited speakers, the contributors to the special tracks and the Session Chairs.

December 2009

Feng Bao
Moti Yung

Inscrypt 2009

5th China International Conference on Information Security and Cryptology

Beijing, China
December 12-15, 2009

Sponsored and organized by

State Key Laboratory of Information Security
(Chinese Academy of Sciences)
Chinese Association for Cryptologic Research

in cooperation with

International Association for Cryptologic Research

Steering Committee

Dengguo Feng	SKLOIS, Chinese Academy of Sciences, China
Dongdai Lin	SKLOIS, Chinese Academy of Sciences, China
Moti Yung	Google Inc. and Columbia University, USA
Chukun Wu	SKLOIS, Chinese Academy of Sciences, China

General Chairs

Dengguo Feng	SKLOIS, Chinese Academy of Sciences, China

Program Committee

Co-chairs

Feng Bao	Institute for Infocomm Research, Singapore
Moti Yung	Google Inc. and Columbia University, USA

Members

Rana Barua	Indian Statistical Institute, India
Zhenfu Cao	Shanghai Jiaotong University, China
Claude Carlet	Universite Paris 8, France
Luigi Catuogno	Ruhr University Bochum, Germany
Liqun Chen	HP Laboratories, UK
Ed Dawson	QUT, Australia
Robert Deng	SMU, Singapore
Xiaotie Deng	City University of Hong Kong, Hong Kong SAR

Jintai Ding	Cincinnati University, USA
Jean-Charles Faugere	INRIA, France
Keith Frikken	Miami University, USA
Alejandro Hevia	University of Chile, Chile
Dennis Hofheinz	CWI, The Netherlands
Brian King	Indiana University - Purdue University, USA
Miroslaw Kutylowski	Wroclaw University of Technology, Poland
Albert Levi	Sabanci University, Turkey
Chao Li	National University of Defence Technology, China
Hui Li	Xidian University, China
Jie Li	University of Tsukuba, Japan
Javier Lopez	University of Malaga, Spain
Xiapu Luo	Georgia Tech, USA
Masahiro Mambo	University of Tsukuba, Japan
Fabio Massacci	University of Trento, Italy
Yi Mu	University of Wollongang, Australia
Svetla Nikova	K.U. Leuven and University of Twente, Belgium
Peng Ning	North Carolina State University, USA
Adam O'Niell	Georgia Tech, USA
Raphael C.-W. Phan	Loughborough University, UK
Olivier Pereira	UCL, Belgium
Josef Pieprzyk	Macquarie University, Australia
Kui Ren	Illinois Institute of Technology, USA
Stelios Sidiroglou-Douskos	MIT, USA
Ioannis Stamatiou	University of Ioannina, Greece
Tsuyoshi Takagi	Future University, Japan
Toshiaki Tanaka	KDDI R&D Labs, Japan
Jacques Traore	Orange Labs, FT, France
Wen-Guey Tzeng	National Chiao Tung University, Taiwan
Daoshun Wang	Tsinghua University, China
Wenling Wu	Chinese Academy of Sciences, China
Yongdong Wu	I2R, Singapore
Shouhai Xu	University of Texas at San Antonio, USA
Yongjin Yeom	ETRI, Korea
Heung Youl Youm	SCH University, Korea
Meng Yu	Western Illinois University, USA
Erik Zenner	Technical University of Denmark
Rui Zhang	AIST, Japan
Yuliang Zheng	University of North Carolina at Charlotte, USA
Jianying Zhou	I2R, Singapore

Proceedings Co-editors

Feng Bao	Institute for Infocomm Research, Singapore
Moti Yung	Google Inc. and Columbia University, USA
Dongdai Lin	SKLOIS, Institute of Software, Chinese Academy of Sciences, China
Jiwu Jing	SKLOIS, Graduate University of Chinese Academy of Sciences, China

Organizing Committee

Co-chairs

Jiwu Jing	SKLOIS, Graduate University of Chinese Academy of Sciences, China
Zhijun Qiang	Chinese Association for Cryptologic Research, China

Members

Chuankun Wu	SKLOIS, Institute of Software, Chinese Academy of Sciences, China
Daren Zha	Graduate University, CAS, China
Xiaoyang Wen	Graduate University, CAS, China
Aihua Zhang	Graduate University, CAS, China

Workshop Chair

Dongdai Lin	SKLOIS, Institute of Software, Chinese Academy of Sciences, China

Publicity Chair

Huafei Zhu	Institute for Infocomm Research, Singapore

Website/Registration

Yicong Liu	Graduate University, CAS, China
Le Kang	Graduate University, CAS, China
Ying Qiu	Institute for Infocomm Research, Singapore

Conference Secretary

Daren Zha	Graduate University, CAS, China
Zongbin Liu	Graduate University, CAS, China

Table of Contents

Integral Cryptanalysis of ARIA

Ping Li[1], Bing Sun[1], and Chao Li[1,2]

[1] Department of Mathematics and System Science, Science College of National University of Defense Technology, Changsha, China, 410073
[2] State Key Laboratory of Information Security, Graduate University of Chinese Academy of Sciences, China, 100190
leave17@gmail.com, happy_come@163.com

Abstract. This paper studies the security of the block cipher ARIA against integral attack. The designers believe that determining whether any given byte position is balanced or not after 3 rounds of encryption is not possible. However, by determining the times that each element of the output of the second round appears is an even integer, we find some 3-round integral distinguishers of ARIA in this paper, which may lead to possible attacks on 4, 5 and 6-round ARIA. Both the data and time complexities of 4-round attack are 2^{25}; the data and time complexities of 5-round attack are $2^{27.2}$ and $2^{76.7}$, respectively; the data and time complexities of 6-round attack are $2^{124.4}$ and $2^{172.4}$, respectively. Moreover, the 4 and 5-round attacks have the lowest data and time complexities compared to existing attacks on ARIA. Our results also show that the choice of S-box and different order of S-boxes do have influence on integral attacks.

Keywords: block cipher, ARIA, integral cryptanalysis, counting method.

1 Introduction

SQUARE attack, proposed by Daemen *et al.* in ref.[8], considers the propagation of sums of (many) ciphertexts with special inputs. In ref.[9], Lucks first applied SQUARE attack to Feistel cipher, which he called saturation attack. In ref.[10], Biryukov and Shamir proposed the multiset attack by which one can break a 4-round SPN cipher even if the S-box is unknown.

In ref.[11], a more generalized attack, namely integral attack, was proposed by Knudsen. The integral of $f(x)$ over some subset V is defined as follows:

$$\int_V f = \sum_{x \in V} f(x),$$

then in FSE 2009, Sun *et al.* generalized the concept of integral and proposed the higher degree integral defined as follows[13]:

$$\int_V (f, i) = \sum_{x \in V} x^i f(x).$$

In ref.[14], by using the algebraic method, Sun *et al.* studied the mathematical foundation of integral attack, and they pointed out that if the algebraic degree of the round function of a Feistel cipher is too low, the integral attack may not be able to recover the round keys even if there do exist some integral distinguishers and they proved that if a cipher is not resilient to integral attack, it is not resilient to interpolation attack either.

Integrals have many interesting features. They are especially well-suited in analyzing ciphers designed with primarily bijective components. Moreover, they exploit the simultaneous relationship between many encryptions, in contrast to differential cryptanalysis where only pairs of encryptions are considered. Consequently, integrals apply to a lot of ciphers which are not vulnerable to differential and linear cryptanalysis. These features have made integral an increasingly popular tool in recent cryptanalysis work. However, these integrals only applies to byte(word)-oriented ciphers, thus in FSE 2008, Z'aba *et al.* proposed the bit-pattern integral attacks which can be applied to bit-oriented ciphers. In fact, the bit-pattern integral is a counting method: if different element of some set $A \subseteq \mathbb{F}_{2^n}$ appears for even times, then $\sum_{x \in A} x = 0$.

ARIA[1] is a 128-bit block cipher designed by a group of Korean experts in 2003. Its design adopts the same idea(wide trail strategy) of the Advanced Encryption Standard(AES)[16]. It was later established as a Korean Standard by the Ministry of Commerce, Industry and Energy in 2004. ARIA supports key length of 128/192/256 bits, and the most interesting characteristic is its involution based on the special usage of neighboring confusion layer and involutional diffusion layer[2].

The security of ARIA was initially analyzed by its designers, including both differential cryptanalysis, linear cryptanalysis, and some other known attacks[1]. Later Biryukov *et al.* performed an evaluation of ARIA, however, they focused mostly on truncated differential cryptanalysis and dedicated linear cryptanalysis [3]. In ref.[5], Wu *et al.* firstly found some non-trivial 4-round impossible differentials which lead to a 6-round attack of ARIA requiring about 2^{121} chosen plaintexts and about 2^{112} encryptions. Li *et al.* presented an algorithm to find 4-round impossible differentials which can also lead to 6-round attack[6]. And recently, Fleischmann *et al.* evaluated its security against boomerang attack which has the lowest memory requirements up to now[7].

Since the branch number of linear permutation ARIA used is 8 which is larger than that of AES, the designers of ARIA believe that no any 3-round integral distinguisher exists and one can construct only 2-round integral distinguishers[1]. In ref.[3], it is said that the fast diffusion does not allow a 3-round distinguisher as in Rijndael. However, in this paper, by counting the times that some elements appears before it passes the S-box, we find some 3-round distinguishers which lead to 4, 5 and 6-round integral attack. Besides, we find a new type of distinguishers which doesn't determine whether a given byte is active, balanced or constant, but adopts the relation between sum of different bytes. Note that in some papers, integrals are regarded as higher order differentials[15]. In this paper, we use integrals instead of higher order differentials.

The rest of this paper is organized as follows. In Section 2 we give a brief description of ARIA. In Section 3 we give a 3-round integral distinguisher. In Section 4, we present the integral attack on 4, 5 and 6-round ARIA, respectively. Finally, Section 5 summarizes this paper.

2 Description of ARIA

ARIA is a 128-bit SPN block cipher accepting keys of 128, 192, and 256-bit, which adopts an involutional binary 16×16 matrix over $\mathbb{F}_2$ in its diffusion layer. The substitution layer consists of sixteen 8×8-bit S-boxes based on the inversion in $\mathbb{F}_{2^8}$. The number of rounds is 12, 14, or 16, depending on the key length[4]. The plaintext/ciphertext, as well as the input and output of the round function, are treated as vectors over $GF(2^8)^{16}$ and we call them states.

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}

For convenience, sometimes we will treat the vectors as 4×4 matrices with elements in $GF(2^8)$, depicted as follows.

x_0	x_4	x_8	x_{12}
x_1	x_5	x_9	x_{13}
x_2	x_6	x_{10}	x_{14}
x_3	x_7	x_{11}	x_{15}

The round function of ARIA firstly applies a *Round Key Addition*, then a *Substitution Layer* and at last a *Diffusion Layer* subsequently. An N-round ARIA iterates the round function $N-1$ times; and in the last round the diffusion layer is replaced by the Round Key Addition. The 3 operations are defined as follows:

Round Key Addition(RKA). The 128-bit round key is simply XORed to the state. The round keys are derived from the cipher key by means of the key schedule.

Substitution Layer(SL). A non-linear byte substitution operates on each byte of the state independently which is implemented by two S-boxes S_1 and S_2. ARIA has two types of S-Box layers for odd and even rounds as shown in Fig.1. Type 1 is used in the odd rounds and type 2 is used in the even rounds.

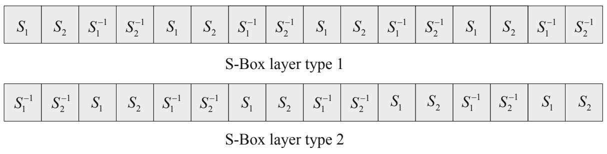

S_1 S_2 S_1^{-1} S_2^{-1} S_1 S_2 S_1^{-1} S_2^{-1} S_1 S_2 S_1^{-1} S_2^{-1} S_1 S_2 S_1^{-1} S_2^{-1}
S-Box layer type 1
S_1^{-1} S_2^{-1} S_1 S_2 S_1^{-1} S_2^{-1} S_1 S_2 S_1^{-1} S_2^{-1} S_1 S_2 S_1^{-1} S_2^{-1} S_1 S_2
S-Box layer type 2

Fig. 1. The two types of S-Box layers

Diffusion Layer(DL). A linear transformation $P : \mathbb{F}_{2^8}^{16} \to \mathbb{F}_{2^8}^{16}$ with branch number 8 was selected to improve the diffusion effect and increase efficiency in both hardware and software implementations[2]. P is given by

$$(x_0, x_1, \ldots, x_{15}) \mapsto (y_0, y_1, \ldots, y_{15}),$$

where

$$\begin{aligned}
y_0 &= x_3 \oplus x_4 \oplus x_6 \oplus x_8 \oplus x_9 \oplus x_{13} \oplus x_{14}, & y_8 &= x_0 \oplus x_1 \oplus x_4 \oplus x_7 \oplus x_{10} \oplus x_{13} \oplus x_{15},\\
y_1 &= x_2 \oplus x_5 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{12} \oplus x_{15}, & y_9 &= x_0 \oplus x_1 \oplus x_5 \oplus x_6 \oplus x_{11} \oplus x_{12} \oplus x_{14},\\
y_2 &= x_1 \oplus x_4 \oplus x_6 \oplus x_{10} \oplus x_{11} \oplus x_{12} \oplus x_{15}, & y_{10} &= x_2 \oplus x_3 \oplus x_5 \oplus x_6 \oplus x_8 \oplus x_{13} \oplus x_{15},\\
y_3 &= x_0 \oplus x_5 \oplus x_7 \oplus x_{10} \oplus x_{11} \oplus x_{13} \oplus x_{14}, & y_{11} &= x_2 \oplus x_3 \oplus x_4 \oplus x_7 \oplus x_9 \oplus x_{12} \oplus x_{14},\\
y_4 &= x_0 \oplus x_2 \oplus x_5 \oplus x_8 \oplus x_{11} \oplus x_{14} \oplus x_{15}, & y_{12} &= x_1 \oplus x_2 \oplus x_6 \oplus x_7 \oplus x_9 \oplus x_{11} \oplus x_{12},\\
y_5 &= x_1 \oplus x_3 \oplus x_4 \oplus x_9 \oplus x_{10} \oplus x_{14} \oplus x_{15}, & y_{13} &= x_0 \oplus x_3 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_{10} \oplus x_{13},\\
y_6 &= x_0 \oplus x_2 \oplus x_7 \oplus x_9 \oplus x_{10} \oplus x_{12} \oplus x_{13}, & y_{14} &= x_0 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_9 \oplus x_{11} \oplus x_{14},\\
y_7 &= x_1 \oplus x_3 \oplus x_6 \oplus x_8 \oplus x_{11} \oplus x_{12} \oplus x_{13}, & y_{15} &= x_1 \oplus x_2 \oplus x_4 \oplus x_5 \oplus x_8 \oplus x_{10} \oplus x_{15}.
\end{aligned}$$

The Key Schedule of ARIA is omitted and we refer to ref.[1] for more details.

3 3-Round Integral Distinguishers of ARIA

Previous results show that there exists 2-round distinguishers of ARIA and it is hardly to construct a 3-round one. However, in this section, we describe how to construct 3-round integral distinguisher of ARIA by using counting method combined with some algebraic techniques.

3.1 Integral Cryptanalysis

The following definitions are essential when applying an integral attack:

Definition 1. *A set* $\{a_i | a_i \in \mathbb{F}_{2^n}, 0 \le i \le 2^n - 1\}$ *is* active*, if for any* $0 \le i < j \le 2^n - 1$*,* $a_i \ne a_j$*.*

Definition 2. *A set* $\{a_i | a_i \in \mathbb{F}_{2^n}, 0 \le i \le 2^n - 1\}$ *is* passive*, if for any* $0 < i \le 2^n - 1$*,* $a_i = a_0$*.*

Definition 3. *A set* $\{a_i | a_i \in \mathbb{F}_{2^n}, 0 \le i \le 2^n - 1\}$ *is* balanced*, if the sum of all element of the set is* 0*, that is* $\sum_{i=0}^{2^n-1} a_i = 0$*.*

The following principles are needed in order to find an integral distinguisher:

(1) An active set remains active after passing a bijective transform.

(2) The linear combination of several active/balanced sets is a balanced set.

(3) The property of a balanced set after passing through a nonlinear transformation is unknown.

Obviously, the third one is the bottleneck of integral attacks. And if we can determine the property of a balanced set after it passes a nonlinear transformation, new integral distinguisher with more rounds can be found. In the rest of this paper, we will determine whether a set is balanced or not by using a counting method.

3.2 2.5-Round Integral Distinguishers of ARIA

In the following paper, C always denote some constant value but not necessarily equal to each other at different positions.

Lemma 1. *Let the input of ARIA be $B = (B_0, B_1, \ldots, B_{15})$, the i-th round key be $k_i = (k_{i,0}, k_{i,1}, \ldots, k_{i,15})$, and the outputs of S-Box layer and P layer of the i-th round be $Z_i = (Z_{i,0}, Z_{i,1}, \ldots, Z_{i,15})$ and $Y_i = (Y_{i,0}, Y_{i,1}, \ldots, Y_{i,15})$, respectively. If B_0 takes all values of $\mathbb{F}_{2^8}$ and B_is are constants where $1 \le i \le 15$, then $Z_{3,6}$, $Z_{3,9}$ and $Z_{3,15}$ are balanced.*

Proof. Let the input be

$$B = \begin{pmatrix} x & C & C & C \\ C & C & C & C \\ C & C & C & C \\ C & C & C & C \end{pmatrix},$$

and $y = S_1(x \oplus k_{1,0})$, then according to the definition of ARIA, the output of the first round is

$$Y_1 = \begin{pmatrix} C & y \oplus \beta_4 & y \oplus \beta_8 & C \\ C & C & y \oplus \beta_9 & y \oplus \beta_{13} \\ C & y \oplus \beta_6 & C & y \oplus \beta_{14} \\ y \oplus \beta_3 & C & C & C \end{pmatrix}.$$

Let $\gamma_i = \beta_i \oplus k_{2,i}$, then

$$Z_2 = \begin{pmatrix} C & S_1^{-1}(y \oplus \gamma_4) & S_1^{-1}(y \oplus \gamma_8) & C \\ C & C & S_2^{-1}(y \oplus \gamma_9) & S_2^{-1}(y \oplus \gamma_{13}) \\ C & S_1(y \oplus \gamma_6) & C & S_1(y \oplus \gamma_{14}) \\ S_2(y \oplus \gamma_3) & C & C & C \end{pmatrix},$$

thus

$$\begin{cases} Y_{2,6} = S_2^{-1}(y \oplus \gamma_9) \oplus S_2^{-1}(y \oplus \gamma_{13}) \oplus C_1 \\ Y_{2,9} = S_1(y \oplus \gamma_6) \oplus S_1(y \oplus \gamma_{14}) \oplus C_2 \\ Y_{2,15} = S_1^{-1}(y \oplus \gamma_4) \oplus S_1^{-1}(y \oplus \gamma_8) \oplus C_3 \end{cases}$$

where C_i are some constants. Now let's take $Y_{2,6}$ as an example. If $\gamma_9 = \gamma_{13}$, then $Y_{2,6} = C_1$; if $\gamma_9 \neq \gamma_{13}$, then $S_2^{-1}(y \oplus \gamma_9) \oplus S_2^{-1}(y \oplus \gamma_{13}) \oplus C_1 = S_2^{-1}(y^* \oplus \gamma_9) \oplus S_2^{-1}(y^* \oplus \gamma_{13}) \oplus C_1$, where $y^* = y \oplus \gamma_9 \oplus \gamma_{13} \neq y$. In both cases, each value of $Y_{2,6}$ appears even times. Thus each value of $Z_{3,6}$ appears even times, which implies that $Z_{3,6}$ is balanced. This ends our proof. $\square$

The distinguisher shown in Lemma 1 can be simply denoted by $[0, (6, 9, 15)]$. Table 1 lists all possible values for $[a, (b, c, d)]$ which means that if only the a-th byte of input takes all values of $\mathbb{F}_{2^8}$ and other bytes are constants, then $Z_{3,b}$, $Z_{3,c}$ and $Z_{3,d}$ are balanced:

Table 1. 2.5-Round Integral Distinguishers of ARIA

Active byte	Balanced bytes	Active byte	Balanced bytes
0	6, 9, 15	8	1, 7, 14
1	7, 8, 14	9	0, 6, 15
2	4, 11, 13	10	3, 5, 12
3	5, 10, 12	11	2, 4, 13
4	2, 11, 13	12	3, 5, 10
5	3, 10, 12	13	2, 4, 11
6	0, 9, 15	14	1, 7, 8
7	1, 8, 14	15	0, 6, 9

3.3 3-Round Integral Distinguishers of ARIA

Theorem 1. *Let the input of ARIA be $B = (B_0, B_1, \ldots, B_{15})$, the i-th round key be $k_i = (k_{i,0}, k_{i,1}, \ldots, k_{i,15})$, and the outputs of S layer and P layer of the i-th round be $Z_i = (Z_{i,0}, Z_{i,1}, \ldots, Z_{i,15})$ and $Y_i = (Y_{i,0}, Y_{i,1}, \ldots, Y_{i,15})$, respectively. If (B_0, B_5, B_8) takes all values of $\mathbb{F}_{2^8}^3$ and B_is are constants where $i \neq 0,\ 5,\ 8$, then $Y_{3,2}$, $Y_{3,5}$, $Y_{3,11}$ and $Y_{3,12}$ are balanced.*

Proof. Since

$$\begin{cases} Y_{3,2} = Z_{3,1} \oplus Z_{3,4} \oplus Z_{3,6} \oplus Z_{3,10} \oplus Z_{3,11} \oplus Z_{3,12} \oplus Z_{3,15}, \\ Y_{3,5} = Z_{3,1} \oplus Z_{3,3} \oplus Z_{3,4} \oplus Z_{3,9} \oplus Z_{3,10} \oplus Z_{3,14} \oplus Z_{3,15}, \\ Y_{3,11} = Z_{3,2} \oplus Z_{3,3} \oplus Z_{3,4} \oplus Z_{3,7} \oplus Z_{3,9} \oplus Z_{3,12} \oplus Z_{3,14}, \\ Y_{3,12} = Z_{3,1} \oplus Z_{3,2} \oplus Z_{3,6} \oplus Z_{3,7} \oplus Z_{3,9} \oplus Z_{3,11} \oplus Z_{3,12}. \end{cases}$$

From Table 1, we have the following 2.5-round distinguishers:

$$[0, (6, 9, 15)], \quad [5, (3, 10, 12)], \quad [8, (1, 7, 14)],$$

thus when (B_0, B_5, B_8) takes all values of $\mathbb{F}_{2^8}^3$, the bytes $Z_{3,1}$, $Z_{3,3}$, $Z_{3,6}$, $Z_{3,7}$, $Z_{3,9}$, $Z_{3,10}$, $Z_{3,12}$, $Z_{3,14}$ and $Z_{3,15}$ are balanced. So we check whether $Z_{3,2}$, $Z_{3,4}$ and $Z_{3,11}$ are balanced.

Let the input be

$$B = \begin{pmatrix} x & C & z & C \\ C & y & C & C \\ C & C & C & C \\ C & C & C & C \end{pmatrix},$$

and $x^* = S_1(x \oplus k_{1,0})$, $y^* = S_2(y \oplus k_{1,5})$, $z^* = S_1(z \oplus k_{1,8})$, then the output of the first round is

$$Y_1 = \begin{pmatrix} z^* \oplus C_0 & x^* \oplus y^* \oplus z^* \oplus C_4 & x^* \oplus C_8 & C \\ y^* \oplus z^* \oplus C_1 & C & x^* \oplus y^* \oplus C_9 & x^* \oplus z^* \oplus C_{13} \\ C & x^* \oplus C_6 & y^* \oplus z^* \oplus C_{10} & x^* \oplus y^* \oplus C_{14} \\ x^* \oplus y^* \oplus C_3 & z^* \oplus C_7 & C & y^* \oplus z^* \oplus C_{15} \end{pmatrix}$$

Let $y^* \oplus z^* = m$ and $\gamma_i = C_i \oplus k_{2,i}$, then

$$\begin{aligned} Y_{2,2} = & S_2^{-1}(m \oplus \gamma_1) \oplus S_1^{-1}(x^* \oplus m \oplus \gamma_4) \oplus S_1(x^* \oplus \gamma_6) \\ & \oplus S_1(m \oplus \gamma_{10}) \oplus S_2(m \oplus \gamma_{15}) \oplus C^* \end{aligned}$$

Since there are 256 different values (y^*, z^*) such that $y^* \oplus z^* = m$, thus each value of $Y_{2,2}$ appears for $256 \times N$ times where N is an integer which implies that $Z_{3,2}$ is balanced.

On the other hand, we have

$$\begin{aligned} Y_{2,4} = & S_1^{-1}(z^* \oplus \gamma_0) \oplus S_1^{-1}(x^* \oplus \gamma_8) \oplus S_1(x^* \oplus y^* \oplus \gamma_{14}) \\ & \oplus S_2(y^* \oplus z^* \oplus \gamma_{15}) \oplus C \end{aligned}$$

Let $x^* \oplus y^* = m$ and $z^* \oplus y^* = n$, then

$$\begin{aligned} Y_{2,4} = & S_1^{-1}(y^* \oplus (n \oplus \gamma_0)) \oplus S_1^{-1}(y^* \oplus (m \oplus \gamma_8)) \\ & \oplus S_1(m \oplus \gamma_{14}) \oplus S_2(n \oplus \gamma_{15}) \oplus C \end{aligned}$$

According to the proof of Lemma 1, each value of $Y_{2,4}$ appears for even times thus $Z_{2,4}$ is balanced.

Let $x^* \oplus y^* = m$, then

$$\begin{aligned} Y_{2,11} = & S_2(m \oplus \gamma_3) \oplus S_1^{-1}(m \oplus z^* \oplus \gamma_4) \oplus S_2(z^* \oplus \gamma_7) \\ & \oplus S_2^{-1}(m \oplus \gamma_9) \oplus S_1(m \oplus \gamma_{14}) \oplus C \end{aligned}$$

Since there are 256 different values (x^*, y^*) such that $x^* \oplus y^* = m$, thus each value of $Y_{2,11}$ appears $256 \times N$ times where N is an integer which implies that $Z_{3,11}$ is balanced.

Since $Z_{3,2}$, $Z_{3,4}$ and $Z_{3,11}$ are balanced, and take Lemma 1 into consideration, $Y_{3,2}$, $Y_{3,5}$, $Y_{3,11}$ and $Y_{3,12}$ are balanced. □

Corollary 1. *Let the input of ARIA be $B = (B_0, B_1, \dots, B_{15})$, the i-th round key be $k_i = (k_{i,0}, k_{i,1}, \dots, k_{i,15})$, and the outputs of S layer and P layer of the i-th round be $Z_i = (Z_{i,0}, Z_{i,1}, \dots, Z_{i,15})$ and $Y_i = (Y_{i,0}, Y_{i,1}, \dots, Y_{i,15})$, respectively. If (B_0, B_5, B_8) takes all values of $\mathbb{F}_{2^8}^3$ and B_is are constants where $i \neq 0,\ 5,\ 8$, then $\sum_{B_0,B_5,B_8} Y_{3,0} = \sum_{B_0,B_5,B_8} Y_{3,7} = \sum_{B_0,B_5,B_8} Y_{3,10}$.*

In previous integral distinguishers, we adopt the fact that the integral of the ciphertexts over some subset is 0. However, Corollary 1 adopts the fact that the sums of different bytes are equal.

4 Integral Attacks on Round-Reduced ARIA

Let the plaintext and ciphertext of ARIA be $PT = (PT_0, PT_1, \dots, PT_{15})$ and $CT = (CT_0, CT_1, \dots, CT_{15})$, respectively; the round key, the outputs of S-Box layer and P layer of the i-th round be $k_i = (k_{i,0}, k_{i,1}, \dots, k_{i,15})$, $Z_i = (Z_{i,0}, Z_{i,1}, \dots, Z_{i,15})$ and $Y_i = (Y_{i,0}, Y_{i,1}, \dots, Y_{i,15})$, respectively. According to Theorem 1, we can mount integral attacks on 4, 5 and 6-round ARIA. We present the attack only on the third byte of the outputs of the 3-round distinguisher described in Theorem 1. Note that the diffusion layer is replaced by the round key addition in the last round of ARIA.

4.1 Integral Attack on 4-Round ARIA

In this subsection, we describe an integral attack on 4-round ARIA. The attack is based on the above 3-round integral distinguishers with additional one round at the end as shown in Fig.2.

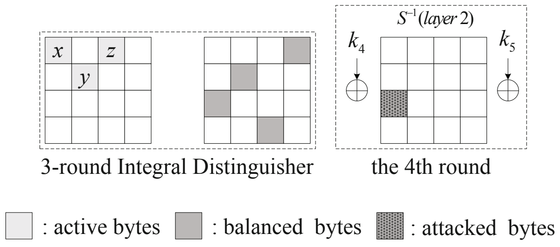


Fig. 2. Integral Attack on 4-Round ARIA

Step 1. Choose a structure of plaintexts of the following form:

$$PT(x,y,z) = \begin{pmatrix} x & C & z & C \\ C & y & C & C \\ C & C & C & C \\ C & C & C & C \end{pmatrix},$$

where (x, y, z) takes all values of $\mathbb{F}_{2^8}^3$ and C denotes some constant. Set 256 counters, for each value of the third byte of ciphertexts $CT_2(x, y, z)$, the corresponding counter plus one.

Step 2. Let $V = \{i|\text{count}[i] \text{ is odd}, i = 0, 1, \ldots, 255\}$. Guess a value for $k_{5,2}$, say k^*, and check whether the following equation holds:

$$\sum_{t \in V} S_1^{-1}(t \oplus k^*) = 0. \tag{1}$$

If Equ.(1) holds, then k^* might be a candidate of $k_{5,2}$, otherwise, it is a wrong guess.

Step 3. Repeat Step 1 and Step 2 until $k_{5,2}$ is uniquely determined.

For a wrong key, the probability that it can pass Equ.(1) is 2^{-8}, thus after analyzing a structure, the number of wrong keys that can pass Equ.(1) is $(2^8 - 1) \times 2^{-8} \approx 1$, thus to uniquely determine $k_{5,2}$, we need to analyze two structures. In Step 2, for each k^*, it needs no more than $2^8 \times 2^8 = 2^{16}$ table lookups.

Accordingly, the data complexity of the attack is about 2^{25} chosen plaintexts; the time complexity is $2^{25} + 2^{16}/(4 \times 16) \approx 2^{25}$; and the attack needs 2^8 bytes to store 256 counters.

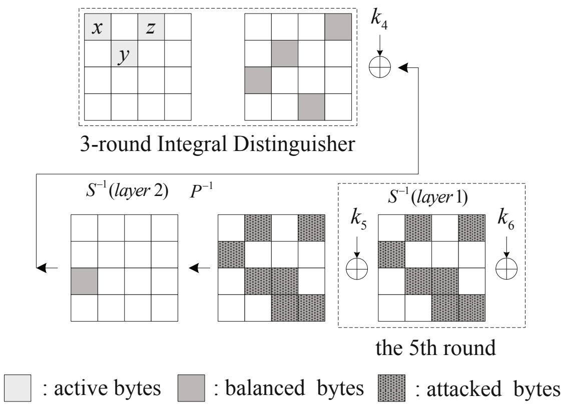


Fig. 3. Integral Attack on 5-Round ARIA

4.2 Integral Attack on 5-Round ARIA

In this subsection, we describe an integral attack on 5-round ARIA. The attack is based on the above 3-round integral distinguishers with additional two rounds at the end as shown in Fig.3.

Step 1. Choose a structure of plaintexts of the following form:

$$PT(x,y,z) = \begin{pmatrix} x & C & z & C \\ C & y & C & C \\ C & C & C & C \\ C & C & C & C \end{pmatrix},$$

where (x,y,z) takes all values of $\mathbb{F}_{2^8}^3$ and C denotes some constant.

Step 2. Guess $k_{6,1}$, $k_{6,4}$, $k_{6,6}$, $k_{6,10}$, $k_{6,11}$, $k_{6,12}$ and $k_{6,15}$, then compute

$$T_1 = S_2^{-1}(CT_1 \oplus k_{6,1}) \oplus S_1^{-1}(CT_4 \oplus k_{6,4}) \oplus S_1(CT_6 \oplus k_{6,6}) \oplus S_1(CT_{10} \oplus k_{6,10}) \oplus S_2(CT_{11} \oplus k_{6,11}) \oplus S_1(CT_{12} \oplus k_{6,12}) \oplus S_2(CT_{15} \oplus k_{6,15}).$$

Set 256 counters, and for each value of T_1, the corresponding counter plus one.

Step 3. Let $k^* = k_{5,1} \oplus k_{5,4} \oplus k_{5,6} \oplus k_{5,10} \oplus k_{5,11} \oplus k_{5,12} \oplus k_{5,15}$, and $V = \{i | \text{count}[i] \text{ is odd}, i = 0, 1, \ldots, 255\}$, then guess a value for k^* and check whether the following equation holds:

$$\sum_{t \in V} S_1^{-1}(t \oplus k^*) = 0. \tag{2}$$

If Equ.(2) doesn't hold, the combination of previous guessed value for the first round, the last round and a linear combination of the fifth round is a wrong guess.
Step 4. Repeat Step 1 to 3 until the combination of the 8 bytes is uniquely determined.

Since the probability that a wrong key can pass the check is 2^{-8} and there are altogether 8 bytes to guess, thus data complexity of the attack is $9 \times 2^{24} \approx 2^{27.2}$.

For each structure(2^{24} plaintexts), it seems that we must guess 8 bytes, however, since in Step 3, the counter have eliminated a large part of states, thus complexity of this step can be omitted compared with previous steps. Therefore for each structure, it only needs to guess 7 bytes, and the time complexity of this attack is $2^{24} \times 2^{56} \times 8/(5 \times 16) \approx 2^{76.7}$ encryptions.

4.3 Integral Attack on 6-Round ARIA

In this subsection, we describe an integral attack on 6-round ARIA. The attack is based on the above 3-round integral distinguishers with additional one round at the beginning and two rounds at the end as shown in Fig.4.

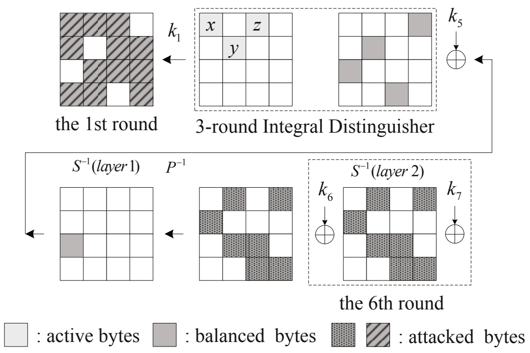


Fig. 4. Integral Attack on 6-Round ARIA

Step 1. Let

$$B(x,y,z) = \begin{pmatrix} S_1^{-1}(z) & S_1^{-1}(x \oplus y \oplus z) & S_1^{-1}(x) & C \\ S_2^{-1}(y \oplus z) & C & S_2^{-1}(x \oplus y) & S_2^{-1}(x \oplus z) \\ C & S_1(x) & S_1(y \oplus z) & S_1(x \oplus y) \\ S_2(x \oplus y) & S_2(z) & C & S_2(y \oplus z) \end{pmatrix},$$

guess $k_{1,0}$, $k_{1,1}$, $k_{1,3}$, $k_{1,4}$, $k_{1,6}$, $k_{1,7}$, $k_{1,8}$, $k_{1,9}$, $k_{1,10}$, $k_{1,13}$, $k_{1,14}$, $k_{1,15}$, and let

$$K = \begin{pmatrix} k_{1,0} & k_{1,4} & k_{1,8} & 0 \\ k_{1,1} & 0 & k_{1,9} & k_{1,13} \\ 0 & k_{1,6} & k_{1,10} & k_{1,14} \\ k_{1,3} & k_{1,7} & 0 & k_{1,15} \end{pmatrix}.$$

Then, choose a structure of plaintexts of the following form:

$$PT(x,y,z) = B(x,y,z) \oplus K,$$

the corresponding ciphertexts are denoted by $CT(x,y,z)$.

Step 2. Guess $k_{7,1}$, $k_{7,4}$, $k_{7,6}$, $k_{7,10}$, $k_{7,11}$, $k_{7,12}$ and $k_{7,15}$, then compute

$$\begin{aligned} T_1 = {} & S_2(CT_1 \oplus k_{7,1}) \oplus S_1(CT_4 \oplus k_{7,4}) \oplus S_1^{-1}(CT_6 \oplus k_{7,6}) \oplus S_1^{-1}(CT_{10} \oplus k_{7,10}) \\ & \oplus S_2^{-1}(CT_{11} \oplus k_{7,11}) \oplus S_1(CT_{12} \oplus k_{7,12}) \oplus S_2^{-1}(CT_{15} \oplus k_{7,15}). \end{aligned}$$

Set 256 counters, and for each value of T_1, the corresponding counter plus one.

Step 3. Let $k^* = k_{6,1} \oplus k_{6,4} \oplus k_{6,6} \oplus k_{6,10} \oplus k_{6,11} \oplus k_{6,12} \oplus k_{6,15}$, and $V = \{i | \text{count}[i] \text{ is odd}, i = 0, 1, \ldots, 255\}$, then guess a value for k^* and check whether the following equation holds:

$$\sum_{t \in V} S_1(t \oplus k^*) = 0. \tag{3}$$

If Equ.(3) doesn't hold, the combination of previous guessed value for the first round, the last round and a linear combination of the fifth round is a wrong guess.

Step 4. Repeat Step 1, Step 2 and Step 3 until the combination of the 20 bytes is uniquely determined.

Since the probability that a wrong key can pass the check is 2^{-8} and there are altogether 20 bytes to guess, thus data complexity of the attack is $21 \times 2^{24} \times 2^{8 \times 12} \approx 2^{124.4}$.

For each structure(2^{24} plaintexts), if the subkeys of the first round are correct, the input of the second round is the same as to the 3-round integral distinguisher, thus the correct subkeys of the last two rounds must pass Equ.(3).

In the last two rounds, it seems that we must guess 8 bytes, however, since in Step 3, the counter have eliminated a large part of states, thus complexity of this step can be omitted. Therefore for each structure, it only needs to guess 7 bytes, and the time complexity of this attack is $2^{96} \times 2^{24} \times 2^{56} \times 8/(6 \times 16) \approx 2^{172.4}$ encryptions.

Table 2. Comparison of Attacks on ARIA

Attack	Rounds	Data	Time	Source
Integral Attack	4	2^{25}	2^{25}	Sec.4.1
Impossible Differential	5	$2^{71.3}$	$2^{71.6}$	[6]
Boomerang Attack	5	2^{57}	$2^{115.5}$	[7]
Integral Attack	5	$2^{27.2}$	$2^{76.7}$	Sec.4.2
Impossible Differential	6	2^{121}	2^{112}	[5]
Impossible Differential	6	$2^{120.5}$	$2^{104.5}$	[6]
Impossible Differential	6	2^{113}	$2^{121.6}$	[6]
Boomerang Attack	6	2^{57}	$2^{171.2}$	[7]
Integral Attack	6	$2^{124.4}$	$2^{172.4}$	Sec.4.3
Truncated Differential	7	2^{81}	2^{81}	[3]
Dedicated Linear Attack	10	2^{119}	2^{88}	[3]

Table 3. 3-Round Optimal Integral Distinguishers of ARIA

Active	Balanced
0 1 10	0, 1,10,11
0 1 11	0, 1,10,11
0 2 5	0, 2, 5, 7
0 2 7	0, 2, 5, 7
0 3 13	1, 2,12,15
0 3 14	1, 2,12,15
0 4 10	1, 7,10,12
0 4 14	1, 7,10,12
0 5 7	0, 2, 5, 7
0 5 8	2, 5,11,12
0 5 10	3
0 5 13	2, 5,11,12
0 5 15	4
0 6 11	9
0 6 13	13
0 7 9	6
0 7 11	0, 5,10,15
0 7 12	0, 5,10,15
0 7 14	3
0 8 13	2, 5,11,12
0 9 14	14
0 10 11	0, 1,10,11
0 10 14	1, 7,10,12
0 10 15	8
0 11 12	0, 5,10,15
0 11 13	3
0 13 14	1, 2,12,15
1 2 12	0, 3,13,14
1 2 15	0, 3,13,14
1 3 4	1, 3, 4, 6
1 3 6	1, 3, 4, 6
1 4 6	1, 3, 4, 6
1 4 9	3, 4,10,13
1 4 11	2
1 4 12	3, 4,10,13
1 4 14	5
1 5 11	0, 6,11,13
1 5 15	0, 6,11,13
1 6 8	7
1 6 10	1, 4,11,14
1 6 13	1, 4,11,14
1 6 15	2
1 7 10	8
1 7 12	12
1 8 15	15
1 9 12	3, 4,10,13
1 10 11	0, 1,10,11
1 10 12	2
1 10 13	1, 4,11,14
1 11 14	9
1 11 15	0, 6,11,13
1 12 15	0, 3,13,14
2 3 8	2, 3, 8, 9
2 3 9	2, 3, 8, 9
2 4 9	11
2 4 15	15
2 5 7	0, 2, 5, 7
2 5 9	2, 7, 8,13
2 5 11	4
2 5 12	1
2 5 14	2, 7, 8,13
2 6 8	3, 5, 8,14
2 6 12	3, 5, 8,14
2 7 8	1
2 7 10	0, 7, 9,14
2 7 13	6
2 7 15	0, 7, 9,14
2 8 9	2, 3, 8, 9
2 8 12	3, 5, 8,14
2 8 13	10
2 9 14	2, 7, 8,13
2 9 15	1
2 10 15	0, 7, 9,14
2 11 12	12
2 12 15	0, 3,13,14
3 4 6	1, 3, 4, 6
3 4 8	3, 6, 9,12
3 4 10	5
3 4 13	0
3 4 15	3, 6, 9,12
3 5 8	10
3 5 14	14
3 6 9	0
3 6 11	1, 6, 8,15
3 6 12	7
3 6 14	1, 6, 8,15
3 7 9	2, 4, 9,15
3 7 13	2, 4, 9,15
3 8 9	2, 3, 8, 9
3 8 14	0
3 8 15	3, 6, 9,12
3 9 12	11
3 9 13	2, 4, 9,15
3 10 13	13
3 11 14	1, 6, 8,15
3 13 14	1, 2,12,15
4 5 14	6, 7,12,13
4 5 15	6, 7,12,13
4 7 9	4, 7, 9,10
4 7 10	4, 7, 9,10
4 8 15	3, 6, 9,12
4 9 10	4, 7, 9,10
4 9 12	3, 4,10,13
4 9 15	5
4 10 13	8
4 10 14	1, 7,10,12
4 11 14	14
4 14 15	6, 7,12,13
5 6 8	5, 6, 8,11
5 6 11	5, 6, 8,11
5 8 11	5, 6, 8,11
5 8 13	2, 5,11,12
5 8 14	4
5 9 14	2, 7, 8,13
5 10 15	15
5 11 12	9
5 11 15	0, 6,11,13
5 14 15	6, 7,12,13
6 7 12	4, 5,14,15
6 7 13	4, 5,14,15
6 8 11	5, 6, 8,11
6 8 12	3, 5, 8,14
6 8 15	10
6 9 12	12
6 10 13	1, 4,11,14
6 11 13	7
6 11 14	1, 6, 8,15
6 12 13	4, 5,14,15
7 8 13	13
7 9 10	4, 7, 9,10
7 9 13	2, 4, 9,15
7 9 14	11
7 10 12	6
7 10 15	0, 7, 9,14
7 11 12	0, 5,10,15
7 12 13	4, 5,14,15
8 10 13	9,11,12,14
8 10 15	9,11,12,14
8 13 15	9,11,12,14
9 11 12	8,10,13,15
9 11 14	8,10,13,15
9 12 14	8,10,13,15
10 13 15	9,11,12,14
11 12 14	8,10,13,15

5 Conclusion

When applying integral attack on AES, we can use the Higher Order Integrals, thus we do not need to guess the subkeys of the first round. However, it is not suitable for the 6-round attack on ARIA, since if we want to apply the higher order integrals, there should be some subset A of $\mathbb{F}_{2^n}^m$ such that the linear transform is also a permutation on A. For example, the MixColumn operation of AES round transformation is a permutation on each column of the matrix. Since the P layer of ARIA doesn't have this property, higher order integral is invalid for ARIA.

We have implemented the 4-round attack on personal computer and all the round keys of the last round can be recovered in a few seconds. To the best of our knowledge, this is the first result that can attack 4-round ARIA in a real world. Table 2 lists some known cryptanalysis results on ARIA, from which one can find that the 5-round attack presented in this paper has the lowest data complexity and time complexity comparing to the known results.

In previous integral distinguishers, we determine whether some given bytes are active, balanced or constant. However, in this paper, we find a integral distinguisher which is neither active, balanced nor constant, see Corollary 1. How to apply this type of distinguishers to other ciphers is under study.

In our analysis, we also find many other 3-round distinguishers, all of which have three active positions, thus they have no difference in attacking 4 and 5-round ARIAs. However, when applying 6-round attack, for some distinguishers one should guess at least 12 bytes(in this case, we call the corresponding distinguisher optimal distinguisher) and for others more than 12. Table 3 lists all the optimal 3-round integral distinguishers of ARIA. Besides, our experiments find that instead of using 4 S-boxes, if ARIA adopts only one S-box, the conclusion of Theorem 1 becomes that all the bytes of Y_3 are balanced; if the order of S-boxes is changed, the distinguishers changes accordingly. Thus it shows that the choice of S-box and different order of S-boxes do have influence on integral attacks.

Since 6-round ARIA is not immune to integral attack, according to ref.[14], 6-round ARIA is not immune to interpolation attack either.

Acknowledgement

The authors wish to thank Ruilin Li and the anonymous reviewers for their useful comments.

The work in this paper is partially supported by the Natural Science Foundation of China(No: 60803156) and the open research fund of State Key Laboratory of Information Security(No: 01–07).

References

1. Kwon, D., Kim, J., Park, S., Sung, S.H., et al.: New Block Cipher: ARIA. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 432–445. Springer, Heidelberg (2004)

2. Koo, B.W., Jang, H.S., Song, J.H.: Constructing and Cryptanalysis of a 16×16 Binary Matrix as a Diffusion Layer. In: Chae, K.-J., Yung, M. (eds.) WISA 2003. LNCS, vol. 2908, pp. 489–503. Springer, Heidelberg (2004)
3. Biryukov, A., De Canniere, C., Lano, J., Ors, S.B., Preneel, B.: Security and Performance Analysis of Aria. Version 1.2., January 7 (2004)
4. National Security Research Institute, Korea. Specification of ARIA. Version 1.0. (January 2005)
5. Wu, W., Zhang, W., Feng, D.: Impossible differential cryptanalysis of Reduced-Round ARIA and Camellia. Journal of Compute Science and Technology 22(3), 449–456 (2007)
6. Li, R., Sun, B., Zhang, P., Li, C.: New Impossible Differentials of ARIA. Cryptology ePrint Archive, Report 2008/227 (2008), `http://eprint.iacr.org/`
7. Fleischmann, E., Gorski, M., Lucks, S.: Attacking Reduced Rounds of the ARIA Block Cipher. To appear in WEWoRC 2009 (2009); Cryptology ePrint Archive, Report 2009/334, `http://eprint.iacr.org/`
8. Daemen, J., Knudsen, L.R., Rijmen, V.: The Block Cipher Square. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)
9. Lucks, S.: The Saturation Attack — A Bait for Twofish. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 1–15. Springer, Heidelberg (2002)
10. Biryukov, A., Shamir, A.: Structural Cryptanalysis of SASAS. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 394–405. Springer, Heidelberg (2001)
11. Knudsen, L.R., Wagner, D.: Integral Cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
12. Z'aba, M.R., Raddum, H., Henricksen, M., Dawson, E.: Bit-Pattern Based Integral Attack. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 363–381. Springer, Heidelberg (2008)
13. Sun, B., Qu, L., Li, C.: New Cryptanalysis of Block Ciphers with Low Algebraic Degree. In: Dunkelman, O. (ed.) Fast Software Encryption. LNCS, vol. 5665, pp. 180–192. Springer, Heidelberg (2009)
14. Sun, B., Li, R., Li, C.: SQUARE attack on Block Ciphers with Low Algebraic Degree. To appear in Science in China, Ser. F-Inf. Sci.
15. Hatano, Y., Sekine, H., Kaneko, T.: Higher Order Differential Attack of *Camellia* (II). In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 129–146. Springer, Heidelberg (2003)
16. Daemen, J., Rijmen, V.: The Design of Rijndael: AES — The Advanced Encryption Standard, Information Security and Cryptography. Springer, Heidelberg (2002)
17. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.: Improved Cryptanalysis of Rijndael. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer, Heidelberg (2001)

Cryptanalysis of the ESSENCE Family of Hash Functions[⋆]

Nicky Mouha[1,2,⋆⋆], Gautham Sekar[1,2,⋆⋆⋆], Jean-Philippe Aumasson[3,†], Thomas Peyrin[4], Søren S. Thomsen[5], Meltem Sönmez Turan[6], and Bart Preneel[1,2]

[1] Department of Electrical Engineering ESAT/SCD-COSIC, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium
[2] Interdisciplinary Institute for BroadBand Technology (IBBT), Belgium
[3] FHNW, Windisch, Switzerland
[4] Ingenico, France
[5] Department of Mathematics, Technical University of Denmark, Matematiktorvet 303S, DK-2800 Kgs. Lyngby, Denmark
[6] Computer Security Division, National Institute of Standards and Technology, USA
{nicky.mouha,gautham.sekar,bart.preneel}@esat.kuleuven.be,
jeanphilippe.aumasson@gmail.com, thomas.peyrin@gmail.com,
ssth@win.dtu.dk, meltem.turan@nist.gov

Abstract. ESSENCE is a family of cryptographic hash functions, accepted to the first round of NIST's SHA-3 competition. This paper presents the first known attacks on ESSENCE. We present a semi-free-start collision attack on 31 out of 32 rounds of ESSENCE-512, invalidating the design claim that at least 24 rounds of ESSENCE are secure against differential cryptanalysis. We develop a novel technique to satisfy the first nine rounds of the differential characteristic. Non-randomness in the outputs of the feedback function F is used to construct several distinguishers on a 14-round ESSENCE block cipher and the corresponding compression function, each requiring only 2^{17} output bits. This observation is extended to key-recovery attacks on the block cipher. Next, we show that the omission of round constants allows slid pairs and fixed points to be found. These attacks are independent of the number of rounds. Finally, we suggest several countermeasures against these attacks, while still keeping the design simple and easy to analyze.

Keywords: Cryptanalysis, hash function, ESSENCE, semi-free-start collision, distinguisher, key-recovery, slide attack.

[⋆] This work was supported in part by the IAP Program P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II.
[⋆⋆] This author is funded by a research grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).
[⋆⋆⋆] This author is supported by an ADAPID project.
[†] Supported by the Swiss National Science Foundation under project no. 113329.

1 Introduction

Recent attacks by Wang et al. on the widely used hash functions MD4 [17], MD5 [18], RIPEMD [17] and SHA-1 [19], as well as other hash functions, show that collisions for these hash functions can be found much faster than expected by the birthday paradox [16].

In search for a new secure hash function standard, NIST announced the SHA-3 hash function competition [13]. The ESSENCE family of cryptographic hash functions, designed by Martin [9], advanced to the first round of this competition. It is a family of block cipher-based hash functions using the Merkle-Damgård mode of operation. The ESSENCE family uses simple algorithms that are easily parallelizable and well-established mathematical principles. ESSENCE comes with a proof of security against linear and differential cryptanalysis, that until this paper remained unchallenged.

First, we describe several undesired properties of the ESSENCE L function. These can be used to build a semi-free-start collision attack [12, pp. 371–372] on 31 out of 32 rounds of the ESSENCE-512 compression function using a differential characteristic. This directly invalidates the design claim that at least 24 rounds of ESSENCE are resistant against differential cryptanalysis [9]. To build our attack, we describe a novel technique to satisfy the conditions imposed by the characteristic in the first nine rounds. We do not know of a similar technique in existing literature.

Then, we find that the ESSENCE compression functions use a non-linear feedback function F that is unbalanced. We first exploit this to build efficient distinguishers on 14-round versions of the ESSENCE block ciphers as well as the compression functions. These distinguishers require only 2^{17} output bits. We then show how to use these results to recover the key with a few known plaintexts and a computational effort less than that of exhaustive search. We also show that, under some circumstances, the attacks on 14-round ESSENCE could be extended to the full 32-round block cipher and compression function.

Following this, we observe that the omission of round constants in ESSENCE leads to several attacks that cannot be prevented by increasing the number of rounds. A slide attack can be applied to any number of rounds of the ESSENCE compression function. We also find fixed points for any number of rounds of the ESSENCE block cipher, that lead to a compression function output of zero.

ESSENCE was not qualified to the second round of the SHA-3 competition; however, its appealing features (like design simplicity and hardware efficiency) make any effort on tweaking it appear worthwhile. Therefore, in this paper, we also suggest some countermeasures to thwart the aforesaid attacks.

In later work, Naya-Plasencia et al. [14] present different results on ESSENCE. Our paper presents not only differential cryptanalysis but also distinguishing attacks and slide attacks. Furthermore, some of our techniques can easily be generalized to other block ciphers and hash functions.

The paper is organized as follows. Section 2 describes the compression function of ESSENCE. In Sect. 3, we define and calculate the branching number of the linear L function for both linear and differential cryptanalysis. As the branching number turns out to be quite low, we use this observation to build a semi-free-start collision attack for 31 out of 32 rounds in Sect. 4. To satisfy the first nine rounds of the differential characteristic of the semi-free-start collision attack, we develop a technique in Sect. 5. Our distinguishers that exploit the weakness of F function are presented in Sect. 6. In the same section, we also show how our distinguishing attacks can be converted into key-recovery attacks on the block ciphers. Following this, we show how the omission of round constants allows us to find slid pairs (Sect. 7) and fixed points (Sect. 8) for any number of rounds. Finally, Sect. 9 enlists our countermeasures and Sect. 10 concludes the paper.

2 Description of the Compression Function of ESSENCE

The inputs to the compression function of ESSENCE are an eight-word chaining value and an eight-word message block, where each word is 32 or 64 bits in length, for ESSENCE-224/256 and ESSENCE-384/512 respectively. The compression function uses a permutation E, that in turn uses a nonlinear feedback function F, a linear transformation L, some XORs and word shifts.

The message block $m = (m_0, \ldots, m_7)$ forms the initial value of an eight-word state $k = (k_0, \ldots, k_7)$. In the case of the block cipher, m is the key $k = (k_0, \ldots, k_7)$. Similarly, the chaining value $c = (c_0, \ldots, c_7)$ is the initial chaining value of an eight-word state $r = (r_0, \ldots, r_7)$. In the case of the block cipher, c is the plaintext. Both states are iterated N times. The designer recommends N to be a multiple of 8, $N \geq 24$ for resistance to differential and linear cryptanalysis and $N = 32$ as a measure of caution [9]. Figure 1 illustrates one round of ESSENCE. The compression function uses a Davies-Meyer feed-forward (see Fig. 2). That is, at the end of N rounds, the value $r_7||r_6||r_5||r_4||r_3||r_2||r_1||r_0$ is XORed with the initial chaining value. The result is the $r_7||r_6||r_5||r_4||r_3||r_2||r_1||r_0$ for the next iteration.

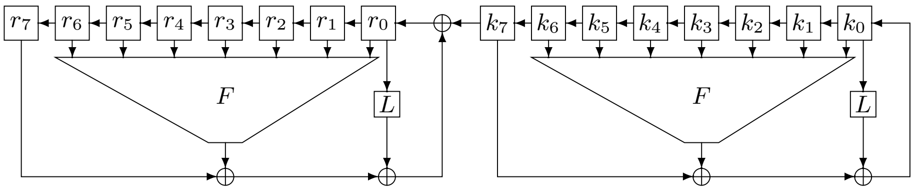


Fig. 1. One round of ESSENCE; each r_n and k_n $(n = 0, \ldots, 7)$ is a 32- or 64-bit word

3 Branching Number of the L Function

The L function of ESSENCE is a linear transformation from 32 (or 64) bits to 32 (or 64) bits and it is the only component that causes diffusion between the

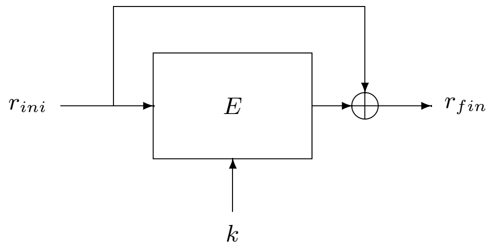


Fig. 2. The compression function of ESSENCE; E is the round function of ESSENCE when iterated N times, k denotes the message block, r_{ini} denotes the initial value of $r_7||r_6||r_5||r_4||r_3||r_2||r_1||r_0$ and r_{fin} denotes the value of r for the next iteration

different bit positions of a word. Therefore, its properties are very important for both linear and differential cryptanalysis.

In this section, we focus on the branching number of the L function for both linear and differential cryptanalysis. Let the branching number for differential cryptanalysis be the minimum number of non-zero input and output differences for the L function. These branching numbers are 10 and 16 for the 32-bit and 64-bit L functions respectively. If we were to consider only one-bit differences at either the input or the output of L, these numbers would be 14 and 27 respectively.

The branching number for linear cryptanalysis can be defined as the (non-zero) minimum number of terms in a linear equation relating the input and output bits of the L function. These branching numbers are 10 and 17 for the 32-bit and 64-bit L function respectively. Considering linear relations that involve only one bit at the input or one bit at the output, we would find branching numbers of 12 and 26 respectively.

Although one-bit differences are spread out well by the L function, this is clearly not the case for differences in multiple bits. This problem is most severe with the 64-bit L function. In the next section, we will show how this property can be used to build narrow trails for all digest sizes of ESSENCE.

4 A 31-Round Semi-Free-Start Collision Attack For ESSENCE-512

In this section, we will focus only on ESSENCE-512 for the sake of brevity and clarity. As the strategy is not specific to any particular digest size, these results can easily be generalized to all digest sizes of ESSENCE.

Although the ESSENCE L function spreads out one-bit differences very well, the previous section showed that this is not the case for differences in multiple bits. We therefore propose to use the differential characteristic of Table 1, to obtain 31-round semi-free-start collisions for ESSENCE-512.

To construct narrow trails, we use the non-zero difference A with the lowest possible Hamming weight. For this difference, we impose the condition

$(\neg A) \wedge L(A) = 0$, where $\neg$ represents the negation operation and all logical operations are to be performed bitwise. This can be formulated as follows: if there is a difference at the output of the L function at a particular bit position, there must be a difference at the input of L at this bit position as well. This requirement is necessary, as the F function can absorb or propagate an input difference at the output, but if no input difference is present, then there won't be an output difference either at this particular bit position. This places a restriction on the output difference of the L function for this bit position.

There exist exactly 8 differences A with a weight of 17 and lower weight differences A do not exist. These differences are available in Appendix A, along with a method to calculate them efficiently.

The last two columns of Table 1 provide an estimate of the probability that the characteristic is satisfied for every round. For these, we have assumed that the F function propagates or absorbs an input difference with equal probability. An more accurate calculation of these probabilities takes into account that the shift register causes input values of the F function to be reused.

We find that this probability is different for bit positions where A and $L(A)$ both contain a difference, and for bit positions where only A contains a difference. As such, of all differences A with weight 17, we select the difference that has the highest weight of $L(A)$. Five such differences exist, and we arbitrarily select the difference with the smallest absolute value, $A =$ `0A001021903036C3`. The corresponding $L(A) =$ `0200100180301283` has weight 11. As such, we find that rounds 10 to 16 of the key schedule, and rounds 18 to 24 of the compression function, each have a probability of $2^{-8.415 \cdot 6 - 8 \cdot 11} = 2^{-138.49}$. For rounds 18 to 23 of the key schedule, we find a probability of $2^{-7.193 \cdot 6 - 7 \cdot 11} = 2^{-120.16}$.

To find semi-free-start collisions, we first search for a message pair that satisfies the key expansion characteristic, and then afterwards search for a chaining value pair that satisfies the compression function characteristic. These two searches can be decoupled, as the chaining value does not influence the key schedule. As such, the probabilities for the message pairs and IV pairs can be summed up instead of multiplied.

As will be shown in the next section, only two round function calls are required to find a message (or IV) that satisfies the first nine rounds of the key expansion (or compression function). To find a pair of messages (or IVs) that satisfy the differential characteristic, we use the same depth-first search algorithm that was introduced for SHA-1 in [2]. The memory requirements of this search algorithm are negligible. We assume that the cost of visiting a node in this search tree is equivalent to one round function call, or 2^{-5} compression function calls. The complexity calculation of [2] then shows that a 31-round semi-free-start collision can be found using the characteristic of Table 1 after $2^{138.49+120.16+1-5} + 2^{138.49+1-5} = 2^{254.65}$ equivalent compression function calls. This is faster than a generic birthday attack, which requires about 2^{256} compression function evaluations.

Table 1. A 31-round semi-free-start collision differential characteristic for the ESSENCE-512 compression function; differences from R to Y are arbitrary, 0 represents the zero difference, $A =$ `0A001021903036C3`

Round	Register R	Register K	Pr for CV	Pr for m
0	0 0 0 0 0 0 0 0	A 0 0 0 0 0 0 0	1	1
1	0 0 0 0 0 0 0 A	0 0 0 0 0 0 0 A	1	1
2	0 0 0 0 0 0 A 0	0 0 0 0 0 0 A 0	2^{-17}	2^{-17}
3	0 0 0 0 0 A 0 0	0 0 0 0 0 A 0 0	2^{-17}	2^{-17}
4	0 0 0 0 A 0 0 0	0 0 0 0 A 0 0 0	2^{-17}	2^{-17}
5	0 0 0 A 0 0 0 0	0 0 0 A 0 0 0 0	2^{-17}	2^{-17}
6	0 0 A 0 0 0 0 0	0 0 A 0 0 0 0 0	2^{-17}	2^{-17}
7	0 A 0 0 0 0 0 0	0 A 0 0 0 0 0 0	2^{-17}	2^{-17}
8	A 0 0 0 0 0 0 0	A 0 0 0 0 0 0 0	2^{-17}	2^{-17}
9	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 A	1	1
10	0 0 0 0 0 0 0 0	0 0 0 0 0 0 A 0	1	2^{-17}
11	0 0 0 0 0 0 0 0	0 0 0 0 0 A 0 0	1	2^{-17}
12	0 0 0 0 0 0 0 0	0 0 0 0 A 0 0 0	1	2^{-17}
13	0 0 0 0 0 0 0 0	0 0 0 A 0 0 0 0	1	2^{-17}
14	0 0 0 0 0 0 0 0	0 0 A 0 0 0 0 0	1	2^{-17}
15	0 0 0 0 0 0 0 0	0 A 0 0 0 0 0 0	1	2^{-17}
16	0 0 0 0 0 0 0 0	A 0 0 0 0 0 0 0	1	2^{-17}
17	0 0 0 0 0 0 0 A	0 0 0 0 0 0 0 A	1	1
18	0 0 0 0 0 0 A 0	0 0 0 0 0 0 A 0	2^{-17}	2^{-17}
19	0 0 0 0 0 A 0 0	0 0 0 0 0 A 0 0	2^{-17}	2^{-17}
20	0 0 0 0 A 0 0 0	0 0 0 0 A 0 0 0	2^{-17}	2^{-17}
21	0 0 0 A 0 0 0 0	0 0 0 A 0 0 0 0	2^{-17}	2^{-17}
22	0 0 A 0 0 0 0 0	0 0 A 0 0 0 0 0	2^{-17}	2^{-17}
23	0 A 0 0 0 0 0 0	0 A 0 0 0 0 0 0	2^{-17}	2^{-17}
24	A 0 0 0 0 0 0 0	A 0 0 0 0 0 0 R	2^{-17}	1
25	0 0 0 0 0 0 0 0	0 0 0 0 0 0 R S	1	1
26	0 0 0 0 0 0 0 0	0 0 0 0 0 R S T	1	1
27	0 0 0 0 0 0 0 0	0 0 0 0 R S T U	1	1
28	0 0 0 0 0 0 0 0	0 0 0 R S T U V	1	1
29	0 0 0 0 0 0 0 0	0 0 R S T U V W	1	1
30	0 0 0 0 0 0 0 0	0 R S T U V W X	1	1
31	0 0 0 0 0 0 0 0	R S T U V W X Y	1	1

5 Finding Message Pairs for the First Nine Rounds

To find messages that satisfy the first few rounds of the characteristic, single-message modification [18] cannot be used. This is because the entire message is loaded into the r-registers before the round function is applied, instead of injecting one message word every round. We therefore propose to use another technique, that turns out to be even more efficient than single-message modification. This concept is explained for the key schedule only, as it is completely analogous for the compression function.

In this section, we will adopt a stream-based notation for the round function. Denote the initial eight-word state $(k_7, k_6, k_5, k_4, k_3, k_2, k_1, k_0)$ as $(x_{-2}, x_{-1}, x_0, x_1, x_2, x_3, x_4, x_5)$. After clocking one for one round, the value of the register k_0 is represented by x_6, and so on. In this text, we will not make a distinction between linear and affine equations, and use the term "linear equation" for any equation that contains no monomials of a degree more than one.

Finding a pair of messages that satisfy the characteristic, can be seen as solving a set of non-linear equations defined by the round function. Solving a set of non-linear equations is a difficult problem in general. This is even more the case as we are not looking for a single solution, but for a very large set of solutions.

What we can do, however, is impose linear conditions on the variables x_0 to x_{12}, in such a way that the round function behaves as a linear function. We then obtain a set of linear equations, of which every solution corresponds to a message pair that follows the first nine rounds of the characteristic. Enumerating the solutions of this linear space has a negligible computation cost compared to a round function evaluation.

For every solution, we have to apply the round function twice to obtain x_{13} and x_{14}. These are guaranteed to follow the characteristic as well. They serve as a starting point to satisfy the conditions of the remaining characteristic in a probabilistic way. After reaching round 31, we can calculate x_{-2} and x_{-1} by applying two inverse round functions. These values will always satisfy the characteristic.

Let $A[j]$ denote the j-th significant bit ($j = 0$ denotes the least significant bit or LSB) of A. The only non-linear function of ESSENCE is the F function. As the F function operates on every bit in parallel, the linear conditions that have to be added, depend on the values $A[j]$ and $L(A)[j]$ at every bit position j. The equations we use are given in Appendix B. Note that for bit positions j where $A[j] = 0$, it is not a problem if $x_0[j]$ or $x_{12}[j]$ are represented by a non-linear expression, as these bits are not involved in any of the linear conditions anyway.

As the equations in Appendix B show, we need to add 10 equations for every bit position j where $A[j] = 1$, and 6 equations if $A[j] = 0$. Also, to represent the 64-bit values x_8 to x_{12} resulting from the round function, we need to add $5 \cdot 64$ additional equations for outputs of the round function. In total, we obtain a set of $10 \cdot 17 + 6 \cdot (64 - 17) + 5 \cdot 64 = 772$ linear equations in $13 \cdot 64 = 832$ binary variables.

We build this system of equations by successively adding $10+5 = 15$ or $6+5 = 11$ more equations for every bit position j. With some small probability, the system of equations becomes inconsistent. If this happens, we add a different set of linear equations for this bit position. Even this may fail with some probability, in which case we add a linearization of the F function using $7 + 5$ instead of $6 + 5$ equations for this particular bit position. This may or may not decrease the number of solutions slightly, but it allows us to avoid backtracking.

For one particular run, using only the equations mentioned in Appendix C, we find a consistent system of 772 linear equations in 832 binary variables. The number of linearly independent equations turns out to be 771. As such, we have found $2^{832-771}/2 = 2^{60}$ pairs of messages that satisfy the first 9 rounds. We divide by two to avoid counting the same pair twice. If more than 2^{60} pairs of messages needed, we can simply run this program again to find the next set of messages. As including these 771 equations would use up a lot of space, we give only one of the 2^{60} message pairs in Table 6.

This technique is very similar to the techniques of multi-message modification [18], tunneling [8], neutral bits [1] and the amplified boomerang attack [7]. These 2^{60} messages correspond to 60 auxiliary differential paths for the amplified boomerang attack. No results are known to us where these auxiliary differential paths were also obtained in a fully automated way.

6 Distinguishing Attacks

Our motivational observation is that the non-linear feedback function F is unbalanced. Exploiting this, we first construct distinguishers on 14-round ESSENCE (both the block cipher and the compression function) and then for the full 32-round ESSENCE. Towards the end of this section, we present key-recovery attacks on the ESSENCE family of block ciphers. These attacks can be seen as an immediate consequence of our distinguishing attacks.

6.1 Weakness in the Feedback Function of ESSENCE

In [9], the designer notes that the security of the algorithms is heavily dependent on F, as it is the only nonlinear function in ESSENCE. This gave us some motivation to study the properties of F. The function F takes seven 32-bit or 64-bit words (say, $a, \ldots, g$) as inputs and produces a 32-bit or 64-bit word as the output. The function works in a bitsliced manner. The exact description of F is largely irrelevant to our analysis; hence, we refer the interested reader to Appendix D.

Let $F(a,b,c,d,e,f,g)[j]$ denote the j-th significant bit ($j = 0$ denotes the LSB) of $F(a,b,c,d,e,f,g)$. Our motivational observation is the following (confirmed both experimentally and from the tables in Appendix D of [9]).

Observation 1: If $a, \ldots, g$ are uniformly distributed, then

$$Pr[F(a,b,c,d,e,f,g)[j] = 0] = \frac{1}{2} + \frac{1}{2^7} \ . \qquad (1)$$

6.2 Distinguishers on 14-Round ESSENCE

In this section, we use Observation 1 to build distinguishers on 14 rounds of ESSENCE. First, we consider the block cipher, then the compression function.

Let $k_n[j]$, $r_n[j]$ and $L(r_n)[j]$ respectively denote the j-th significant bits ($j = 0$ denotes the LSB) of k_n, r_n and $L(r_n)$. In the beginning, suppose the key k and the initial value r are such that $k_0[0] = r_0[0]$. Then, after 7 rounds, $k_7[0] = r_7[0]$. Now, if after the 7th round, $L(r_0)[0] = 0$ and $F(r_6, r_5, r_4, r_3, r_2, r_1, r_0)[0] = 0$ (from Observation 1, this occurs with $0.5 + 2^{-7}$ probability[1]), then after the 8th round, we will have $r_0[0] = 0$. Note that the condition $L(r_0)[0] = 0$ after the 7th round is the same as the condition $L(r_1)[0] = 0$ after the 8th round. Therefore, when the key and the plaintext are initially related in the form $k_0[0] = r_0[0]$, and when the outputs after 8 rounds satisfy the condition $L(r_1)[0] = 0$ (this occurs with probability 1/2), then $Pr[r_0[0] = 0] = 1/2 + 2^{-7}$. Now, r_0 and r_1 after the 8th round are respectively equal to r_6 and r_7 after the 14th round. Hence, when the key and the plaintext are related in the form $k_0[0] = r_0[0]$, and when the outputs after 14 rounds satisfy the condition $L(r_7)[0] = 0$, then

$$Pr[r_6[0] = 0] = \frac{1}{2} + \frac{1}{2^7} \ . \tag{2}$$

6.3 The Distinguisher

A distinguisher is an algorithm that distinguishes one probability distribution from another. In cryptography, a distinguisher is an algorithm that distinguishes a stream of bits from a stream of bits uniformly distributed at random (i.e., bitstream generated by an ideal cipher).

Our distinguisher on ESSENCE is constructed by collecting n outputs $r_6[0]$, after 14 rounds, generated by as many keys (so that the n samples are independent) such that $k_0[0] = r_0[0]$ initially. Let D_0 and D_1 denote the distributions of the outputs from 14-round ESSENCE block cipher and a random permutation, respectively. Given $L(r_7)[0] = 0$, let p_0 and p_1 respectively denote the probability that $r_6[0] = 0$ holds given the outputs are collected from 14-round ESSENCE and the probability that $r_6[0] = 0$ holds given the outputs are generated by a random source. That is, $p_0 = 1/2 + 2^{-7}$ (from (2)) and $p_1 = 1/2$. Then, $\mu_0 = n \cdot p_0$ and $\mu_1 = n \cdot p_1$ are the respective means of D_0 and D_1. Similarly, $\sigma_0 = \sqrt{n \cdot p_0 \cdot (1 - p_0)}$ and $\sigma_1 = \sqrt{n \cdot p_1 \cdot (1 - p_1)}$ denote the respective standard deviations of D_0 and D_1. When n is large, both these binomial distributions can be approximated with the normal distribution. Now, if $|\mu_0 - \mu_1| > 2(\sigma_0 + \sigma_1)$, i.e., $n > 2^{16}$, the output of the cipher can be distinguished from a random permutation with a success probability of 0.9772 (since the cumulative distribution function of the normal distribution gives the value 0.9772 at $\mu + 2\sigma$) provided $L(r_7)[0] = 0$. To test whether n is large enough for the normal approximation to the binomial distribution to hold, we use a commonly employed rule of thumb: $n \cdot p > 5$ and $n \cdot (1 - p) > 5$, where $p \in \{p_0, p_1\}$. A simple calculation proves that both the above inequalities hold when $n = 2^{16}$. Since the condition $L(r_7)[0] = 0$

[1] The bit $L(r_0)[0]$ is the XOR-sum of $r_0[0]$ and several other bits of r_0. We assume that all $r_0[j]$ are independent and uniformly distributed. Then the condition $L(r_0)[0] = 0$ does not affect $Pr[r_0[0] = 0]$ and therefore the bias in $Pr[F(r_6, r_5, r_4, r_3, r_2, r_1, r_0)[0] = 0]$ is also unaffected.

holds with 0.5 probability, we need to generate $2 \cdot 2^{16} = 2^{17}$ samples of $r_6[0]$ from as many keys (such that $k_0[0] = r_0[0]$ initially) to build the distinguisher with a success probability of 0.9772. Our simulations support this result.

6.4 Distinguishers Using Biases in Other Bits

Since the function F operates on its input bits in a bitsliced manner, it is easy to see that the distinguisher presented for the LSB of r_6 also works for more significant bits. In other words, if initially $k_0[j] = r_0[j]$, for any j in $\{0, \ldots, 31\}$, then with 2^{16} samples of $r_6[j]$ at the the end of 14 rounds, it is possible to distinguish 14-round ESSENCE block cipher from a random permutation with a success probability of 0.9772.

6.5 Distinguishers for the Compression Function

The ESSENCE compression function is a Davies-Meyer construction in which the output of the block cipher is XORed with the initial chaining value. In other words, the output of the compression function is the XOR-sum of the values of $r_7||r_6||r_5||r_4||r_3||r_2||r_1||r_0$ before and after applying the permutation E. This XOR-sum is the chaining value $r_7||r_6||r_5||r_4||r_3|| r_2||r_1||r_0$ for the next iteration. As we assume that an attacker can observe both the chaining value input and the compression function output, it is trivial to undo the Davies-Meyer feedforward and apply the distinguishers of the 14-round block cipher.

These observations are extended to 32-round ESSENCE in Appendix E.

6.6 Key-Recovery Attacks

In this section, we show that the distinguishing attacks on the ESSENCE family of block ciphers can be converted into key-recovery attacks.

Let us say that we have n known plaintexts. Considering that the plaintexts are initially loaded directly into the r-registers [10], we expect $n/2$ plaintexts to have $r_0[j] = 0$. Without loss of generality, let us consider this partition of the plaintext space where $r_0[j] = 0$. Now, from our analysis in Sect. 6.2, we can collect statistics on $L(r_7)[j] \oplus r_6[j]$ at the end of the 14 rounds and observe its tendency for sufficiently large n — if $L(r_7)[j] \oplus r_6[j] = 0$ more often, then the key bit $k_0[j] = 0$; likewise, if $L(r_7)[j] \oplus r_6[j] = 1$ more often, then the key bit $k_0[j] = 1$ (the results are swapped if we begin with plaintexts in which $r_0[j] = 1$).

Using a similar analysis, we can recover the rest of the key bits in k_0. The number of known plaintexts required is 2^{15}. This is obtained as follows, using standard linear cryptanalysis [11]. We are interested in finding whether, after 14 rounds, the number of times that $L(r_7)[j] \oplus r_6[j] = 0$ holds is greater than $n/4$. Accordingly, we determine the key bit $k_0[j]$. Unlike in the distinguishing attacks, a confidence interval for the uniform distribution is not required. From [11] we obtain that the success probability of this method is 0.9772 when $n/2 = |p - 1/2|^{-2}$, where p is the probability that $L(r_7)[j] \oplus r_6[j] = 0$ (or 1). Substituting

$p = 1/2 \pm 2^{-7}$ in the above formula for n, we get $n = 2^{15}$. It follows that the probability that this recovered key word (k_0) is correct is $(0.9772)^{32} \approx 0.48$. The other 224 bits of the key can be exhaustively searched. Thereby, we expect that $2^{224}/0.48 \approx 2^{225.1}$ keys have to be tested before the correct key is obtained with guaranteed success. This key-recovery attack can also be applied on the block cipher of ESSENCE-224 (which is identical to the block cipher of ESSENCE-256) with the same complexities. For the block ciphers of ESSENCE-384/512, we require 2^{15} known plaintexts and a computational effort equivalent to testing $2^{448}/(0.9772)^{64} \approx 2^{450.1}$ keys (where exhaustive search requires testing 2^{512} keys) for guaranteed success.

These observations are extended to 32-round ESSENCE in Appendix F.

7 Slide Attack

In this part of the study, we provide an efficient method to find two inputs (c, m) and (c', m') such that their output (after feed-forward) r and r' are shifted versions of each other; i.e., if $r_i = r'_{i+1}$ for $0 \leq i < 7$.

The necessary conditions on (c, m) and (c', m') are

1. $c_i = c'_{i+1}$ for $0 \leq i \leq 7$,
2. $c'_0 = m_7 \oplus c_7 \oplus F(c_6, \ldots, c_0) \oplus L(c_0)$,
3. $m_i = m'_{i+1}$ for $0 \leq i \leq 7$,
4. $m'_0 = m_7 \oplus F(m_6, \ldots, m_0) \oplus L(m_0)$.

If these conditions hold, then after 32 rounds (and XORing with the initial value), the output of the compression function satisfies $r_i = r'_{i+1}$ for $0 \leq i < 7$.

As an example, let $m_i = 0$ for all i. Then we must choose $m'_i = 0$ for all $i > 0$, and $m'_0 = 1^n$ where 1^n represents the 32-bit or 64-bit unsigned integer of which all bits are set. Let $c_i = 0$ for all i, let $c'_i = 0$ for all $i > 0$, and let $c'_0 = 1^n$. Then, the two outputs of the compression function (with $N = 32$) are:

c	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
c'	FFFFFFFF	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m'	FFFFFFFF	00000000	00000000	00000000	00000000	00000000	00000000	00000000
R	6B202EF2	BB610A07	97E43146	9BD34AE3	C8BC7CBF	B8EE4A3C	B6118DC5	775F7BBF
R'	C07ABCFA	6B202EF2	BB610A07	97E43146	9BD34AE3	C8BC7CBF	B8EE4A3C	B6118DC5

For every choice of (c, m), an input (c', m') such that this property on the compression function outputs is obtained can be found in time equivalent to about one compression function evaluation. Hence, in total about 2^{512} pairs of inputs producing slid pairs can be found by the above method. This observation can easily be extended to slide the output by $2, 3, \ldots, 7$ steps.

7.1 Slid Pairs with Identical Chaining Values

It is also possible to find slid pairs with $c = c'$. Let the initial state of the register R be of the form $(c_0, c_0, \ldots, c_0)$, where c_0 is selected randomly. For a message block m of the form $(m_0, m_1, \ldots, m_7)$ where $m_7 = F(c_0, \ldots, c_0) \oplus L(c_0)$ and the rest of the m_i's are selected arbitrarily, select m' as $(m'_0, m'_1, \ldots, m'_7)$, such that $m'_{i+1} = m_i$ for $i = 0, 1, 2, \ldots, 6$ and $m'_0 = m_7 \oplus F(m_6, \ldots, m_0) \oplus L(m_0)$. Then, the outputs of the compression function for m and m' also satisfy $r_i = r'_{i+1}$ for $0 \leq i < 7$. It is possible to select c in 2^{32} different ways, and for each selected c, we can choose $2^{7 \cdot 32}$ different message blocks, therefore the number of such slid pairs is 2^{256}. As an example, assume $c_0 =$ `243f6a88`, which is the truncated fractional part of π, and all "free" message words are zero.

c, c'	243F6A88 243F6A88 243F6A88 243F6A88 243F6A88 243F6A88 243F6A88 243F6A88
m	00000000 00000000 00000000 00000000 00000000 00000000 00000000 F6B1EB63
m'	094E149C 00000000 00000000 00000000 00000000 00000000 00000000 00000000
R	BE31AA01 EB6E9F07 EAD99889 6FE79B44 391CCD35 67FDB8B6 FC3AA0F6 6E80148E
R'	F86D77C6 BE31AA01 EB6E9F07 EAD99889 6FE79B44 391CCD35 67FDB8B6 FC3AA0F6

8 Fixed Points for the ESSENCE Block Cipher

If a fixed point for one round of the ESSENCE block cipher can be found, this automatically leads to a fixed point for all 32 steps of the block cipher. After applying the Davies-Meyer feed-forward, the resulting compression function output will then be zero.

If two different fixed points are found, this would lead to a free-start collision. This free-start collision is preserved after the output padding is applied.

For a fixed point for one round, $c_0 = c_1 = \ldots = c_7$ and $m_0 = m_1 = \ldots = m_7$ should hold. This is obvious: after one step, all register values move one place, but must have the same value as in the previous step to form a fixed point. Moreover, the round update functions should satisfy the following equations.

$$F(c_0, c_0, c_0, c_0, c_0, c_0, c_0) \oplus c_0 \oplus L(c_0) \oplus m_0 = c_0 \ ,$$
$$F(m_0, m_0, m_0, m_0, m_0, m_0, m_0) \oplus m_0 \oplus L(m_0) = m_0 \ .$$

Solving the equations, one gets the following values for ESSENCE-256 and ESSENCE-512:

	ESSENCE-256	ESSENCE-512
c_0	993AE9B9	D5B330380561ECF7
m_0	307A380C	10AD290AFFB19779

Using similar methods, we have found that the only fixed points for two, three or four rounds is the same fixed point for one round applied two, three or four times respectively. We have not been able to extend this result for more rounds. As such, we have not been able to find a free-start collisions using this technique. Depending how the compression function is used, however, it might be undesirable that we can easily find inputs that fix the compression function output to zero.

9 Measures to Improve the Security of ESSENCE

From the analysis in Sect. 3–6, it is clear that ESSENCE has weaknesses in L and F.

The concatenation of both the input and output of the L function can be seen as an error-correcting code with $[n, k] = [64, 32]$ or $[128, 64]$. The branching number is then equal to the error-correcting code of these dimensions with the highest minimum weight. Best known results from coding theory [6] can be used to construct an L function with a branching number for both linear and differential cryptanalysis of 12 or 22 respectively. Better codes may exist according to currently known upper bounds for the minimum weight, but have so far not been found.

A search can be made for variants of these codes (possibly with a slightly lower branching number) that satisfy all design criteria for the L function. Although the resulting function will always be linear, it may however not be possible to implement it as an LFSR.

In (5), the function F is in algebraic normal form (ANF). We know that the coefficient of the maximum degree monomial in this ANF is equal to the XOR-sum of all the entries in the truth table of F. To thwart the attacks in Sect. 6 and Appendix F, it is necessary that F is balanced. Discarding the maximum degree monomial is a possible solution.

Other countermeasures include increasing the number of rounds and adding round constants. In Sect. 7 and Sect. 8, we saw how the omission of round constants allowed slid pairs and fixed points to be found. Increasing the number of rounds does not thwart these attacks, but it increases the security margin against the semi-free-start collision attacks of this paper.

10 Conclusions and Open Problems

In this paper, we first presented a semi-free-start collision attack on 31 out of 32 rounds with a complexity of $2^{254.65}$ compression function evaluations. We find messages that satisfy the first nine rounds of the differential characteristic of the semi-free-start collision attack as the solution of a large set of linear equations. We found that six linear input conditions are sufficient to make F behave as a linear function in Table 5. It is an open problem if solutions using fewer equations exist.

We also presented a set of distinguishers on 14-round ESSENCE. The distinguishers can be applied to the block cipher as well as the compression function. Each of the distinguishers on 14-round ESSENCE requires 2^{17} output bits. The distinguishers work on all digest sizes of ESSENCE with the same complexity. It has also been shown how the distinguishing attacks can be turned into key-recovery attacks.

We then showed how the omission of round constants allowed slid pairs and fixed points to be found. This cannot be prevented by increasing the number of rounds.

Finally, we suggested some measures to improve the security of ESSENCE. These suggestions are rather preliminary and need to be worked on further in order to obtain a more secure family of hash functions.

Acknowledgments. The authors would like to thank Christophe De Cannière, Sebastiaan Indesteege, Gaëtan Leurent, Willi Meier, Tomislav Nad, María Naya-Plasencia, Vincent Rijmen and Andrea Röck for their useful comments and suggestions.

Special thanks go out to the designer of ESSENCE, Jason Worth Martin, who not only gave us useful feedback, but was also very supportive when we wanted to make these results public. He has verified the correctness of the results in this paper.

Part of this work was performed at the Hash Function Retreat, hosted by the Graz University of Technology as an initiative of the SymLab group of the ECRYPT II project. We are very grateful to Mario Lamberger, Florian Mendel, Tomislav Nad, Christian Rechberger, Vincent Rijmen and Martin Schläffer for their excellent organization of this retreat.

María Naya-Plasencia, Andrea Röck, Thomas Peyrin, Jean-Philippe Aumasson, Gaëtan Leurent and Willi Meier have obtained other, non-overlapping results on ESSENCE [14] in parallel with these results. Their paper uses different characteristics and another way of finding conforming messages.

References

1. Biham, E., Chen, R.: Near-Collisions of SHA-0. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)
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. Cramer, R. (ed.): EUROCRYPT 2005. LNCS, vol. 3494. Springer, Heidelberg (2005)
4. Dinur, I., Shamir, A.: Side Channel Cube Attacks on Block Ciphers. Cryptology ePrint Archive, Report 2009/127 (2009), http://eprint.iacr.org/
5. Dziembowski, S., Pietrzak, K.: Leakage-Resilient Cryptography. In: FOCS, pp. 293–302. IEEE Computer Society, Los Alamitos (2008)
6. Grassl, M.: Tables of Linear Codes and Quantum Codes (June 2008), http://www.codetables.de/
7. Joux, A., Peyrin, T.: Hash Functions and the (Amplified) Boomerang Attack. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 244–263. Springer, Heidelberg (2007)

8. Klima, V.: Tunnels in Hash Functions: MD5 Collisions Within a Minute. Cryptology ePrint Archive, Report 2006/105 (2006), http://eprint.iacr.org/
9. Martin, J.W.: ESSENCE: A Family of Cryptographic Hashing Algorithms. Submitted to the NIST SHA-3 hash function competition, http://www.math.jmu.edu/ martin/essence/Supporting_Documentation/essence_compression.pdf (2009/01/20)
10. Martin, J.W.: Personal Communication (2009)
11. Matsui, M.: Linear Cryptoanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
12. Menezes, A., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
13. 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 (2008/10/17)
14. Naya-Plasencia, M., Roeck, A., Peyrin, T., Aumasson, J.-P., Leurent, G., Meier, W.: Cryptanalysis of ESSENCE (2009) (Unpublished)
15. Pietrzak, K.: A Leakage-Resilient Mode of Operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2010)
16. Preneel, B.: Analysis and design of cryptographic hash functions. PhD thesis, Katholieke Universiteit Leuven (1993)
17. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer (ed.) [3], pp. 1–18
18. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer (ed.) [3], pp. 19–35
19. Wang, X., Yu, H., Yin, Y.L.: Efficient Collision Search Attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005)

A Finding the Lowest Weight Difference A

We wish to find a difference A that satisfies

$$(\neg A) \wedge L(A) = 0 \tag{3}$$

and

$$\text{hw}(A) \leq w \ , \tag{4}$$

where $\text{hw}(A)$ is the number of bits set in A and w is to be as small as possible.

We proceed as follows. Let w represent the (still unknown) weight of the lowest weight difference A. We then split w into two integers w_0 and w_1, such that $w_0 + w_1 = w$ and $|w_1 - w_0| \leq 1$. Let L^{-1} represent the inverse L function, such that $L^{-1}(L(x)) = x$. Let $M(x) = L(x) \oplus x$. The design of ESSENCE guarantees that M is invertible, as L is not allowed to have any eigenvalues in the ground field.

First step: We enumerate all x where $\mathrm{hw}(x) \leq w_0$. After calculating $A = L^{-1}(x)$, we check (3) and (4).

Second step: We enumerate all y where $\mathrm{hw}(y) \leq w_1$. After calculating $A = M^{-1}(y)$, we check if (3) and (4).

Equation (3) implies that the bit positions where $L(A)$ is 1, is always a subset of bit positions where A is 1. Therefore, we only have to consider two cases: the case where the set of bit positions where $L(A)$ is 1 contains no more than w_0 elements, and the case where the set bit positions where $L(A)$ is 0 and A is 1 contains not more than w_1 elements. As $w_0 + w_1 = w$, these two steps are guaranteed to find all A that satisfy (3) and (4). If no solution is found, we increase w by one and perform the two steps again, enumerating only the new values of x and y.

The total complexity of this search is $\left(\sum_{i=0}^{w_0} C_i^{64}\right) + \left(\sum_{j=0}^{w_1} C_j^{64}\right)$. As we find $w = 17$ here, the total number of 64-bit linear function evaluations is $\left(\sum_{i=0}^{8} C_i^{64}\right) + \left(\sum_{j=0}^{9} C_j^{64}\right) \approx 2^{35}$. This calculation can be performed in less than a minute on a recent desktop computer. The solutions are shown in Table 2.

Table 2. All differences A with $\mathrm{hw}(A) = 17$ that satisfy (3); there are no solutions where $\mathrm{hw}(A) < 17$ and (3)

A
2461822430680025
48C3044860D0004A
91860890C1A00094
0A001021903036C3
1400204320606D86
2800408640C0DB0C
5000810C8181B618
A001021903036C30

B Making *F* Behave as a Linear Transformation

We consider three separate cases, depending on the values of A and $L(A)$ for a particular bit position j.

If $A[j] = 1$, we can enumerate all possible input conditions, such that F behaves linearly and has the required differential behavior. Because we enumerate all possibilities, we obtain an optimal result: it is not possible to add fewer than 10 linear equations. All existing solutions where 10 linear equations are added, are shown in Table 3 (for $L(A)[j] = 1$) and Table 4 (for $L(A)[j] = 0$).

If $A[j] = 0$: the differential behavior is always satisfied: if there is no input difference, there will not be an output difference either. We found that adding 6 equations is sufficient. We do not rule out the possibility that fewer than 6 equations are sufficient. The solutions we found are given in Table 4.

Table 3. Making F linear and imposing the required differential behavior for position j where $A[j] = L(A)[j] = 1$ can be done by adding no more than 10 linear equations; exactly four such solutions exist

	Solution 1	Solution 2	Solution 3	Solution 4
	$x_0 \oplus x_2 = 1$	$x_1 = 1$	$x_1 = 1$	$x_1 = 1$
	$x_1 = 0$	$x_2 \oplus x_5 = 0$	$x_2 \oplus x_5 = 0$	$x_2 = 1$
	$x_3 = 1$	$x_2 \oplus x_7 = 1$	$x_2 \oplus x_7 = 1$	$x_3 = 0$
	$x_4 = 1$	$x_2 \oplus x_8 = 0$	$x_2 \oplus x_8 = 0$	$x_4 = 1$
	$x_5 = 1$	$x_2 \oplus x_9 = 0$	$x_2 \oplus x_9 = 0$	$x_5 = 1$
	$x_7 = 0$	$x_2 \oplus x_{12} = 1$	$x_3 = 0$	$x_7 = 0$
	$x_8 = 1$	$x_3 = 0$	$x_4 = 1$	$x_8 = 1$
	$x_9 = 0$	$x_4 = 1$	$x_{10} = 0$	$x_9 = 1$
	$x_{10} = 0$	$x_{10} = 0$	$x_{11} = 0$	$x_{10} = 0$
	$x_{12} = 1$	$x_{11} = 0$	$x_{12} = 1$	$x_{11} = 0$
$F(x_0, \ldots, x_6) =$	$x_6 \oplus 1$	$x_0 \oplus x_6$	$x_0 \oplus x_6$	$x_0 \oplus x_6$
$F(x_1, \ldots, x_7) =$	$x_2 \oplus 1$	$x_2 \oplus 1$	$x_2 \oplus 1$	0
$F(x_2, \ldots, x_8) =$	0	$x_2 \oplus 1$	$x_2 \oplus 1$	0
$F(x_3, \ldots, x_9) =$	0	x_5	x_5	1
$F(x_4, \ldots, x_{10}) =$	1	1	1	1
$F(x_5, \ldots, x_{11}) =$	1	0	0	0
$F(x_6, \ldots, x_{12}) =$	0	$x_7 \oplus 1$	0	$x_{12} \oplus 1$

Table 4. Making F linear and imposing the required differential behavior for position j where $A[j] = 1$ and $L(A)[j] = 0$ can be done by adding no more than 10 linear equations; exactly one such solution exists

	Solution 1
	$x_0 \oplus x_2 = 0$
	$x_1 = 0$
	$x_3 = 1$
	$x_4 = 1$
	$x_5 = 1$
	$x_7 = 0$
	$x_8 = 1$
	$x_9 = 0$
	$x_{10} = 0$
	$x_{12} = 1$
$F(x_0, \ldots, x_6) =$	1
$F(x_1, \ldots, x_7) =$	$x_2 \oplus 1$
$F(x_2, \ldots, x_8) =$	0
$F(x_3, \ldots, x_9) =$	0
$F(x_4, \ldots, x_{10}) =$	1
$F(x_5, \ldots, x_{11}) =$	1
$F(x_6, \ldots, x_{12}) =$	0

We will omit the index j, so that x_0 to x_{12} represent one-bit variables. The expressions $F(x_0, \ldots, x_6)$ and $F(x_6, \ldots, x_{12})$ are not added to the system of linear equations of the attack, as this is not necessary. They are only mentioned to show that their differential behavior is correct.

Table 5. Making F linear for position j where $A[j] = L(A)[j] = 0$ can be done by adding no more than 6 linear equations; at least six such solutions exist

	Solution 1	Solution 2	Solution 3	Solution 4	Solution 5	Solution 6
	$x_3 = 0$ $x_4 = 0$ $x_5 = 1$ $x_6 = 0$ $x_7 = 1$ $x_9 = 1$	$x_3 = 0$ $x_4 = 0$ $x_5 = 1$ $x_6 = 1$ $x_7 = 1$ $x_8 = 1$	$x_3 = 1$ $x_4 = 1$ $x_5 = 1$ $x_6 = 1$ $x_7 = 1$ $x_8 = 1$	$x_4 = 0$ $x_5 = 1$ $x_6 = 1$ $x_7 = 0$ $x_8 = 1$ $x_9 = 0$	$x_4 = 0$ $x_5 = 1$ $x_6 = 1$ $x_7 = 0$ $x_8 = 1$ $x_{10} = 0$	$x_4 = 0$ $x_5 = 1$ $x_6 = 1$ $x_7 = 0$ $x_8 = 1$ $x_{11} = 1$
$F(x_1, \ldots, x_7) =$	$x_1 \oplus 1$	x_2	x_1	$x_1 \oplus x_2 \oplus 1$	$x_1 \oplus x_2 \oplus 1$	$x_1 \oplus x_2 \oplus 1$
$F(x_2, \ldots, x_8) =$	$x_2 \oplus x_8 \oplus 1$	$x_2 \oplus 1$	x_2	$x_3 \oplus 1$	$x_3 \oplus 1$	$x_3 \oplus 1$
$F(x_3, \ldots, x_9) =$	$x_8 \oplus 1$	$x_9 \oplus 1$	x_9	0	x_9	x_9
$F(x_4, \ldots, x_{10}) =$	x_8	0	$x_9 \oplus x_{10} \oplus 1$	$x_{10} \oplus 1$	$x_9 \oplus 1$	$x_9 \oplus x_{10} \oplus 1$
$F(x_5, \ldots, x_{11}) =$	$x_8 \oplus x_{10} \oplus 1$	$x_{10} \oplus x_{11} \oplus 1$	$x_{10} \oplus x_{11} \oplus 1$	$x_{10} \oplus 1$	$x_9 \oplus 1$	$x_9 \oplus x_{10} \oplus 1$

Table 6. A message pair satisfying the first 9 rounds of the characteristic of Table 1

i	m_i	m'_i	$m_i \oplus m'_i$
0	FFFFFFFFFFFFFFFF	FFFFFFFFFFFFFFFF	0000000000000000
1	1A001021983836CB	1A001021983836CB	0000000000000000
2	5809832A1DEA2458	5809832A1DEA2458	0000000000000000
3	8AEF5FEBEB9FDAAB	8AEF5FEBEB9FDAAB	0000000000000000
4	32F9D8578015D297	32F9D8578015D297	0000000000000000
5	0D031372423B91AC	0D031372423B91AC	0000000000000000
6	B804AC08CD97E348	B804AC08CD97E348	0000000000000000
7	E8BB8E649DC3B35F	E2BB9E450DF3859C	0A001021903036C3

C A Message Pair for the First Nine Rounds

We give a message pair that satisfies the first 9 rounds of the characteristic of Table 1 in Table 6.

D The Feedback Function *F*

We denote the field of two elements by $\mathbb{F}_2$. The nonlinear feedback function, F, of ESSENCE-224/256 (respectively ESSENCE-384/512) takes seven 32-bit (respectively 64-bit) input words and outputs a single 32-bit (respectively 64-bit) word as follows:

$$\begin{aligned} F(a,b,c,d,e,f,g) = \; & abcdefg + abcdef + abcefg + acdefg + \\ & abceg + abdef + abdeg + abefg + \\ & acdef + acdfg + acefg + adefg + \\ & bcdfg + bdefg + cdefg + \\ & abcf + abcg + abdg + acdf + adef + \\ & adeg + adfg + bcde + bceg + bdeg + cdef + \end{aligned}$$

$$\begin{aligned}
&abc + abe + abf + abg + acg + adf + \\
&adg + aef + aeg + bcf + bcg + bde + \\
&bdf + beg + bfg + cde + cdf + def + \\
&deg + dfg + \\
&ad + ae + bc + bd + cd + \\
&ce + df + dg + ef + fg + \\
&a + b + c + f + 1 \ ,
\end{aligned} \tag{5}$$

where the multiplication and addition are taken in $\mathbb{F}_2$ (i.e., they are the same as bitwise XOR and bitwise AND, respectively).

E Distinguishing Attacks on the Full 32-Round ESSENCE-256

The attacks described in Sect. 6.2 can be easily extended to the full ESSENCE-256 block cipher. Let us suppose the key k and the plaintext are related such that after 18 rounds, $r_0[0] = k_0[0]$. Given this, using similar arguments as those used to derive (2), we obtain that at the end of 32 rounds, if $L(r_7)[0] = 0$, then

$$Pr[r_6[0] = 0] = \frac{1}{2} + \frac{1}{2^7} \ . \tag{6}$$

We can thus construct a distinguisher by collecting 2^{17} outputs $r_6[0]$, after 32 rounds, generated by as many keys (so that the samples are independent) given that after 18 rounds, $k_0[0] = r_0[0]$. In other words, the adversary first tests whether $k_0[0] = r_0[0]$ after 18 rounds. If this condition is satisfied, she collects the output $r_6[0]$ after 32 rounds provided $L(r_7)[0] = 0$. Therefore, this constitutes a known-key distinguishing attack which one may view as an attack on a large set of weak keys. Alternatively, the attack scenario may be such that two bits of the internal state after 18 rounds are leaked to the adversary. A similar assumption was made in [4], as a model for certain side-channel attacks. More generally, this scenario is captured by the notion of leakage resilience [5,15], i.e., security when "even a bounded amount of arbitrary (adversarially chosen) information on the internal state (...) is leaked during computation" [5]. Although this assumption leads to trivial attacks (e.g., observe the full internal state of AES at the penultimate rounds), it assists to evaluate security against a wider range of adversaries, and to better understand the resilience of algorithms against "extreme" adversaries.

Since the condition $k_0[0] = r_0[0]$ (after 18 rounds) holds with 0.5 probability, the attacker would need to examine with $2^{17} \cdot 2 = 2^{18}$ randomly generated keys to mount the distinguishing attack with a success probability of 0.9772.

It is easy to see that distinguishers of the same complexity can be built by collecting any other bit of r_6 (after 32 rounds) because F operates in a bitsliced manner. As in Sect. 6.5, when the attacker can observe both the chaining value input and the compression function output, the above distinguishers can be applied onto the compression function as well.

F Key-Recovery Attacks on 32-Round ESSENCE

In Appendix E, we extended the distinguisher on 14-round ESSENCE-256 to 32 rounds by selecting plaintexts based upon the intermediate value of $r_0[j]$ and $k_0[j]$ at round 18. This result may be viewed in terms of a known plaintext key-recovery attack against a vulnerable implementation of the ESSENCE-256 block cipher. Let us say that we are attacking such an implementation of the 32-round ESSENCE-256 block cipher where through some means (side-channel analysis, cache pollution, etc.) we can read bit j of r_0 after 18 rounds. Like in Sect. 6.6, we focus on a subset of 2^{14} plaintexts where $r_0[j] = 0$ (or 1) for all 2^{14} texts after 18 rounds. Applying the same analysis as in Sect. 6.6 to the remaining 14 rounds gives us the value of $k_0[j]$ at round 18. If our vulnerable implementation allows us to read all the bit positions of r_0, then with probability 0.48, we can recover the full key-word k_0 at round 18. Since the key schedule is a bijection (and easily invertible) we can recover the original key with minimal effort. Again, a similar analysis can be applied to the other members of the ESSENCE family of block ciphers.

Differential-Multiple Linear Cryptanalysis

Zhiqiang Liu[1,⋆], Dawu Gu[1], Jing Zhang[1], and Wei Li[2]

[1] Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai 200240, P.R. China
{ilu_zq,dwgu,diandian}@sjtu.edu.cn
[2] School of Computer Science and Technology,
Donghua University, Shanghai 201620, P.R.China
liwei.cs.cn@gmail.com

Abstract. Differential-linear cryptanalysis was introduced by Langford *et al* in 1994. After that, Biham *et al* proposed an enhanced differential-linear cryptanalysis in 2002. In this paper, we present an extension to the enhanced differential-linear cryptanalysis, called differential-multiple linear cryptanalysis, in which a differential characteristic can be concatenated with multiple linear characteristics to derive a differential-multiple linear distinguisher. Furthermore, we introduce a technique about how to find a differential-multiple linear distinguisher based on a differential-linear distinguisher for Feistel and SPN block ciphers. For illustration, this extension is applied to describe a differential-multiple linear distinguisher for 7-round DES, and then the best-known key recovery attack on 9-round DES is presented based on the differential-multiple linear distinguisher. As a matter of fact, our work is a new attempt to concatenate a differential characteristic with multiple linear characteristics to derive a new cryptanalytic tool which may be helpful to analyze a variety of block ciphers including Feistel and SPN schemes.

Keywords: Differential cryptanalysis, Multiple linear cryptanalysis, Differential-Linear cryptanalysis, Differential-Multiple linear cryptanalysis, DES.

1 Introduction

Differential cryptanalysis [5], introduced by Biham *et al* in 1990, is one of the most powerful chosen plaintext attacks against modern cryptosystems. After that, it has been used effectively to analyze many well-known ciphers and some variants of this attack have been presented such as higher order differential attack [11], truncated differential attack [11], impossible differential attack [2], boomerang attack [15], rectangle attack [3], and so on.

Linear cryptanalysis [13], proposed by Matsui in 1993, is another one of the most powerful known plaintext attacks against modern block ciphers. Based on this technique, Kaliski *et al* [10] proposed the idea of generalizing linear

⋆ This work has been supported by Chinese Specialized Research Fund for the Doctoral Program of Higher Education (No.200802480019).

cryptanalysis using multiple linear approximations (the original approach given by Matsui considered only the best linear approximation) in 1994. However, their method imposed a strict constraint as it requires to use approximations deriving the same parity bit of the secret key, which restricted at the same time the number and the quality of the approximations available. As a result, an approach removing the constraint was proposed by Biryukov *et al* in 2004 [7].

Differential-Linear cryptanalysis [12], presented by Langford *et al* in 1994, showed that differential and linear cryptanalysis can be combined together to formulate a more effective attack on modern block ciphers. After that, this technique has been successfully applied to analyze the reduced round DES [12] and reduced round IDEA [8,9]. Furthermore, this attack can be regarded as an example for constructing the distinguisher used in cryptanalysis on modern block ciphers by combining two much shorter and simpler parts. As we know, modern block ciphers are devised to avoid good long differential and linear characteristics in order to resist traditional attacks such as differential and linear cryptanalysis, but usually good short ones still exist. By using the technique given in [12], an adversary can construct two consecutive good short differential or linear characteristics, and then combine them to obtain a long distinguisher so that the adversary can mount an effective attack on a block cipher with more rounds. Except for the combination of a differential characteristic and a linear approximation described in [12], similar combinations were later developed and used in other kinds of cryptanalysis such as cryptanalysis using impossible differentials [2,1], boomerang attack, and rectangle attack, which use combinations of shorter and simpler differential characteristics. In 2002, Biham *et al* [4] proposed an enhanced differential-linear cryptanalysis to generalize the cryptanalytic tool so as to make it more powerful.

In this paper, we present an extension to the enhanced differential-linear cryptanalysis, called differential-multiple linear cryptanalysis, in which a differential characteristic can be concatenated with multiple linear characteristics to derive a differential-multiple linear distinguisher. Furthermore, we introduce a technique about how to find a differential-multiple linear distinguisher based on a differential-linear distinguisher for Feistel and SPN block ciphers. For illustration, we mount a key recovery attack on 9-round DES by using this extension.

The rest of the paper is structured as follows. Section 2 introduces the notations used throughout this paper and describes the method of multiple linear cryptanalysis and enhanced differential-linear cryptanalysis. Section 3 presents our extension to the enhanced differential-linear cryptanalysis, that is, differential-multiple linear cryptanalysis. Section 4 proposes a technique about how to find a differential-multiple linear distinguisher based on a differential-linear distinguisher for Feistel and SPN block ciphers. Section 5 applies our extension to describe a differential-multiple linear distinguisher on 7-round DES, from which a key recovery attack can be mounted on 9-round DES. Finally, Section 6 summarizes the paper.

2 Preliminaries

The following notations are used throughout the paper.

- $\oplus$ denotes bitwise exclusive OR (XOR).
- $\cdot$ denotes bitwise inner product.
- $\|$ denotes concatenation operation.
- $|x|$ denotes the absolute value of a real number x.
- $\circ$ denotes the composition operation.
- 0x denotes the hexadecimal notation.

2.1 Multiple Linear Cryptanalysis

Linear cryptanalysis [13] analyzes a block cipher E by investigating a correlation between the inputs and outputs of E and then obtains a linear approximation (also called linear characteristic and denoted as $(\Gamma_P \rightarrow \Gamma_C)$ of E with following type:

$$\Gamma_P \cdot P \oplus \Gamma_C \cdot C = \Gamma_K \cdot K, \tag{1}$$

where P, C and K denote plaintext, ciphertext, and secret key respectively, Γ_P, Γ_C and Γ_K stand for the mask of plaintext P, ciphertext C, and secret key respectively.

If equation (1) holds with probability $p \neq 1/2$, we call it an effective linear approximation of the block cipher E, and the linear approximation can be used to distinguish E from a random permutation since equation (1) holds with probability 1/2 for a random permutation. Let $\varepsilon = p - 1/2$ be the bias of a linear approximation on E, then the greater $|\varepsilon|$ is, the more effective the corresponding linear approximation will be.

In 1994, Kaliski *et al* [10] proposed the idea of generalizing linear cryptanalysis using multiple linear approximations. However, their method imposed a strict constraint as it requires to use approximations deriving the same parity bit of the secret key, which restricted the number and the quality of the approximations available simultaneously. As a result, an approach removing the constraint was proposed in 2004 [7]. An important consequence of the work in [7] is that the theoretical data complexity of the generalized multiple linear cryptanalysis decreases comparing with the original linear cryptanalysis.

Suppose that one has access to m approximations on E of the following form:

$$\Gamma_P^i \cdot P \oplus \Gamma_C^i \cdot C = \Gamma_K^i \cdot K \quad (1 \leq i \leq m). \tag{2}$$

Let ε_i, $c_i = 2\varepsilon_i$ be the bias and the imbalance of the i-th linear approximation respectively. According to [7], the number of plaintext-ciphertext pairs required in the multiple linear cryptanalysis is inversely proportional to the capacity of the system (i.e., equations (2)) that is defined as:

$$\overline{C}^2 = \sum_{i=1}^{m} c_i^2 = 4 \times \sum_{i=1}^{m} \varepsilon_i^2. \tag{3}$$

Therefore, one can reduce the number of necessary plaintext-ciphertext pairs for a successful key recovery attack by increasing the capacity when using the multiple linear cryptanalytic tool.

2.2 Enhanced Differential-Linear Cryptanalysis

Let E be a block cipher and decompose the cipher into $E = E_1 \circ E_0$, where E_0 represents the first part of the cipher and E_1 represents the last part. Let $\triangle$, ∇ be the input and output differences of a differential characteristic, and Γ_P, Γ_C be the input and output masks of a linear characteristic. The main idea of differential-linear cryptanalysis on block cipher E given in [12] is to concatenate a truncated differential $\triangle \to \nabla$ for E_0 with probability 1 with a linear approximation $\Gamma_P \to \Gamma_C$ with probability $1/2 + \varepsilon$ (or with bias ε) for E_1, where the bits of the intermediate encryption data masked in Γ_P have a zero difference in ∇.

Suppose that a pair of plaintexts P_1 and P_2 satisfying the above condition, then we have $P_1 \oplus P_2 = \triangle$, and $\Gamma_P \cdot E_0(P_1) \oplus \Gamma_P \cdot E_0(P_2) = 0$. Moreover, for the linear approximation $\Gamma_P \to \Gamma_C$ with probability $1/2 + \varepsilon$ for E_1 (that is, the equation $\Gamma_P \cdot P \oplus \Gamma_C \cdot C = \Gamma_K \cdot K$ holds with probability $1/2 + \varepsilon$ for E_1), we have $\Gamma_P \cdot E_0(P_1) \oplus \Gamma_C \cdot C_1 = \Gamma_K \cdot K$ and $\Gamma_P \cdot E_0(P_2) \oplus \Gamma_C \cdot C_2 = \Gamma_K \cdot K$ both hold with probability $1/2 + \varepsilon$, where C_1 and C_2 are the corresponding ciphertexts of P_1 and P_2 respectively (i.e., $C_i = E_1(E_0(P_i))$). Then we immediately obtain the following equation:

$$\Gamma_C \cdot C_1 \oplus \Gamma_C \cdot C_2 = 0, \tag{4}$$

which holds with probability $1/2 + 2\varepsilon^2$ (Assume that $\Gamma_P \cdot E_0(P_1) \oplus \Gamma_C \cdot C_1 = \Gamma_K \cdot K$ and $\Gamma_P \cdot E_0(P_2) \oplus \Gamma_C \cdot C_2 = \Gamma_K \cdot K$ are uncorrelated). As a consequence, the differential-linear distinguisher by checking the parity of $\Gamma_C \cdot C_1 \oplus \Gamma_C \cdot C_2$ can be used to distinguish E from a random permutation since equation (4) holds with probability $1/2$ for a random permutation.

The enhanced differential-linear cryptanalysis [4], proposed in 2002, is to deal with truncated differential $\triangle \to \nabla$ for E_0 with probability p in the above differential-linear distinguisher. In the case that the truncated differential for E_0 is satisfied (with probability p), the above analysis keeps valid, hence the probability that equation (4) holds is $p(1/2 + 2\varepsilon^2)$. While in the case that the truncated differential for E_0 is not satisfied (with probability $1 - p$), we assume that a random parity of $\Gamma_C \cdot C_1 \oplus \Gamma_C \cdot C_2$ happens, therefore, the probability that equation (4) holds in this case is $(1 - p)/2$. Then the probability that a pair of plaintexts P_1 and P_2 with input difference $\triangle$ satisfy equation (4) is $p(1/2 + 2\varepsilon^2) + (1 - p)/2 = 1/2 + 2p\varepsilon^2$. Moreover, it was also indicated in [4] that the technique can still work if $\Gamma_P \cdot \nabla$ remains constant. If $\Gamma_P \cdot \nabla = 0$, we can get the same analysis result as above. Similarly, for the case $\Gamma_P \cdot \nabla = 1$, the probability that a pair of plaintexts P_1 and P_2 with input difference $\triangle$ meet $\Gamma_C \cdot C_1 \oplus \Gamma_C \cdot C_2 = 1$ is $1/2 + 2p\varepsilon^2$.

3 Differential-Multiple Linear Cryptanalysis

We now present an extension to the enhanced differential-linear cryptanalysis, called differential-multiple linear cryptanalysis, in which we concatenate a differential characteristic with multiple linear characteristics to derive a differential-multiple linear distinguisher.

Let E be a block cipher and decompose the cipher into $E = E_1 \circ E_0$, where E_0 represents the first part of the cipher and E_1 represents the last part. We will use a truncated differential characteristic $\triangle \rightarrow \nabla$ for E_0 with probability p, as well as multiple linear characteristics $\Gamma_P^i \rightarrow \Gamma_C^i$ $(1 \leq i \leq m)$ with probability $1/2 + \varepsilon_i$ for E_1 which can be expressed as $\Gamma_P^i \cdot P \oplus \Gamma_C^i \cdot C = \Gamma_K^i \cdot K$.

We want to cover the following condition:

$$\nabla \cdot \Gamma_P^i = 0 \quad (1 \leq i \leq m). \tag{5}$$

Then we claim that the following equations

$$\Gamma_C^i \cdot C_1 \oplus \Gamma_C^i \cdot C_2 = 0 \quad (1 \leq i \leq m) \tag{6}$$

hold with probability $1/2 + 2p\varepsilon_i^2$ respectively for any pair of plaintexts P_1 and P_2 with input difference $\triangle$, where C_1 and C_2 are the corresponding ciphertexts of P_1 and P_2 respectively (i.e., $C_i = E_1(E_0(P_i))$).

We will examine the above result in two cases. On the one hand, consider the case that a pair of plaintext P_1 and P_2 satisfying the truncated differential characteristic $\triangle \rightarrow \nabla$ for E_0 (with probability p). For the multiple linear characteristics $\Gamma_P^i \rightarrow \Gamma_C^i$ $(1 \leq i \leq m)$ with probability $1/2 + \varepsilon_i$ for E_1, we have that $\Gamma_P^i \cdot E_0(P_1) \oplus \Gamma_C^i \cdot C_1 = \Gamma_K^i \cdot K$ and $\Gamma_P^i \cdot E_0(P_2) \oplus \Gamma_C^i \cdot C_2 = \Gamma_K^i \cdot K$ hold with probability $1/2 + \varepsilon_i$ for E_1, then we can immediately obtain that

$$\Gamma_P^i \cdot E_0(P_1) \oplus \Gamma_P^i \cdot E_0(P_2) \oplus \Gamma_C^i \cdot C_1 \oplus \Gamma_C^i \cdot C_2 = 0 \tag{7}$$

holds with probability $1/2 + 2\varepsilon_i^2$ for E_1. Combing with the condition that P_1 and P_2 satisfy the truncated differential characteristic $\triangle \rightarrow \nabla$ for E_0 and the equations (5), we have

$$\begin{aligned} & \Gamma_P^i \cdot E_0(P_1) \oplus \Gamma_P^i \cdot E_0(P_2) \oplus \Gamma_C^i \cdot C_1 \oplus \Gamma_C^i \cdot C_2 \\ = \; & \Gamma_P^i \cdot (E_0(P_1) \oplus E_0(P_2)) \oplus \Gamma_C^i \cdot C_1 \oplus \Gamma_C^i \cdot C_2 \\ = \; & \Gamma_P^i \cdot \nabla \oplus \Gamma_C^i \cdot C_1 \oplus \Gamma_C^i \cdot C_2 \\ = \; & \Gamma_C^i \cdot C_1 \oplus \Gamma_C^i \cdot C_2, \end{aligned} \tag{8}$$

thus in this case, the linear equations (6) hold with probability $p(1/2 + 2\varepsilon_i^2)$ for E respectively. On the other hand, consider the case that a pair of plaintext P_1 and P_2 with input difference $\triangle$ but not satisfying the truncated differential characteristic $\triangle \rightarrow \nabla$ for E_0 (with probability $1 - p$). In this case, we assume that a random parity of $\Gamma_C^i \cdot C_1 \oplus \Gamma_C^i \cdot C_2$ happens, therefore, the probability that each equation in (6) holds is $(1 - p)/2$. Then the probability that a pair of plaintexts P_1 and P_2 with input difference $\triangle$ satisfy the i-th equation in (6) for E is $p(1/2 + 2\varepsilon_i^2) + (1 - p)/2 = 1/2 + 2p\varepsilon_i^2$.

The above result allows us to get m linear approximations $\Gamma_C^i \cdot C_1 \oplus \Gamma_C^i \cdot C_2 = 0$ $(1 \leq i \leq m)$ with bias $2p\varepsilon_i^2$ for the block cipher E. Then by using the technique proposed in [7], we can construct a differential-multiple linear distinguisher for E by checking the parities of $\Gamma_C^i \cdot C_1 \oplus \Gamma_C^i \cdot C_2$ $(1 \leq i \leq m)$. Moreover, a key recovery attack based on the differential-multiple linear distinguisher can be mounted on $E' = E_2 \circ E$ (where E_2 represents the last round of the cipher E') using standard technique such as guessing part of the last round subkey, if for every i $(1 \leq i \leq m)$, any plaintext P and its corresponding ciphertext C, $\Gamma_C^i \cdot E(P)$ is related to the same subset of bits in the last round subkey, in other words, $\Gamma_C^i \cdot E(P)$ can be obtained by performing a partial decryption of the ciphertext C using the same subset of bits in the last round subkey for any i $(1 \leq i \leq m)$. Thus with the technique introduced in [7], the key recovery attack for the cipher E' has the data complexity proportional to $1/\overline{C}^2$, where $\overline{C}^2 = 16p^2 \sum_{i=1}^{m} \varepsilon_i^4$, and the exact number of chosen plaintext pairs is a function of the desired success rate and the number of guessed subkey bits, which can be approximated by using the measure given in [14].

4 Deriving a Differential-Multiple Linear Distinguisher from an Existing Differential-Linear Distinguisher

Let E be an n-round DES-like balanced Feistel block cipher and decompose the cipher into $E = E_1 \circ E_0$, where E_0 represents the first s rounds of the cipher, E_1 represents the last t rounds and $n = s + t$. Let the bias of a linear approximation on the i-th round F-function of E be defined as:

$$(\Gamma X_i, \Gamma Y_i) = \delta_i = \Pr\{\Gamma X_i \cdot X_i \oplus \Gamma Y_i \cdot Y_i = 0\} - 1/2, \tag{9}$$

where X_i and Y_i denote the input and output of the i-th round F-function of E, and ΓX_i, ΓY_i represent their masks respectively. According to the piling up lemma in [13], the total bias ε_{tot} on n rounds is given by:

$$\varepsilon_{tot} = [\delta_1, \delta_2, \ldots, \delta_n] = 2^{n-1} \prod_{i=1}^{n} \delta_i, \tag{10}$$

and the absolute value of the bias for the best linear approximation on the whole cipher E is then defined as:

$$B_n = \max_{\substack{\Gamma Y_i = \Gamma X_{i-1} \oplus \Gamma Y_{i-2} \\ (3 \leq i \leq n)}} |[(\Gamma X_1, \Gamma Y_1), (\Gamma X_2, \Gamma Y_2), \ldots, (\Gamma X_n, \Gamma Y_n)]|. \tag{11}$$

Suppose we have already found a differential-linear distinguisher (denoted as D) for E: D consists of a truncated differential characteristic $\triangle \rightarrow \nabla$ for E_0 with probability p, and a linear characteristic $\Gamma_P \rightarrow \Gamma_C$ with bias ε for E_1 such that $\nabla \cdot \Gamma_P = 0$. Then we can derive a differential-multiple linear distinguisher from D generally as below:

• Find all possible quartets $Q^i = (\mathit{\Gamma} X_1^i,\ \mathit{\Gamma} Y_1^i,\ \mathit{\Gamma} X_2^i,\ \mathit{\Gamma} Y_2^i)$ each of which satisfies $\nabla \cdot (\mathit{\Gamma} Y_1^i \| (\mathit{\Gamma} X_1^i \oplus \mathit{\Gamma} Y_2^i)) = 0$;

• For all above quartets Q^i, search for m best linear characteristics $\mathit{\Gamma}_P^i \rightarrow \mathit{\Gamma}_C^i$ $(1 \leq i \leq m)$ for E_1, each of which is derived from the concatenation of t one-round linear approximations $\mathit{\Gamma} X_1^i \rightarrow \mathit{\Gamma} Y_1^i,\ \mathit{\Gamma} X_2^i \rightarrow \mathit{\Gamma} Y_2^i,\ \ldots,\ \mathit{\Gamma} X_t^i \rightarrow \mathit{\Gamma} Y_t^i$ satisfying $\mathit{\Gamma} Y_j^i = \mathit{\Gamma} X_{j-1}^i \oplus \mathit{\Gamma} Y_{j-2}^i$ $(3 \leq j \leq t)$, thus we have $\mathit{\Gamma}_P^i = \mathit{\Gamma} Y_1^i \| (\mathit{\Gamma} X_1^i \oplus \mathit{\Gamma} Y_2^i)$ and $\mathit{\Gamma}_C^i = \mathit{\Gamma} Y_t^i \| (\mathit{\Gamma} X_t^i \oplus \mathit{\Gamma} Y_{t-1}^i)$;

• Check each of the m linear characteristics and find the ones of which each $\mathit{\Gamma}_C^i$ corresponds to the same subset of bits in the following round subkey;

• Then we can construct a differential-multiple linear distinguisher by combining the truncated differential characteristic $\triangle \rightarrow \nabla$ for E_0 and the list of linear characteristics $\mathit{\Gamma}_P^i \rightarrow \mathit{\Gamma}_C^i$ meeting the above condition.

The algorithm which searches for such kind of linear characteristics is a simple modification of the branch-and-bound algorithm in [7] by setting the condition that for each input mask $\mathit{\Gamma}_P^i = \mathit{\Gamma} Y_1^i \| (\mathit{\Gamma} X_1^i \oplus \mathit{\Gamma} Y_2^i)$, $\nabla \cdot \mathit{\Gamma}_P^i = 0$ should be met, and for each output mask $\mathit{\Gamma}_C^i = \mathit{\Gamma} Y_t^i \| (\mathit{\Gamma} X_t^i \oplus \mathit{\Gamma} Y_{t-1}^i)$, the same subset of bits in the following round subkey will be influenced.

As for McGuffin-like unbalanced Feistel block ciphers and SPN block ciphers, similar technique can be applied to derive a differential-multiple linear distinguisher from an existing differential-linear distinguisher.

5 Differential-Multiple Linear Attack on 9-Round DES

We now present our differential-multiple linear attack on 9-round DES. In our attack, a 7-round differential-multiple linear distinguisher for the rounds from the second round to the eighth round of DES will be constructed firstly, then 12 bits of the subkey of the first round and 6 bits of the subkey of the ninth round can be retrieved from the attack.

The 7-round differential-multiple linear distinguisher is derived from the 7-round differential-linear characteristic given in [4] by using the means introduced in Section 4. The 4-round truncated differential characteristic $\triangle \rightarrow \nabla$ (from the second round to the fifth round of DES) used in our attack is expressed in Fig. 1(The differential characteristic is inherited from [4], and (the right half of ∇)$\|$(the left half of ∇) is denoted by ∇'). Note that the left half of difference $\triangle$ (i.e., 0x 00 00 02 02) influences 2 active S-boxes in the first round of DES, that is, the S-boxes $S6$ and $S8$. Moreover, the six 3-round linear characteristics (from the sixth round to the eighth round of DES) $\mathit{\Gamma}_P^i \rightarrow \mathit{\Gamma}_C^i$ $(1 \leq i \leq 6)$ used in our attack are described in Fig. 2, Fig. 3, Fig. 4, Fig. 5, Fig. 6 and Fig. 7 respectively (The six linear characteristics are obtained by applying the technique given in Section 4), where $\nabla' \cdot \mathit{\Gamma}_P^i = 0$ and $\mathit{\Gamma}_C^i$ corresponds to the same subset of bits in the subkey of the ninth round for any $i (1 \leq i \leq 6)$. As a matter of fact, the right half of each $\mathit{\Gamma}_C^i$ influences the same active S-box in the ninth round of DES, that is, the S-box $S1$. Thus we have that the following equations

$$\mathit{\Gamma}_C^i \cdot C_1 \oplus \mathit{\Gamma}_C^i \cdot C_2 = 0 \ (1 \leq i \leq 6) \qquad (12)$$

hold with probability $1/2 + 2 \times 12/64 \times (25/128)^2 \approx 1/2 + 2^{-6.13}$, $1/2 + 2 \times 12/64 \times (25/512)^2 \approx 1/2 + 2^{-10.13}$, $1/2 + 2 \times 12/64 \times (9/128)^2 \approx 1/2 + 2^{-9.07}$, $1/2 + 2 \times 12/64 \times (25/512)^2 \approx 1/2 + 2^{-10.13}$, $1/2 + 2 \times 12/64 \times (1/32)^2 \approx 1/2 + 2^{-11.41}$ and $1/2 + 2 \times 12/64 \times (49/512)^2 \approx 1/2 + 2^{-8.19}$ respectively, where C_1 and C_2 are the corresponding ciphertexts of plaintexts P_1 and P_2 ($P_1 \oplus P_2 = \triangle$) under the 7-round DES (from the second round to the eighth round of DES) respectively. Then we get a 7-round differential-multiple linear distinguisher for the rounds from the second round to the eighth round of DES by checking the parities of $\Gamma_C^i \cdot C_1 \oplus \Gamma_C^i \cdot C_2$ $(1 \le i \le 6)$.

Following the approach given in Section 3, a key recovery attack can be mounted on the 9-round DES by applying the above 7-round differential-multiple linear distinguisher. Since there are 2 active S-boxes in the round before the differential-multiple linear distinguisher (i.e., the first round of DES) and 1 active S-box in the round after the distinguisher (i.e., the ninth round of DES), we need to perform partial encryptions of all the plaintexts involved in the attack for each guess of subkey bits in the first round that enters the two active S-boxes $S6$ and $S8$, and find all the plaintext pairs which lead to the difference $\triangle$ at the entrance to the second round. Then for each of these pairs and each guess of the subkey bits in the ninth round that enters the active S-box $S1$, we implement the partial decryptions of the ciphertexts corresponding to the pair and check whether the equations (12) hold or not. Thus based on the *Attack Algorithm MK 2* in [7], we can determine the probability that the guessed subkey bits are correct by exploiting the linear equations (12).

For the linear system consisting of the linear equations (12), the capacity of the system is about $4 \times ((2^{-6.13})^2 + (2^{-10.13})^2 + (2^{-9.07})^2 + (2^{-10.13})^2 + (2^{-11.41})^2 + (2^{-8.19})^2) \approx 2^{-10.14}$. Thus the necessary number of chosen plaintext pairs required in our attack is proportional to $2^{10.14}$. Since 18 subkey bits need to be retrieved in our attack, according to the Table 2 given in [14], we need to prepare $2^{13.1}$ (that is, $2^{2.96} \times 2^{10.14}$) chosen plaintext pairs in the attack so as to achieve a high success probability of 89% approximately. Following gives the detailed description of our differential-multiple linear attack on the 9-round DES.

Before the distillation and analysis phase, we will use the structures of chosen plaintexts proposed in [4] to select plaintexts needed in our attack as below:

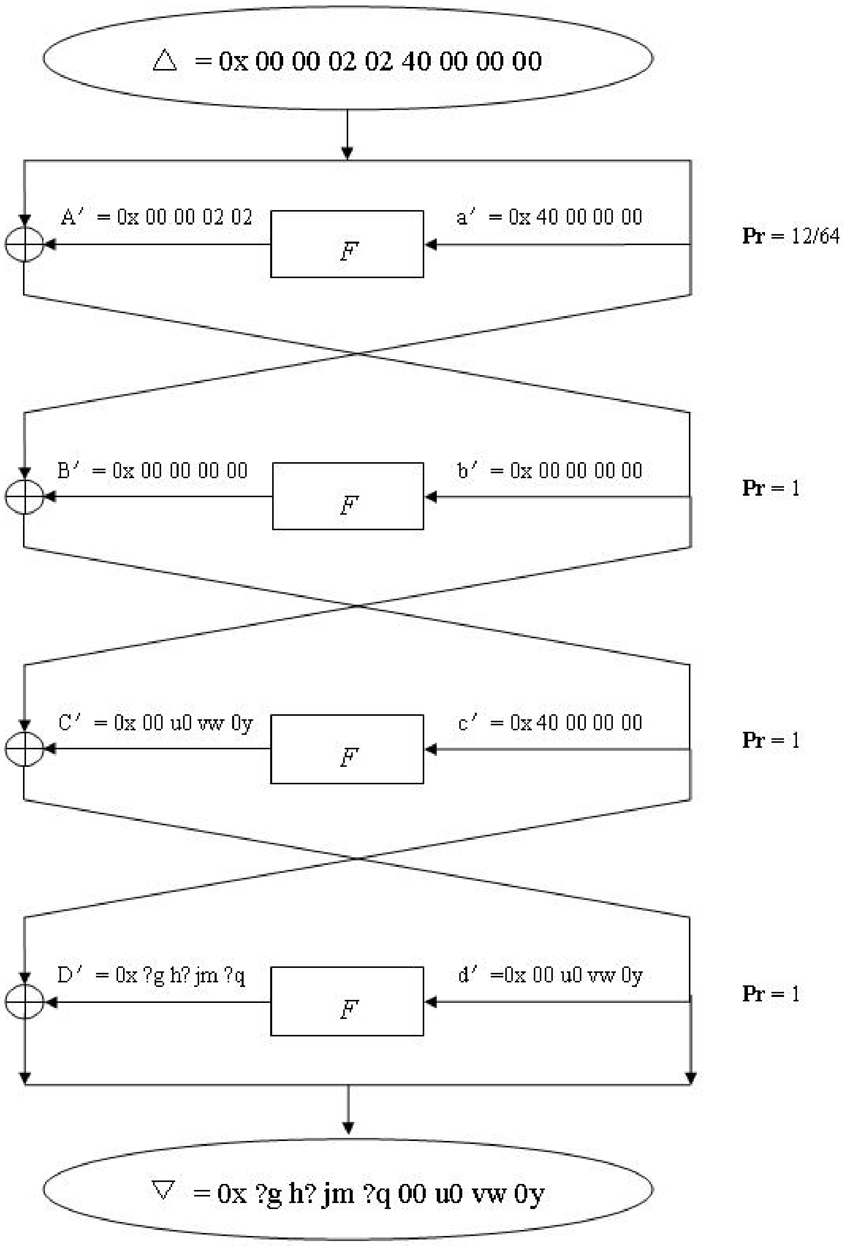
1. Select $2^{14.1}$ plaintexts, consisting of $2^{5.1}$ structures, and each structure is generated by selecting:

• Any plaintext P_0,

• The plaintexts $P_1, \dots, P_{255}$ which differ from P_0 by all the 255 possible (non-empty) subsets of the eight bits masked by 0x 18 22 28 28 00 00 00 00 (the eight bits correspond to the output bits of $S6$ and $S8$),

• The plaintexts $P_{256}, \dots, P_{511}$ selected as $P_i = P_{i-256} \oplus$ 0x 40 00 00 00 00 00 02 02.

As a matter of fact, each structure can result in 2^8 expected plaintext pairs which have difference $\triangle$ at the entrance to the second round of DES.



where u, v∈ {0, 8}, w, y∈ {0, 2}, g∈ {0, 1, 4, 5, 8, 9, 0xC, 0xD}, h∈ {0, 2, 4, 6}, j∈ {0, ⋯, 7}, m∈ {0, 1, 8, 9}, q∈ {0, 4, 8, 0xC}, and ? means any possible value.

Fig. 1. The 4-round Differential Characteristic ($\triangle \rightarrow \nabla$) with $p = 12/64$

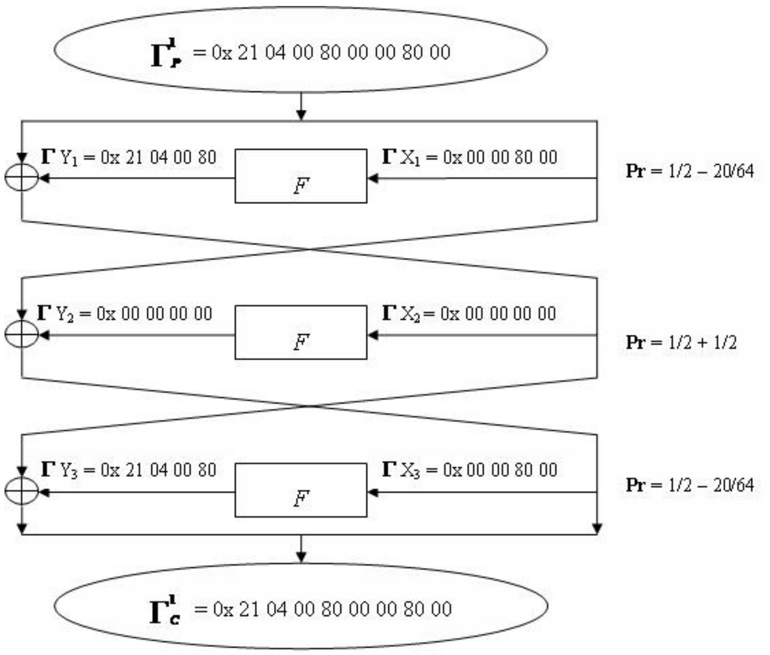


Fig. 2. The 3-round Linear Characteristic ($\Gamma_P^1 \rightarrow \Gamma_C^1$) with $p = 1/2 + 25/128$

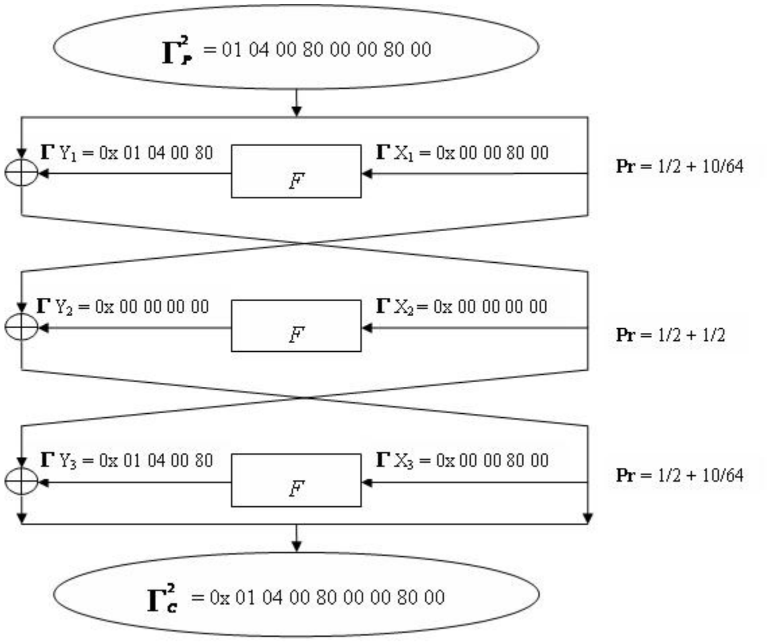
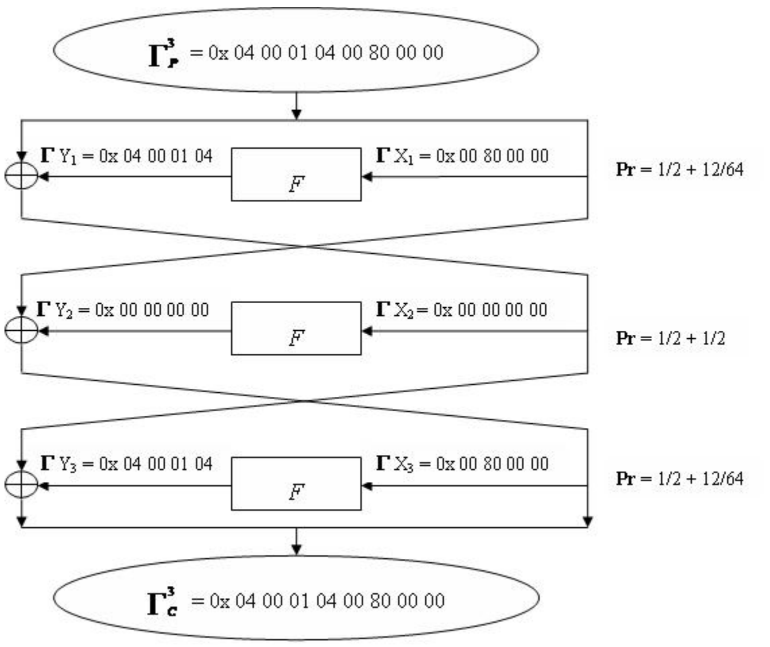
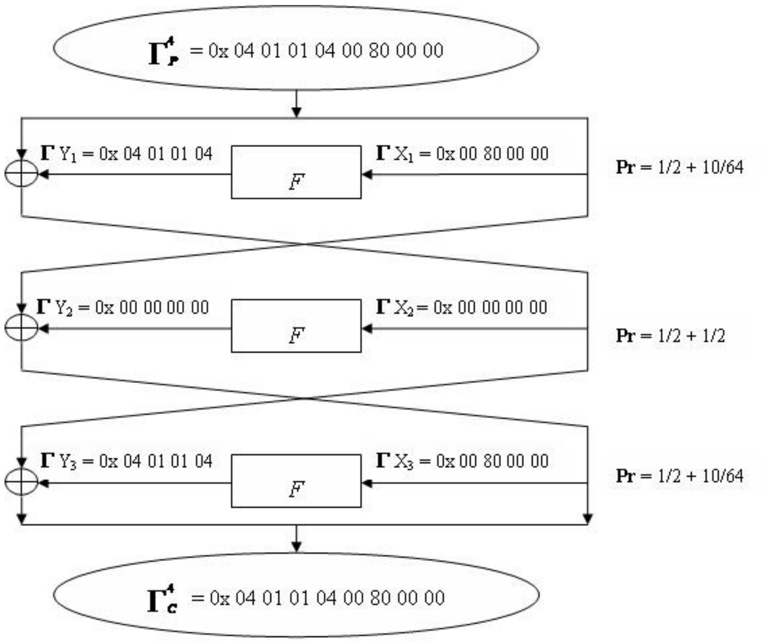


Fig. 3. The 3-round Linear Characteristic ($\Gamma_P^2 \rightarrow \Gamma_C^2$) with $p = 1/2 + 25/512$



Γ_P^3 = 0x 04 00 01 04 00 80 00 00
ΓY_1 = 0x 04 00 01 04
ΓX_1 = 0x 00 80 00 00
F
Pr = 1/2 + 12/64
ΓY_2 = 0x 00 00 00 00
ΓX_2 = 0x 00 00 00 00
F
Pr = 1/2 + 1/2
ΓY_3 = 0x 04 00 01 04
ΓX_3 = 0x 00 80 00 00
F
Pr = 1/2 + 12/64
Γ_C^3 = 0x 04 00 01 04 00 80 00 00

Fig. 4. The 3-round Linear Characteristic ($\Gamma_P^3 \rightarrow \Gamma_C^3$) with $p = 1/2 + 9/128$



Γ_P^4 = 0x 04 01 01 04 00 80 00 00
ΓY_1 = 0x 04 01 01 04
ΓX_1 = 0x 00 80 00 00
F
Pr = 1/2 + 10/64
ΓY_2 = 0x 00 00 00 00
ΓX_2 = 0x 00 00 00 00
F
Pr = 1/2 + 1/2
ΓY_3 = 0x 04 01 01 04
ΓX_3 = 0x 00 80 00 00
F
Pr = 1/2 + 10/64
Γ_C^4 = 0x 04 01 01 04 00 80 00 00

Fig. 5. The 3-round Linear Characteristic ($\Gamma_P^4 \rightarrow \Gamma_C^4$) with $p = 1/2 + 25/512$

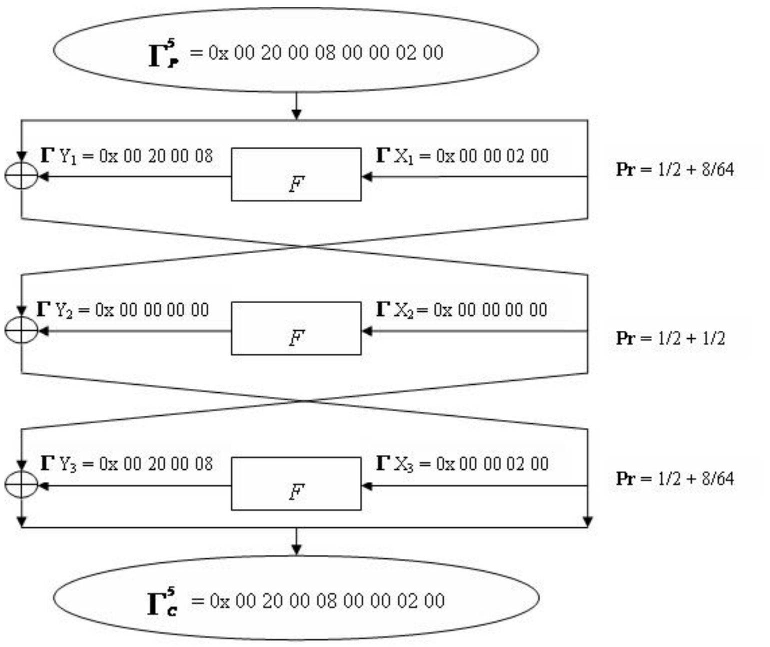


Fig. 6. The 3-round Linear Characteristic ($\mathit{\Gamma}_P^5 \rightarrow \mathit{\Gamma}_C^5$) with $p = 1/2 + 1/32$

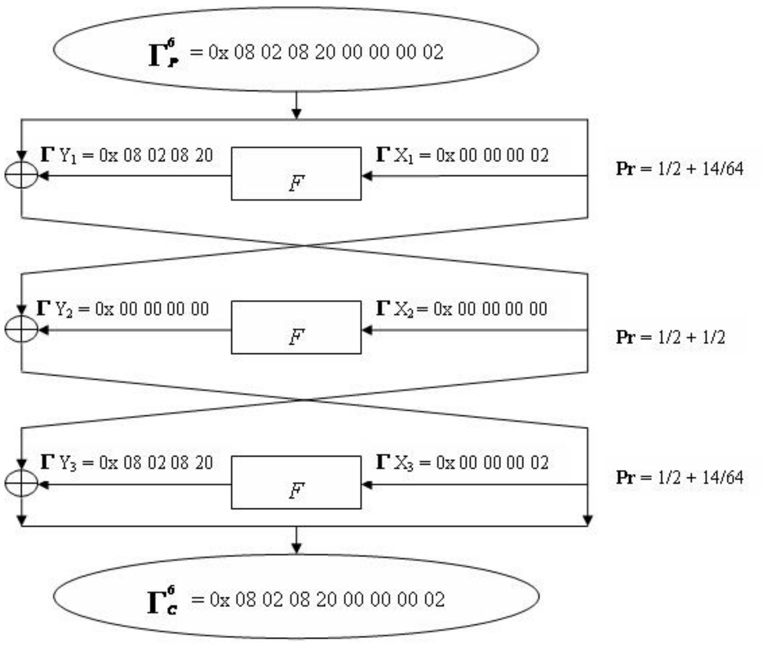


Fig. 7. The 3-round Linear Characteristic ($\mathit{\Gamma}_P^6 \rightarrow \mathit{\Gamma}_C^6$) with $p = 1/2 + 49/512$

2. Request the ciphertexts of these plaintext structures (encrypted under the unknown key K).

Distillation and analysis phase

1. Initialize a vector $\mathbf{T} = (T_{i,\eta})_{1\leq i\leq 6,\ 0\leq\eta\leq 2^6-1}$ composed of 6×2^6 counters, where $T_{i,\eta}$ corresponds to the i-th linear equation in (12) and the subkey candidate η which represents the possible value of the 6 bits of K_9 entering the active S-box $S1$ in the ninth round of DES.

2. For each guessed value of the 12 bits of K_1 entering the two active S-boxes $S6$ and $S8$ in the first round of DES, do the following:

(I). Partially encrypt each plaintext and find the pairs satisfying the difference $\triangle$ at the entrance to the second round of DES.

(II). For each chosen plaintext pair (P, P') obtained above, let C and C' denote the corresponding ciphertexts of P and P' under the 9-round DES respectively. Then for each possible value of η, implement the partial decryptions of C and C' respectively and compute the parities in terms of the linear approximations given by the equations (12). If the parity for the i-th linear approximation is 0, increment the relevant counter $T_{i,\eta}$ by 1, and decrement by 1 otherwise.

(III). Let $\hat{\mathbf{c}} = (\hat{c}_{i,\eta})_{1\leq i\leq 6,\ 0\leq\eta\leq 2^6-1}$ denote a vector consisting of 6×2^6 elements, where $\hat{c}_{i,\eta}$ represents the estimated imbalance for the i-th linear equation in (12) and the subkey candidate η which represents the possible value of the 6 bits of K_9 entering the active S-box $S1$ in the ninth round of DES. Then we have $\hat{c}_{i,\eta} = T_{i,\eta}/2^{13.1}$.

(IV). Compute $\|\hat{\mathbf{c}}\|^2 = \sum\limits_{i=1}^{6}\sum\limits_{\eta=0}^{2^6-1}\hat{c}_{i,\eta}^2$, and for each subkey candidate η, calculate $\|\hat{\mathbf{c}}_\eta\|^2 = \sum\limits_{i=1}^{6}\hat{c}_{i,\eta}^2$.

(V). For a given subkey candidate η, a vector $\mathbf{c}_\eta$ of theoretical imbalances with 6×2^6 elements could be constructed as follows:

$$\mathbf{c}_\eta = (0,\ldots,0,c_1,\ldots,c_6,0,\ldots,0), \tag{13}$$

where c_i $(1\leq i\leq 6)$ corresponds to the imbalance of the i-th linear equation in (12), and the location of the subvector $(c_1,\ldots,c_6)$ depends on the value of η.

(VI). For each possible subkey candidate η, the Euclidean distance between the vector of estimated imbalances and the vector of theoretical imbalances is measured by the following equation:

$$\begin{aligned}\|\hat{\mathbf{c}}-\mathbf{c}_\eta\|^2 &= \sum_{i=1}^{6}(\hat{c}_{i,\eta}-c_i)^2 + \sum_{\eta'\neq\eta}\sum_{i=1}^{6}\hat{c}_{i,\eta'}^2 \\ &= \sum_{i=1}^{6}(\hat{c}_{i,\eta}-c_i)^2 + (\|\hat{\mathbf{c}}\|^2 - \|\hat{\mathbf{c}}_\eta\|^2).\end{aligned} \tag{14}$$

(VII). Store the subkey candidate η along with the guessed bits of K_1 and the corresponding Euclidean distance $\|\hat{\mathbf{c}}-\mathbf{c}_\eta\|^2$ if the distance is minimal under the guessed bits of K_1.

3. For all possible values of the guessed bits of K_1, compare the stored Euclidean distances and take the subkey candidate η together with the guessed bits of K_1 as the correct key information if the corresponding Euclidean distance is minimal.

The data complexity of the attack is $2^{14.1}$ chosen plaintexts. The time complexity of the attack is dominated mainly by the partial decryptions of all ciphertext pairs in the step 2(II) of the distillation and analysis phase. Thus the time complexity of the attack is about $2^{12} \times 2^{14.1} \times 2^{6}/72 \approx 2^{25.93}$ 9-round DES encryptions. The complexities of our attack together with the previously known attacks on 9-round DES are summarized in Table 1.

Table 1. Summary of Attacks on 9-round DES

Type of Attack	Success Rate	Complexity	
		Data	Time
Differential [6]	99.97%	2^{24} CP	2^{32} Enc
Linear [13]	89%	$2^{21.03}$ KP	$2^{32.86}$ Enc
Differential-Linear [4]	88.8%	$2^{15.8}$ CP	$2^{29.2}$ Enc
Differential-Multiple Linear (this paper)	89%	$2^{14.1}$ CP	$2^{25.93}$ Enc

KP - Known plaintexts, CP - Chosen plaintexts.
Enc - Encryptions.

6 Conclusion

In this paper, we have presented an extension to the enhanced differential-linear cryptanalysis, called differential-multiple linear cryptanalysis, which allows using multiple linear characteristics to mount a multiple linear attack. Moreover, we have introduced an approach about how to construct a differential-multiple linear distinguisher from an existing differential-linear distinguisher for Feistel and SPN block ciphers. For the purpose of illustration, our extension has been applied to attack 9-round DES, resulting in the best-known key recovery attack on 9-round DES. As a matter of fact, our work is a new attempt to concatenate a differential characteristic with multiple linear characteristics to derive a new cryptanalytic tool which may be helpful to analyze a variety of block ciphers including Feistel and SPN schemes, and we hope that further research can be done on the differential-multiple linear cryptanalysis to achieve better results in the future.

Acknowledgements

We are very grateful to the anonymous referees for their comments and editorial suggestions.

References

1. 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)
2. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. Journal of Cryptology 18(4), 291–311 (2005)
3. Biham, E., Dunkelman, O., Keller, N.: The rectangle attack - rectangling the Serpent. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 340–357. Springer, Heidelberg (2001)
4. Biham, E., Dunkelman, O., Keller, N.: Enhancing differential-linear cryptanalysis. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 254–266. Springer, Heidelberg (2002)
5. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991)
6. Biham, E., Shamir, A.: Differential cryptanalysis of the Data Encryption Standard. Springer, Heidelberg (1993)
7. Biryukov, A., De Cannière, C., Quisquater, M.: On multiple linear approximations. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 1–22. Springer, Heidelberg (2004)
8. 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)
9. Hawkes, P.: Differential-linear weak key classes of IDEA. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 112–126. Springer, Heidelberg (1998)
10. Kaliski, B.S., Robshaw, M.J.B.: Linear cryptanalysis using multiple approximations. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 26–39. Springer, Heidelberg (1994)
11. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
12. Langford, S.K., Hellman, M.E.: Differential-linear cryptanalysis. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 17–25. Springer, Heidelberg (1994)
13. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
14. Selçuk, A.A.: On probability of success in linear and differential cryptanalysis. Journal of Cryptology 21(1), 131–147 (2008)
15. Wagner, D.: The boomerang attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)

Differential Attack on Five Rounds of the SC2000 Block Cipher[⋆]

Jiqiang Lu

Department of Mathematics and Computer Science,
Eindhoven University of Technology,
5600 MB Eindhoven, The Netherlands
lvjiqiang@hotmail.com

Abstract. SC2000 is a 128-bit block cipher with a user key of 128, 192 or 256 bits, which employs a total of 6.5 rounds if a 128-bit user key is used. It is a CRYPTREC recommended e-government cipher. In this paper we describe one 4.75-round differential characteristic with probability 2^{-126} of SC2000 and thirty 4.75-round differential characteristics with probability 2^{-127}. Finally, we exploit these 4.75-round differentials to conduct a differential cryptanalysis attack on a 5-round reduced version of SC2000 when used with a 128-bit key. The attack suggests for the first time that the safety margin of SC2000 with a 128-bit key decreases below one and a half rounds.

Keywords: Block cipher, SC2000, Differential cryptanalysis.

1 Introduction

The SC2000 block cipher [1,2] has a 128-bit block size and a user key of 128, 192 or 256 bits, which employs a total of 6.5 rounds for a 128-bit user key, and a total of 7.5 rounds for a 192 or 256-bit key. It was designed to "have high performance on a wide range of platforms from the low-end processors used in smart cards and mobilephones to the high-end ones that will be available in the near future by suitably implementing it in each platform, and also to have high security" [3]. In 2002, SC2000 became a CRYPTREC recommended e-government cipher [4], after a thorough analysis of its security and performance. In this paper we consider the version of SC2000 that uses 128 key bits.

The SC2000 designers [1,2] first analysed the security of SC2000 against differential cryptanalysis [5] as well as certain other cryptanalytic methods, such as linear cryptanalysis [6]. In 2001, Raddum and Knudsen [7] presented a differential attack on 4.5-round SC2000, which is based on two 3.5-round differential characteristics with probabilities 2^{-106} and 2^{-107}, respectively. In 2002, by exploiting a few short differentials with large probabilities, Biham et al. [8] presented boomerang [9] and rectangle [10] attacks on 3.5-round SC2000, following the work described in [11]. In the same year, Yanami et al. [12] described a

⋆ The work was done when the author was with Royal Holloway, University of London (UK).

Table 1. Cryptanalytic results on SC2000

Attack Type	*Rounds*	*Data*	*Time*	*Source*
Boomerang attack	3.5	2^{67} ACPC	2^{67} Memory accesses	[8]
Rectangle attack	3.5	$2^{84.6}$ CP	$2^{84.6}$ Memory accesses	[8]
Linear attack	4.5	$2^{104.3}$ KP	$2^{83.3}$ Memory accesses	[12]
	4.5	$2^{115.2}$ KP	$2^{42.3}$ Memory accesses	[12]
Differential attack	4.5	2^{111} CP	2^{111} Encryptions	[7]
	4.5	2^{104} CP	2^{20} Memory accesses	[12]
	5	2^{127} CP	2^{132} Memory accesses	This paper

2-round iterative differential characteristic with probability 2^{-58}, and obtained a 3.5-round differential characteristic with probability 2^{-101} by concatenating the 2-round differential twice and then removing the first half round; finally they presented a differential attack on 4.5-round SC2000 with a time complexity smaller than that of the attack of Raddum and Knudsen. Yanami et al. also presented linear attacks on 4.5-round SC2000. These are the best previously published cryptanalytic results on SC2000 in terms of the numbers of attacked rounds.

In this paper we describe one 4.75-round differential characteristic with probability 2^{-126} and thirty 4.75-round differential characteristics with probability 2^{-127}, building on the two-round iterative differential characteristic with probability 2^{-58} of Yanami et al. Finally, using these 4.75-round differential characteristics we present a differential cryptanalysis attack on 5-round SC2000, faster than an exhaustive key search. The attack is the first published attack on 5-round SC2000. Table 1 sumarises both the previous and our new cryptanalytic results on SC2000, where ACPC, CP and KP respectively refer to the required numbers of adaptive chosen plaintexts and ciphertexts, chosen plaintexts, and known plaintexts.

The remainder of this paper is organised as follows. In the next section, we give the notation, and describe differential cryptanalysis and the SC2000 block cipher. In Section 3, we give the 4.75-round differential characteristics. In Section 4, we present our differential attack on 5-round SC2000. Section 5 concludes the paper.

2 Preliminaries

In this section we give the notation used throughout this paper, and then briefly describe differential cryptanalysis and the SC2000 cipher.

2.1 Notation

In all descriptions we assume that the bits of a n-bit value are numbered from 0 to $n-1$ from left to right, the most significant bit is the 0-th bit, a number without a prefix expresses a decimal number, and a number with prefix $0x$ expresses a hexadecimal number. We use the following notation.

$\oplus$	bitwise logical exclusive OR (XOR) operation
$\wedge$	bitwise logical AND operation
$\circ$	functional composition. When composing functions X and Y, X $\circ$ Y denotes the function obtained by first applying X and then applying Y
$\bowtie$	exchange of the left and right halves of a bit string
$\overline{X}$	bitwise logical complement of a bit string X

2.2 Differential Cryptanalysis

Differential cryptanalysis [5] takes advantage of how a specific difference in a pair of inputs of a cipher can affect a difference in the pair of outputs of the cipher, where the pair of outputs are obtained by encrypting the pair of inputs using the same key. The notion of difference can be defined in several ways; the most widely discussed is with respect to the XOR operation. The difference between the inputs is called the input difference, and the difference between the outputs of a function is called the output difference. The combination of the input difference and the output difference is called a differential. The probability of a differential is defined as follows.

Definition 1. *If α and β are n-bit blocks, then the probability of the differential (α, β) for a block cipher* $\mathbf{E}$*, written $\varDelta\alpha \to \varDelta\beta$, is defined to be*

$$\mathrm{Pr}_{\mathbf{E}}(\varDelta\alpha \to \varDelta\beta) = \Pr_{P \in \{0,1\}^n}(\mathbf{E}(P) \oplus \mathbf{E}(P \oplus \alpha) = \beta).$$

For a random function, the expected probability of a differential for any pair (α, β) is 2^{-n}. Therefore, if $\mathrm{Pr}_{\mathbf{E}}(\varDelta\alpha \to \varDelta\beta)$ is larger than 2^{-n}, we can use the differential to distinguish $\mathbf{E}$ from a random function, given a sufficient number of chosen plaintext pairs.

2.3 The SC2000 Block Cipher

SC2000 takes as input a 128-bit plaintext. For simplicity, we describe the plaintext P as four 32-bit words (d, c, b, a). The following three elementary functions I, B and R are used to define the SC2000 round function, as shown in Fig. 1.

- The I function: the bitwise logical XOR ($\oplus$) operation of the 128-bit input with a 128-bit round subkey of four 32-bit words.
- The B function: a non-linear substitution, which applies the same 4×4 S-box S_4 32 times in parallel to the input. For a 128-bit input (d', c', b', a'), the output (d'', c'', b'', a'') is obtained in the following way: $(d''_k, c''_k, b''_k, a''_k) = S_4(d'_k, c'_k, b'_k, a'_k)$, where X_k is the k-th bit of the word X $(0 \le k \le 31)$.
- The R function: a substitution-permutation Feistel structure, which consists of three subfunctions S, M and L. Each of the right two 32-bit words of the input to the R function is divided into 6 groups containing 6, 5, 5, 5, 5 and 6 bits, respectively. These six groups are then passed sequentially through the S function, consisting of two 6×6 S-boxes S_6 and four 5×5 S-boxes S_5, and

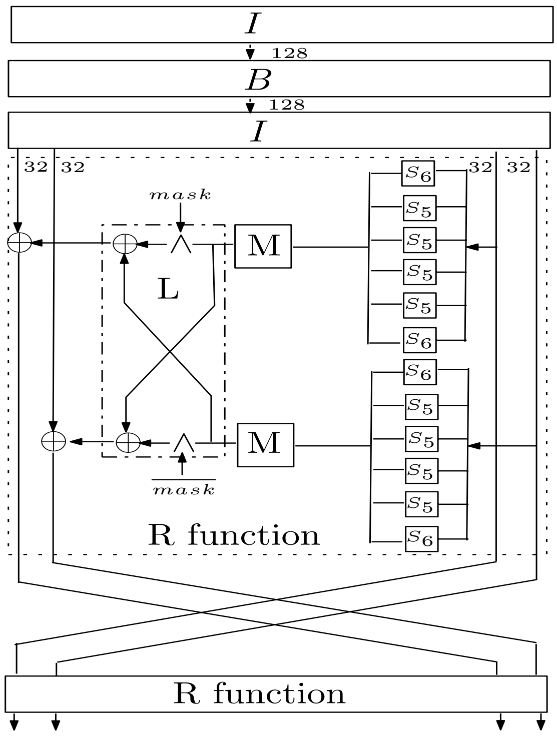


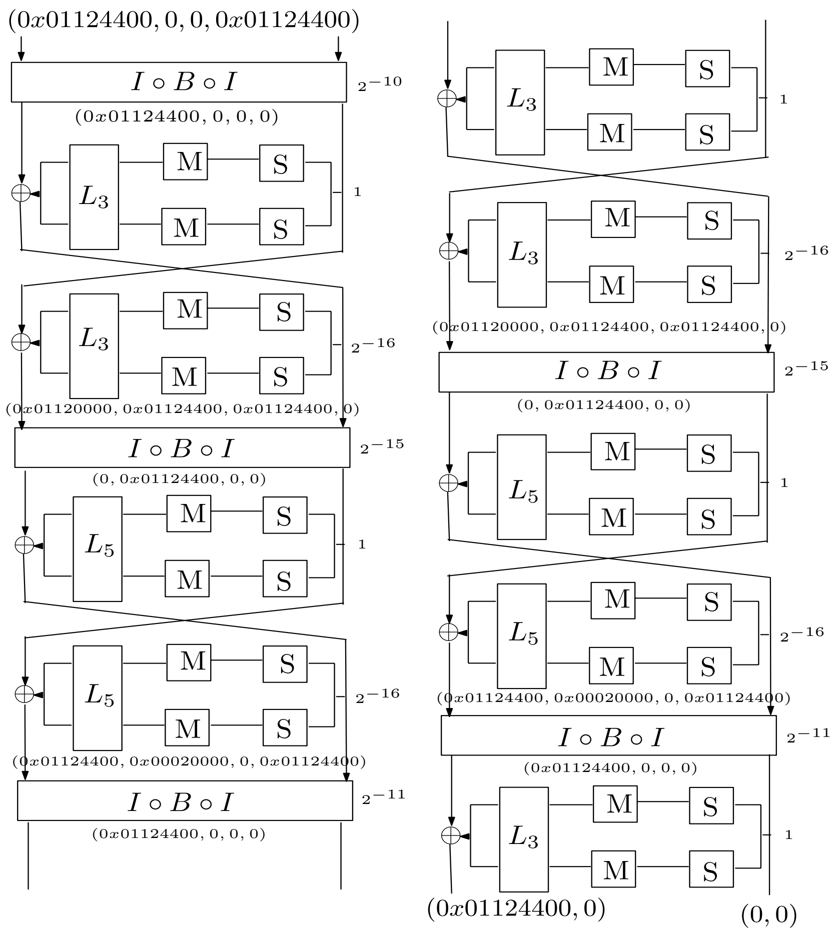
Fig. 1. The round function of SC2000

the linear M function that consists of 32 32-bit words $(M[0], \cdots, M[31])$. Given an input a, the output of the M function is defined as $a_0 \times M[0] \oplus \cdots \oplus a_{31} \times M[31]$. The outputs of the two M functions are then input to the L function. For a 64-bit input (a^*, b^*) the output of the L function is defined as $((a^* \wedge mask) \oplus b^*, (b^* \wedge \overline{mask}) \oplus a^*)$, where $mask$ is a constant (and $\overline{mask}$ is the complement of $mask$). Two masks $0x55555555$ and $0x33333333$ are used in SC2000, in the even and odd rounds, respectively. Finally, the output of the L function is XORed with the left two 32-bit words of the input to the R function, respectively. We denote the L and R functions with mask $0x55555555$ as L_5 and R_5, respectively, and the L and R functions with mask $0x33333333$ as L_3 and R_3, respectively.

The round function of SC200 is made up of two I functions, one B function and two R functions. We write K_j^i for the subkey used in the jth I function of Round i, and write $K_{j,l}^i$ for the l-th bit of K_j^i, where $0 \leq i \leq 6, j = 0, 1, 0 \leq l \leq 127$. The full 6.5-round encryption procedure of SC2000 can be described as: $I_{K_0^0} \circ B \circ I_{K_1^0} \circ R_5 \bowtie R_5 \circ I_{K_0^1} \circ B \circ I_{K_1^1} \circ R_3 \bowtie R_3 \circ I_{K_0^2} \circ B \circ I_{K_1^2} \circ R_5 \bowtie R_5 \circ I_{K_0^3} \circ B \circ I_{K_1^3} \circ R_3 \bowtie R_3 \circ I_{K_0^4} \circ B \circ I_{K_1^4} \circ R_5 \bowtie R_5 \circ I_{K_0^5} \circ B \circ I_{K_1^5} \circ R_3 \bowtie R_3 \circ I_{K_0^6} \circ B \circ I_{K_1^6}$. Note that we refer to the first round as Round 0. See [2] for its key schedule.

3 4.75-Round Differential Characteristics of SC2000

In this section we describe the 4.75-round differential characteristics.



(0x01124400, 0, 0, 0x01124400)
I ∘ B ∘ I
2^−10
(0x01124400, 0, 0, 0)
L3
M
S
1
2^−16
(0x01120000, 0x01124400, 0x01124400, 0)
2^−15
(0, 0x01124400, 0, 0)
L5
(0x01124400, 0x00020000, 0, 0x01124400)
2^−11
(0x01124400, 0)
(0, 0)

Fig. 2. A 4.75-round differential characteristic with probability 2^{-126}

3.1 2-Round Iterative Differential Characteristic of Yanami et al.

In 2002, Yanami et al. [12] described the results of a search over all the possible two-round iterative differential characteristics with only one active S function in every round for any two consecutive rounds $I \circ B \circ I \circ R_5 \bowtie R_5 \circ I \circ B \circ I \circ R_3 \bowtie R_3$. Their result is that the best two-round iterative differential characteristic (i.e. that with the highest probability) is $(\alpha, \beta, \beta, 0) \rightarrow (\alpha, \beta, \beta, 0)$ with probability 2^{-58}: $(\alpha, \beta, \beta, 0) \xrightarrow{I \circ B \circ I / 2^{-15}} (0, \beta, 0, 0) \xrightarrow{R_5 \bowtie R_5 / 2^{-16}} (\beta, \gamma, 0, \beta) \xrightarrow{I \circ B \circ I / 2^{-11}}$

$(\beta, 0, 0, 0) \xrightarrow{R_3 \bowtie R_3/2^{-16}} (\alpha, \beta, \beta, 0)$, where $\alpha = 0x01120000$, $\beta = 0x01124400$ and $\gamma = 0x00020000$.

3.2 The 4.75-Round Differential Characteristics

As a result, we can obtain a 4-round differential characteristic $(\alpha, \beta, \beta, 0) \rightarrow (\alpha, \beta, \beta, 0)$ with probability 2^{-116} by concatenating the above two-round iterative differential twice. It is essential to try to exploit an efficient (i.e. with a relatively high probability) differential operating over more than four rounds in order to break more rounds of SC2000. However, this 4-round differential cannot be extended to a differential characteristic operating over more than four rounds with a probability larger than 2^{-128}, as appending even a half round $R_3 \bowtie R_3$ at the beginning will cost a probability of 2^{-16} and appending a B function at the end will cost at least a probability of 2^{-13}.

Nevertheless, observe that from the above two-round iterative differential characteristic it follows that two-round iterative differential characteristic $(\beta, \gamma, 0, \beta) \rightarrow (\beta, \gamma, 0, \beta)$ for any two consecutive rounds $I \circ B \circ I \circ R_3 \bowtie R_3 \circ I \circ B \circ I \circ R_5 \bowtie R_5$ also holds with a probability of 2^{-58}: $(\beta, \gamma, 0, \beta) \xrightarrow{I \circ B \circ I/2^{-11}} (\beta, 0, 0, 0) \xrightarrow{R_3 \bowtie R_3/2^{-16}} (\alpha, \beta, \beta, 0) \xrightarrow{I \circ B \circ I/2^{-15}} (0, \beta, 0, 0) \xrightarrow{R_5 \bowtie R_5/2^{-16}} (\beta, \gamma, 0, \beta)$. It might seem counter-intuitive at first, but there is a major difference between this and the previous iterative 2-round differential characteristic: we can append a 0.75-round differential characteristic $(\beta, \gamma, 0, \beta) \xrightarrow{I \circ B \circ I \circ R_3} (\beta, 0, 0, 0)$ with a probability of 2^{-11} at the end of this differential characteristic! Therefore, we can obtain a 4.75-round differential characteristic $(\beta, \gamma, 0, \beta) \rightarrow (\beta, 0, 0, 0)$ with probability 2^{-127}. By changing the input difference to the difference $(\beta, 0, 0, \beta)$ we can get a 4.75-round differential characteristic with probability 2^{-126}, and this 4.75-round differential characteristic is depicted in Fig. 2. When we change the input difference for only one of the five active S_4 S-boxes to a value in $\{0x1, 0x2, 0x6, 0x7, 0xD, 0xF\}$, we get a total of thirty 4.75-round differential characteristics with probability 2^{-127}. We denote by $\mathbf{\Omega}$ the set of the 31 input differences for the thirty-one 4.75-round differential characteristics. The differential distribution table of the S_4 S-box is given in [12], and the differential distribution table of the S_5 S-box is shown in Table 2 in Appendix A. (The characteristics do not make an active S_6 S-box, so we do not give its differential distribution table.)

In a natural way, we might try to find a better differential characteristic on greater than four rounds by first exploiting short differentials with similar structures and then concatenating them, for the above 4.75-round differential obtained from the two-round iterative differential is just a special case among these. Motivated by this idea, we perform a computer search over all the possible differentials for such one round $R \bowtie R \circ I \circ B \circ I$ with only one R function active and the right two 32-bit input differences and one of the left two 32-bit input differences being zero; moreover, in order to ensure that the resulting differential is capable of being concatenated with itself, we also require that the right two 32-bit output words and one of the left two 32-bit output words have a zero difference. Surprisingly, we find that the differential characteristics $(\beta, 0, 0, 0) \xrightarrow{R_3 \bowtie R_3 \circ I \circ B \circ I} (0, \beta, 0, 0)$ and $(0, \beta, 0, 0) \xrightarrow{R_5 \bowtie R_5 \circ I \circ B \circ I} (\beta, 0, 0, 0)$ in the above two-round iterative differential are the best (i.e. with the highest probabilities) among those with the same forms, respectively. Our search for other similar forms gives no better result.

4 Differential Attack on 5-Round SC2000

In this section, we present a differential cryptanalysis attack on 5 rounds of SC2000 when used with a 128-bit key.

4.1 Preliminary Results

We first concentrate on the propagation of the output difference of the 4.75-round differential characteristic described above through the following $R_3 \circ I_{K_0^6}$ operation. The output difference $(\beta, 0, 0, 0)$ will definitely propagate to a difference with the form $(\Delta Y \wedge 0x33333333, \Delta Y, \beta, 0)$ after the following R_3 function, where $Y \in GF(2^{32})$. Since the S_5 function has a uniform differential probability of 2^{-4}, there are totally 2^{16} possible values for ΔY; we denote the set of all the 2^{16} possible differences $(\Delta Y \wedge 0x33333333, \Delta Y, \beta, 0)$ by $\mathbf{\Gamma}$. The difference $(\Delta Y \wedge 0x33333333, \Delta Y, \beta, 0)$ will be kept after the following linear $I_{K_0^6}$ function.

On the other hand, having known the 128-bit difference after the $I_{K_0^6}$ function for a ciphertext pair, we only need to guess the 64 subkey bits $(K_{0,64}^6, \cdots, K_{0,127}^6)$ of K_0^6 to check whether this pair could produce the difference $(\beta, 0, 0, 0)$ just before the adjacent R_3 function. For our case, as a candidate difference just after the $I_{K_0^6}$ function should be with the form $(\Delta Y \wedge 0x33333333, \Delta Y, \beta, 0)$, we only need to guess the 20 subkey bits $K_{0,70}^6, \cdots, K_{0,89}^6$ corresponding to the four active S_5 S-Boxes in the adjacent R_3 function to determine whether a ciphertext pair with a candidate difference could produce the output difference of the above 4.75-round differential characteristics.

4.2 Attack Procedure

By using the 4.75-round differential characteristics, we can mount a differential attack on the following 5 rounds of SC2000: $I_{K_0^1} \circ B \circ I_{K_1^1} \circ R_3 \bowtie R_3 \circ I_{K_0^2} \circ B \circ I_{K_1^2} \circ R_5 \bowtie R_5 \circ I_{K_0^3} \circ B \circ I_{K_1^3} \circ R_3 \bowtie R_3 \circ I_{K_0^4} \circ B \circ I_{K_1^4} \circ R_5 \bowtie R_5 \circ I_{K_0^5} \circ B \circ I_{K_1^5} \circ R_3 \bowtie R_3 \circ I_{K_0^6}$[1]. The attack procedure is as follows.

1. Choose 2^{107} structures, where a structure is defined to be a set of 2^{20} plaintexts with the 20 bits for the five active S_4 S-boxes taking all the possible values and the other 108 bits fixed. In a chosen-plaintext attack scenario, obtain the corresponding ciphertexts. Keep only the plaintext pairs that have a difference belonging to the set $\mathbf{\Omega}$ and whose ciphertext pairs have a difference belonging to the set $\mathbf{\Gamma}$.

[1] Strictly speaking, this is a little more than 5 rounds.

2. Guess the 20 subkey bits $(K^6_{0,70}, \cdots, K^6_{0,89})$ of K^6_0 in the $I_{K^6_0}$ function, and do as follows.
   (a) For every remaining ciphertext pair: Partially decrypt the corresponding 20 bits of the two ciphertexts through the $I_{K^6_0}$ function and the four active S_5 S-boxes in the adjacent R_3 operation, compute the 64-bit difference just after the L_3 operation in the R_3 operation, then XOR it with the left 64-bit difference of the ciphertext pair, and finally check whether the resultant 64-bit difference is zero.
   (b) Count the number of the ciphertext pairs with the 64-bit difference computed in Step 2(a) being zero, and record this number for the guessed $(K^6_{0,70}, \cdots, K^6_{0,89})$. Repeat Step 2 with another guess; and go to Step 3 if all the guesses are tested.
3. For each of the top m ranking guesses for $(K^6_{0,70}, \cdots, K^6_{0,89})$ according to the numbers recorded in Step 2(b), (specific values of m will be given below), exhaustively search for the remaining 108 key bits with two known plaintext/ciphertext pairs. If a 128-bit key is suggested, output it as the user key of the 5-round SC2000.

4.3 Complexity Analysis

The attack requires 2^{127} chosen plaintexts. Typically, encrypting chosen plaintexts is assumed to be done by some "challenger" who holds the user key (i.e. the challenger's running time), and is not counted as part of the time complexity of an attack. In Step 1, it is expected that $2^{107} \times \frac{(2^{20})^2}{2} \times \frac{31}{16^5} \approx 2^{130.96}$ plaintext pairs remain after the filtering condition about $\mathbf{\Omega}$, and $2^{130.96} \times \frac{2^{16}}{2^{128}} = 2^{18.96}$ ciphertext pairs remain after the filtering condition about $\mathbf{\Gamma}$; and it requires about $31 \times 2^{127} \approx 2^{132}$ memory accesses to filter out the satisfying ciphertext pairs. The time complexity of Step 2 is dominated by the partial decryptions, which is approximately $2 \cdot 2^{20} \cdot 2^{18.96} \cdot \frac{1}{4} \cdot \frac{1}{5} \approx 2^{36}$ 5-round SC2000 encryptions. Step 3 has a time complexity of $m \times 2^{108}$ 5-round SC2000 encryptions. The signal-to-noise ratio for the attack is $\frac{\frac{30}{31} \times 2^{-127} + \frac{1}{31} \times 2^{-126}}{2^{-128}} \approx 2^{1.04}$. In Step 2(b), for the correct key guess the number of the ciphertext pairs with the 64-bit difference computed in Step 2(a) being zero is expected to be $2^{130.96} \times (\frac{30}{31} \times 2^{-127} + \frac{1}{31} \times 2^{-126}) = 16$. According to Theorem 3 of [13], we have that the success probability for the attack is about 67% when $m = 1$, and is about 98.7% when $m = 2^{15}$.

5 Conclusions

SC2000 is a 128-bit block cipher, which is one of the CRYPTREC e-Government Recommended Ciphers. In this paper we have described a few 4.75-round differential characteristics with a probability of larger than 2^{-128}. Finally, using the 4.75-round differential characteristics we have presented a differential attack on 5-round SC2000 when used with 128 key bits. The presented attack is theoretical, like most cryptanalytic attacks on block ciphers; and from a cryptanalytic view it suggests for the first time that the safety margin of SC2000 with a 128-bit key decreases within one and a half rounds.

Acknowledgments

The author is very grateful to Prof. Chris Mitchell and the anonymous referees for their editorial comments.

References

1. Shimoyama, T., Yanami, H., Yokoyama, K., Takenaka, M., Itoh, K., Yajima, J., Torii, N., Tanaka, H.: The SC 2000 block cipher. In: Proceedings of the First Open NESSIE Workshop (2000)
2. Shimoyama, T., Yanami, H., Yokoyama, K., Takenaka, M., Itoh, K., Yajima, J., Torii, N., Tanaka, H.: The block cipher SC2000. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 312–327. Springer, Heidelberg (2002)
3. Fujitsu Laboratories, http://jp.fujitsu.com/group/labs/en/techinfo/technote/crypto/sc2000.html
4. Cryptography Research and Evaluatin Committees — CRYPTREC Report (2002), http://www.ipa.go.jp/security/enc/CRYPTREC/index-e.html
5. Biham, E., Shamir, A.: Differential cryptanalysis of the Data Encryption Standard. Springer, Heidelberg (1993)
6. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
7. Raddum, H., Knudsen, L.R.: A differential attack on reduced-round SC2000. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 190–198. Springer, Heidelberg (2001)
8. Biham, E., Dunkelman, O., Keller, N.: New results on boomerang and rectangle attacks. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 1–16. Springer, Heidelberg (2002)
9. Wagner, D.: The boomerang attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)
10. Biham, E., Dunkelman, O., Keller, N.: The rectangle attack — rectangling the Serpent. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 340–357. Springer, Heidelberg (2001)
11. Dunkelman, O., Keller, N.: Boomerang and rectangle attacks on SC2000. In: Proceedings of the Second Open NESSIE Workshop (2001)
12. Yanami, H., Shimoyama, T., Dunkelman, O.: Differential and linear cryptanalysis of a reduced-round SC2000. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 34–48. Springer, Heidelberg (2002)
13. Selçuk, A.A.: On probability of success in linear and differential cryptanalysis. Journal of Cryptology 21(1), 131–147 (2008)

A The Differential Distribution Table of the S_5 S-box

Table 2. The differential distribution table of the S_5 S-box

input	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	2	2	0	0	2	0	0	2	0	0	2	0	2	2	2	0	2	0	2	0	2	0	0	2	2	0	2	2
2	0	0	0	2	2	2	2	0	2	0	2	0	2	2	0	0	2	0	0	2	0	0	0	0	0	2	2	2	0	2	2	2
3	0	0	0	0	0	0	2	0	0	0	2	2	2	2	2	0	0	2	2	2	2	0	2	0	2	2	2	0	2	2	0	0
4	0	0	0	2	2	0	2	2	0	2	0	0	0	2	0	0	2	0	2	0	2	2	2	2	0	0	2	2	2	2	0	0
5	0	0	0	0	0	2	2	2	2	2	0	2	0	2	2	0	0	2	0	0	0	2	0	2	2	0	2	0	0	2	2	2
6	0	0	0	0	0	2	0	2	2	2	2	0	2	0	0	0	0	0	2	2	2	2	2	2	0	2	0	0	2	0	2	2
7	0	0	0	2	2	0	0	2	0	2	2	2	2	0	2	0	2	2	0	2	0	2	0	2	2	2	0	2	0	0	0	0
8	0	2	2	2	2	2	2	0	0	2	2	0	2	0	2	0	0	2	0	0	2	0	2	2	0	0	0	0	0	2	0	2
9	0	2	2	0	0	0	2	0	2	2	2	2	2	0	0	0	2	0	2	0	0	0	0	2	2	0	0	2	2	2	2	0
10	0	2	2	0	0	0	0	0	2	2	0	0	0	2	2	0	2	2	0	2	2	0	2	2	0	2	2	2	0	0	2	0
11	0	2	2	2	2	2	0	0	0	2	0	2	0	2	0	0	0	0	2	2	0	0	0	2	2	2	2	0	2	0	0	2
12	0	2	2	0	0	2	0	2	0	0	2	0	2	2	2	0	2	2	2	0	0	2	0	0	0	0	2	2	2	0	0	2
13	0	2	2	2	2	0	0	2	2	0	2	2	2	2	0	0	0	0	0	0	2	2	2	0	2	0	2	0	0	0	2	0
14	0	2	2	2	2	0	2	2	2	0	0	0	0	0	2	0	0	2	2	2	0	2	0	0	0	2	0	0	2	2	2	0
15	0	2	2	0	0	2	2	2	0	0	0	2	0	0	0	0	2	0	0	2	2	2	2	0	2	2	0	2	0	2	0	2
16	0	2	0	0	2	2	2	0	2	2	2	2	0	2	0	2	0	2	0	2	2	2	0	0	0	0	0	2	2	0	0	0
17	0	2	0	2	0	0	2	0	0	2	2	0	0	2	2	2	2	0	2	2	0	2	2	0	2	0	0	0	0	0	2	2
18	0	2	0	2	0	0	0	0	0	2	0	2	2	0	0	2	2	2	0	0	2	2	0	0	0	2	2	0	2	2	2	2
19	0	2	0	0	2	2	0	0	2	2	0	0	2	0	2	2	0	0	2	0	0	2	2	0	2	2	2	2	0	2	0	0
20	0	2	0	2	0	2	0	2	2	0	2	2	0	0	0	2	2	2	2	2	0	0	2	2	0	0	2	0	0	2	0	0
21	0	2	0	0	2	0	0	2	0	0	2	0	0	0	2	2	0	0	0	2	2	0	0	2	2	0	2	2	2	2	2	2
22	0	2	0	0	2	0	2	2	0	0	0	2	2	2	0	2	0	2	2	0	0	0	2	2	0	2	0	2	0	0	2	2
23	0	2	0	2	0	2	2	2	2	0	0	0	2	2	2	2	2	0	0	0	2	0	0	2	2	2	0	0	2	0	0	0
24	0	0	2	2	0	0	0	0	2	0	0	2	2	2	2	2	0	0	0	2	0	2	2	2	0	0	0	2	2	2	0	2
25	0	0	2	0	2	2	0	0	0	0	0	0	2	2	0	2	2	2	2	2	2	2	0	2	2	0	0	0	0	2	2	0
26	0	0	2	0	2	2	2	0	0	0	2	2	0	0	2	2	2	0	0	0	0	2	2	2	0	2	2	0	2	0	2	0
27	0	0	2	2	0	0	2	0	2	0	2	0	0	0	0	2	0	2	2	0	2	2	0	2	2	2	2	2	0	0	0	2
28	0	0	2	0	2	0	2	2	2	2	0	2	2	0	2	2	2	0	2	2	2	0	0	0	0	0	2	0	0	0	0	2
29	0	0	2	2	0	2	2	2	0	2	0	0	2	0	0	2	0	2	0	2	0	0	2	0	2	0	2	2	2	0	2	0
30	0	0	2	2	0	2	0	2	0	2	2	2	0	2	2	2	0	0	2	0	2	0	0	0	0	2	0	2	0	2	2	0
31	0	0	2	0	2	0	0	2	2	2	2	0	0	2	0	2	2	2	0	0	0	0	2	0	2	2	0	0	2	2	0	2

Pairing-Based Nominative Signatures with Selective and Universal Convertibility

Wei Zhao and Dingfeng Ye

State Key Laboratory of Information Security,
Graduate University of Chinese Academy of Sciences,
Beijing 100049, P.R. China
wzh@is.ac.cn

Abstract. A nominative signature scheme allows a nominator and a nominee jointly generate a signature in such a way that only the nominee can check the validity of the signature and further convince a third party of the fact. In Inscrypt 2008, Zhao et al. proposed selectively and universally convertible nominative signatures, which equips the nominee with additional ability to publish a *selective* proof to convert a nominative signature into a publicly verifiable one (i.e. selective convertibility), or issue a *universal* proof to make all nominative signatures with respect to the nominator and the nominee publicly verifiable (i.e. universal convertibility). Finally, they left an open problem to construct a selectively and universally convertible nominative signature scheme from bilinear pairings which is provably secure under the conventional assumptions. In this paper, based on standard digital signature and undeniable signature, we propose a new selectively and universally convertible nominative signature scheme from bilinear pairings. Our scheme is efficient which is a one-move (i.e. non-interactive) convertible nominative signature scheme, and possesses short signature length compared with Zhao et al.'s scheme. Moreover, formal proofs are given to show that our scheme is secure under some conventional assumptions in the random oracle model. Based on our construction and further analysis, we think that nominative signatures are just the dual form of undeniable signatures in the concept; whether their dual property in the construction of the schemes has generality needs further investigation.

Keywords: Nominative signatures, Convertible, Selective, Universal, Bilinear pairings, Probable security.

1 Introduction

Nominative signature (NS) is a cryptographic paradigm proposed by Kim et al. [9] to restrict the public verifiability of standard digital signature. In a nominative signature scheme, a nominator A (i.e. the signer) and a nominee B (i.e. the verifier) jointly generate a signature σ so that the validity of σ can only be verified by B. Furthermore, if σ is valid, B can convince a third party C of the validity of σ using confirmation protocol; otherwise, B can convince a third party C of the invalidity of σ using disavowal protocol. Compared with undeniable signatures [1,2], nominative signatures hand over the power of signature verification to the verifier B, so, it can be considered as the dual concept of undeniable signatures.

At ACISP 2004, the concept of convertible nominative signatures is introduced by Huang and Wang [7]. This new concept enables the nominee to convert a nominative signature into a publicly verifiable one. Moreover, Huang and Wang proposed a concrete scheme based on Kim et al.'s nominative signature scheme [9]. Then, Zhao et al. [17] further proposed selectively and universally convertible nominative signatures. "Selectively convertible" means that the nominee can use a selectively convert algorithm to generate a selective proof for a NS with respect to the nominator and the nominee. Thus, anyone can check the validity of this signature using the proof and the public keys of the nominator and the nominee. However, the validity of other nominative signatures remain unknown and can only be verified via the comfirmation/disavowal protocol with the help of the nominee. While "universally convertible" refers to the case where the nominee can use a universally convert algorithm to generate a universal proof which can convert all NS with respect to the nominator and the nominee into publicly verifiable ones. Thus, one can check the validity of any NS with respect to the nominator and the nominee without the help of the nominee. In fact, Huang-Wang's scheme is only a selectively convertible nominative signature scheme.

Since nominative signatures were proposed in 1996, it was not until recently that this notion has been formalized in Liu et al.'s work [12] at ICICS 2007. Liu et al. [12] defined the first formal security models for nominative signatures and pointed out that the security notions for nominative signature consist of unforgeability, invisibility, non-impersonation and non-repudiation. Moreover, they proposed the first provably secure nominative signature scheme based on Chaum's undeniable signature scheme [2] and a strongly unforgeable signature scheme. However, their construction requires multi-round communications between the nominator and the nominee for signature generation. So, Liu et al. [11] further proposed a one-round NS schemes based on ring signature, Liu et al. [5] further proposed a one-move NS scheme from bilinear pairings. As suggested in [9,7,12,5], (convertible) nominative signatures have potential applications in the scenarios where a signed message is personally private or commercially sensitive, such as protecting medical/academic records, user certification system.

Let's look at an application example of nominative signatures given in the work [5]: protecting patient's medical records. In this scenario, the patient acts as the nominee and the hospital acts as the nominator. The patient wants his/her medical records to be certified and signed by the hospital authority, and meanwhile does not want anybody (including the hospital) to disseminate his/her medical records. To realize this target, the patient and the hospital jointly generate a nominative signature on the patient's medical records, then the privacy of the patient's medical records can be protected. In some situations (for example, the patient engages in a lawsuit with the hospital), however, the patient can use

CNS to issue a selective proof or universal proof. Then, the judge and jury can easily checks the validity of the patient's medical records with these proofs, but does not need to execute the confirmation/disavowal protocol with the patient. In this case, the hospital cannot deny that it has signed the patient's medical records.

Related work. The first convertible nominative signature scheme was introduced by Huang and Wang [7]. However, it was found by several work [14,4,15] that the nominator in Huang-Wang's scheme can verify the validity of a nominative signature and also show to anyone that the nominative signature is indeed a valid one without the help of the nominee. Therefore, Huang-Wang's scheme fails to meet invisibility and non-impersonation of nominative signatures. Very recently, Zhao et al. [17] proposed an improvement to fix the flaws of Huang-Wang's scheme and showed the improved scheme is provably secure under some standard assumptions.

In addition, Liu et al. [10] proposed a selectively convertible nominative signature scheme based on ring signature and the proof protocols for verifiable decryption of discrete logarithm [3]. In Inscrypt 2008, Zhao et al. [16] proposed a new convertible nominative signature scheme from bilinear pairings which own selectively and universally convertible properties. However, their construction is provably secure under some non-standard assumptions in the random oracle model. Finally, Zhao et al. left an open problem to construct a selectively and universally convertible nominative signature scheme from bilinear pairings which is provably secure under some conventional hard assumptions.

Our contributions. In this paper, we first improve the security models for convertible nominative signatures proposed in [16]. Then we propose a new pairing-based selectively and universally convertible nominative signature scheme. Specially, our scheme adopts some techniques in the construction of undeniable signature scheme [6] and can be seen as a construction based on standard digital signature and convertible undeniable signature. Compared with Zhao er al.'s scheme [16], our scheme has short signature length, and is formally proven to satisfy all the security properties for nominative signatures under some conventional hard problems in the random oracle model.

Our construction seems to give a general method of constructing nominative signatures. However, we find that our method does not have generality. Based on our construction and further analysis, we think nominative signatures are just the dual form of undeniable signatures in the concept. Whether there exists a general construction of nominative signatures based on undeniable signatures needs further investigation, which will be used to decide whether the dual property in the construction of these two types of signatures has generality.

2 Preliminaries

Let $\mathbb{G}$ and $\mathbb{G}_1$ be cyclic groups of prime order p and g be the generator of $\mathbb{G}$. A bilinear pairing is a map $e: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ with the following properties:

1. Bilinear: $e(g^a, g^b) = e(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_p^*$,
2. Non-degenerate: $e(g, g) \neq 1_{\mathbb{G}_1}$,
3. Computable: e is efficiently computable.

The following problems are assumed to be hard for any polynomial time algorithm.

Computational Diffie-Hellman (CDH) Problem. Given a tuple (g, g^a, g^b) where $a, b \in_R \mathbb{Z}_p^*$, find g^{ab}.

3-Decisional Diffie-Hellman (3-DDH) Problem. Given a tuple (g, g^a, g^b, g^c, h) where $a, b, c \in_R \mathbb{Z}_p^*$ and $h \in_R \mathbb{G}$, decide whether $h \overset{?}{=} g^{abc}$.

2.1 Zero-Knowledge Interactive Proof

Given a tuple $(g, U, V, W) = (g, g^u, g^v, g^w)$ where u, v and $w \in \mathbb{Z}_p^*$, if $w = uv \bmod p$, then the tuple is a DH-tuple; otherwise, it is not a DH-tuple. Here we review perfect zero-knowledge interactive proof (ZKIP) protocols for languages of DH-tuple and non-DH-tuple, which were proposed by Chaum [2] and also described by Ogata et al. [13]. In the following figures, com(s') and decom(s') denote the commitment and the decommitment of s' respectively.

Signer		Verifier
		$a, b \xleftarrow{R} \mathbb{Z}_p$
	$\xleftarrow{c}$	$c = g^a V^b$
$r \xleftarrow{R} \mathbb{Z}_p$		
$z_1 = cg^r$		
$z_2 = z_1^u$	$\xrightarrow{z_1, z_2}$	
	$\xleftarrow{a, b}$	
$c \overset{?}{=} g^a V^b$	$\xrightarrow{r}$	
		$z_1 \overset{?}{=} g^{a+r} V^b$
		$z_2 \overset{?}{=} U^{a+r} W^b$

(a) ZKIP for DH-tuple

Signer		Verifier
		$s \xleftarrow{R} \{0, 1, \cdots, k\}$
		$a \xleftarrow{R} \mathbb{Z}_p$
		$c = g^a V^s$
	$\xleftarrow{c, c'}$	$c' = U^a W^s$
find s', s.t.		
$(c^u/c') = (V^u/W)^{s'}$	$\xrightarrow{\text{com}(s')}$	
	$\xleftarrow{a}$	
$c \overset{?}{=} g^a V^{s'}$	$\xrightarrow{\text{decom}(s')}$	
		$s' \overset{?}{=} s$

(b) ZKIP for non-DH-tuple

3 Definition and Security Models of Convertible Nominative Signatures

In this section, we review the definition and security models of *selectively* and *universally* convertible nominative signatures. Specially, the security models are defined more reasonable than Zhao et al.'s work [16]. Throughout the paper, we still denote by A, B and C the nominator, the nominee and the verifier (a third party) respectively as in Zhao et al.'s work.

3.1 Definition of Convertible Nominative Signatures

The convertible nominative signature scheme consists of the following algorithms and protocols:

System Setup: a probabilistic algorithm that on input 1^k where $k \in \mathbb{N}$ is a security parameter, generates the common parameters denoted by *cp*.

Key Generation: a probabilistic algorithm that on input *cp*, generates a public/secret key pair (pk, sk) for a user in the system.

Signing Protocol: an interactive (or non-interactive) algorithm. The common inputs of A and B are *cp* and a message m. A has an additional input pk_B, indicating that A nominates B as the nominee; and B has an additional input pk_A, indicating that A is the nominator. At the end of the protocol, either A or B outputs a convertible nominative signature σ, or $\perp$ indicating the failure of the protocol.

Signature Space : This is determined by pk_A and pk_B. We emphasize that the signature space has to be specified explicitly in the convertible nominative signature scheme.

Ver$^{\textbf{nominee}}$(nominee-only verification): a deterministic algorithm that on input *cp*, a message-signature pair (m, σ), a public key pk_A and a secret key sk_B, returns *valid* or *invalid*.

Confirmation/Disavowal Protocol: an interactive (or non-interactive) algorithm between B and C. On input *cp*, a message-signature pair (m, σ) and the public keys (pk_A, pk_B), B sets a bit μ to 1 if $\mathbf{Ver}^{\mathbf{nominee}}(m, \sigma, pk_A, sk_B)$ = *valid*; otherwise, μ is set to 0. B first sends μ to C. If $\mu = 1$, the **Confirmation Protocol** is carried out; otherwise, the **Disavowal Protocol** is carried out. At the end of the protocol, C outputs either *accept* or *reject* while B has no output.

Selectively Convert: a probabilistic (or deterministic) algorithm that on input *cp*, the public/secret key pair (pk_B, sk_B), the public key pk_A and a message-signature pair (m, σ), outputs a selective proof $P^{m,\ \sigma}_{pk_A, pk_B}$ of the given message-signature pair.

Selectively Verify: a deterministic algorithm that on input *cp*, the public keys pk_A and pk_B, a message-signature pair (m, σ) and the selective proof $P^{m,\ \sigma}_{pk_A, pk_B}$, outputs *valid* or *invalid*.

Universally Convert: a deterministic algorithm that on input *cp*, the public/secret key pair (pk_B, sk_B) and the public key pk_A, outputs the universal proof P_{pk_A, pk_B}.

Universally Verify: a deterministic algorithm that on input *cp*, the public keys pk_A and pk_B, any message-signature pair (m, σ) with respect to A and B and the universal proof P_{pk_A, pk_B}, outputs *valid* or *invalid*.

Correctness : Suppose that all the algorithms and protocols of a convertible nominative signature scheme are carried out by honest entities A, B and C, then the scheme is said to satisfy the correctness requirement if

1. $\mathbf{Ver}^{\mathbf{nominee}}(m, \sigma, pk_A, sk_B) = valid$;
2. C outputs *accept* at the end of the **Confirmation Protocol**;
3. On input (m, σ) together with a valid selective proof $P_{pk_A, pk_B}^{m,\ \sigma}$, the **Selectively Verify** algorithm outputs *valid*;
4. On input any message-signature pair (m, σ) together with a universal proof P_{pk_A, pk_B}, the **Universally Verify** algorithm outputs *valid*.

The security models of convertible nominative signatures will be defined using the game between an adversary $\mathcal{F}$ and a simulator $\mathcal{S}$. $\mathcal{F}$ is allowed to access the following oracles and submit its queries to $\mathcal{S}$ adaptively:

– CreateUser Oracle: On input an identity, say I, it generates a key pair (pk_I, sk_I) using the **Key Generation** algorithm and returns pk_I.
– Corrupt Oracle: On input a public key pk, if pk is generated by the CreateUser Oracle or in $\{pk_A, pk_B\}$, the corresponding secret key is returned; otherwise, $\perp$ is returned. pk is said to be corrupted.
– Signing Oracle: On input a message m, two distinct public keys pk_1 (the nominator) and pk_2 (the nominee) such that at least one of them is uncorrupted, and one parameter called $role \in$ {nil, nominator, nominee},
  • if *role* is nil, $\mathcal{S}$ simulates a run of the **Signing Protocol** and then returns a valid convertible nominative signature σ and a transcript of the execution of the **Signing Protocol**.
  • If *role* is nominator, $\mathcal{S}$ (as nominee with public key pk_2) simulates a run of the **Signing Protocol** with $\mathcal{F}$ (as nominator with public key pk_1).
  • If *role* is nominee, $\mathcal{S}$ (as nominator with public key pk_1) simulates a run of the **Signing Protocol** with $\mathcal{F}$ (as nominee with public key pk_2).
– Confirmation/Disavowal Oracle: On input a message-signature pair (m, σ) and two public keys pk_1 (nominator) and pk_2 (nominee). Let sk_2 be the corresponding secret key of pk_2, the oracle responds based on whether a passive attack or an active attack is mounted.
  • In a passive attack, if $\mathbf{Ver}^{\mathbf{nominee}}(m, \sigma, pk_1, sk_2) = valid$, the oracle returns a bit $\mu = 1$ and a transcript of the **Confirmation Protocol**. Otherwise, $\mu = 0$ and a transcript of the **Disavowal Protocol** is returned.
  • In an active attack, if $\mathbf{Ver}^{\mathbf{nominee}}(m, \sigma, pk_1, sk_2) = valid$, the oracle returns $\mu = 1$ and executes the **Confirmation Protocol** with $\mathcal{F}$ (acting as a verifier). Otherwise, the oracle returns $\mu = 0$ and executes the **Disavowal Protocol** with $\mathcal{F}$.
– Selectively Convert Oracle: On input a valid message-signature pair (m, σ) and two public keys pk_1 (nominator) and pk_2 (nominee), it runs the **Selectively Convert** algorithm to generate the selective proof $P_{pk_1, pk_2}^{m,\ \sigma}$ and returns it to $\mathcal{F}$.
– Universally Convert Oracle: On input two public keys pk_1 (nominator) and pk_2 (nominee), it runs the **Universally Convert** algorithm to generate the universal proof P_{pk_1, pk_2} and returns it to $\mathcal{F}$.

The security notions for convertible nominative signatures include: Unforgeability, Invisibility, Non-impersonation and Non-repudiation.

3.2 Unforgeability

The *existential unforgeability* means that an adversary should not be able to forge a valid convertible nominative signature if at least one of the secret keys of A and B is not known.

To discuss the unforgeability of convertible nominative signatures, the potential adversaries are divided into the following three types:

- **Adversary** $\mathcal{F}_0$ who has only the public keys of the nominator A and the nominee B.
- **Adversary** $\mathcal{F}_I$ who has the public keys of the nominator A and the nominee B and also has B's secret key;
- **Adversary** $\mathcal{F}_{II}$ who has the public keys of the nominator A and the nominee B and also has A's secret key.

It is obvious that if a convertible nominative signature scheme is unforgeable against $\mathcal{F}_I$ (or $\mathcal{F}_{II}$), then it is also unforgeable against $\mathcal{F}_0$.

Game Unforgeability ($\mathcal{F}_I$): Let $\mathcal{S}$ be the simulator and $\mathcal{F}_I$ be the adversary.

1. (Initialization Phase) Let $k \in \mathbb{N}$ be a security parameter. First, $cp \leftarrow$ **SystemSetup** (1^k) is executed and key pairs $(pk_A,\ sk_A)$ and $(pk_B,\ sk_B)$ for nominator A and nominee B, respectively, are generated using the **Key Generation** algorithm. $\mathcal{F}_I$ is invoked with inputs $(1^k,\ pk_A,\ pk_B)$.
2. (Attacking Phase) $\mathcal{F}_I$ can make queries to all the oracles defined in Section 3.1.
3. (Output Phase) $\mathcal{F}_I$ outputs a pair (m^*, σ^*).

$\mathcal{F}_I$ *wins* the game if $\mathbf{Ver}^{\mathbf{nominee}}(m^*, \sigma^*, pk_A, sk_B) = valid$ and (1) $\mathcal{F}_I$ has never corrupted pk_A; (2) $(m^*, pk_A, pk_B, role)$ has never been queried to the Signing Oracle for any valid value of *role*. $\mathcal{F}_I$'s advantage in this game is defined to be $Adv(\mathcal{F}_I) = \Pr[\mathcal{F}_I \text{ wins}]$.

Game Unforgeability ($\mathcal{F}_{II}$): It is defined similarly to the above game. Specially, the descriptions of all phases are the same as the above game, so we omit them. When all phases are over,

$\mathcal{F}_{II}$ *wins* the game if $\mathbf{Ver}^{\mathbf{nominee}}(m^*, \sigma^*, pk_A, sk_B) = valid$ and (1) $\mathcal{F}_{II}$ has never corrupted pk_B; (2) $(m^*, pk_A, pk_B, role)$ has never been queried to the Signing Oracle for any valid value of *role*. $\mathcal{F}_{II}$'s advantage in this game is defined to be $Adv(\mathcal{F}_{II}) = \Pr[\mathcal{F}_{II} \text{ wins}]$.

Definition 1. *A convertible nominative signature scheme is said to be existential unforgeable if no probabilistic polynomial time (PPT) adversaries $\mathcal{F}_I$ and $\mathcal{F}_{II}$ have a non-negligible advantage in the above games.*

3.3 Invisibility

This property essentially means that it is impossible for an adversary (even the nominator A) to determine whether a given message-signature pair (m, σ) is valid without the help of the nominee, the selective proof $P_{pk_A,pk_B}^{m,\ \sigma}$ or the universal proof P_{pk_A,pk_B}.

Game Invisibility: Let $\mathcal{D}'$ be the simulator and $\mathcal{D}$ be the distinguisher.

1. (Initialization Phase) The initialization phase is the same as that of **Game Unforgeability**.
2. (Preparation Phase) At the beginning of this phase, $\mathcal{D}$ can adaptively access to all the oracles defined in Section 3.1. When all queries finish, $\mathcal{D}$ submits the challenge $(m^*, pk_A, pk_B, role)$ to the Signing Oracle with the restrictions that:

   (a) pk_B has not been submitted to the Corrupt Oracle;
   (b) $(m^*, pk_A, pk_B, role)$ has not been submitted to the Signing Oracle;
   (c) (pk_A, pk_B) has not been submitted to the Universally Convert Oracle.

   Then $\mathcal{D}'$ (acting as nominee) will carry out a run of the **Signing Protocol** with $\mathcal{D}$ (acting as nominator). Let σ^{valid} be the convertible nominative signature generated by $\mathcal{D}'$ at the end of the protocol. Note that $\mathbf{Ver}^{\mathbf{nominee}}(m^*, \sigma^{valid}, pk_A, sk_B) = valid$.

   The challenge signature σ^* is then generated based on the outcome of a random coin toss b. If $b = 1$, $\mathcal{D}'$ sets $\sigma^* = \sigma^{valid}$. If $b = 0$, σ^* is chosen uniformly at random from the signature space of the convertible nominative signature scheme with respect to pk_A and pk_B. Then the challenge signature σ^* is returned to $\mathcal{D}$.

   After receiving the challenge signature σ^*, $\mathcal{D}$ can still access all the oracles adaptively except that:

   (a) pk_B cannot be submitted to the Corrupt Oracle;
   (b) $(m^*, pk_A, pk_B, role)$ cannot be submitted to the Signing Oracle;
   (c) $(m^*, \sigma^*, pk_A, pk_B)$ cannot be submitted to the Confirmation/Disavowal Oracle and Selectively Convert Oracle;
   (d) (pk_A, pk_B) can not be submitted to the Universally Convert Oracle.
3. (Guessing Phase) Finally, $\mathcal{D}$ outputs a guess b'.

$\mathcal{D}$ *wins* the game if $b' = b$. $\mathcal{D}$'s advantage in this game is defined to be $Adv(\mathcal{D}) = |\Pr[b' = b] - \frac{1}{2}|$.

Definition 2. *A convertible nominative signature scheme is said to have the property of invisibility if no PPT distinguisher $\mathcal{D}$ has a non-negligible advantage in the above game.*

3.4 Non-impersonation

Non-impersonation means that the validity of a nominative signature can only be determined by the help of the nominee, someone else (even the nominator A) should not be able to show the validity of the nominative signature to a third party. Concretely, this notion requires that:

1. Only with the knowledge of the public key of the nominee B, it should be difficult for an impersonator $\mathcal{I}_I$ to execute the **Confirmation/Disavowal Protocol**.
2. Only with the knowledge of the public key of the nominee B, it should be difficult for an impersonator $\mathcal{I}_{II}$ to generate the selective proof for a message-signature pair.
3. Only with the knowledge of the public key of the nominee B, it should be difficult for an impersonator $\mathcal{I}_{III}$ to generate the universal proof.

Game Impersonation of Confirmation/Disavowal Protocol: Let $\mathcal{S}$ be the simulator and $\mathcal{I}_I$ be the impersonator.

1. (Initialization Phase) The initialization phase is the same as that of **Game Unforgeability**.
2. (Preparation Phase) In this phase, $\mathcal{I}_I$ is permitted to access all the oracles defined in Section 3.1. At some point, $\mathcal{I}_I$ prepares a triple (m^*, σ^*, μ) where m^* is some message, σ^* is a convertible nominative signature and μ is a bit.
3. (Attacking Phase) If $\mu = 1$, $\mathcal{I}_I$ (as nominee) executes the **Confirmation Protocol** with $\mathcal{S}$ (as a verifier) on common inputs $(m^*, \sigma^*, pk_A, pk_B)$. If $\mu = 0$, $\mathcal{I}_I$ executes the **Disavowal Protocol** with $\mathcal{S}$ on the same inputs.

$\mathcal{I}_I$ *wins* the game if $\mathcal{S}$ acting as the verifier outputs *accept* while $\mathcal{I}_I$ has the following restriction: $\mathcal{I}_I$ has never submitted pk_B to the Corrupt Oracle. $\mathcal{I}_I$'s advantage in this game is defined to be $Adv(\mathcal{I}_I)$ =Pr[$\mathcal{I}_I$ wins].

Game Impersonation of Selectively Convert Algorithm: Let $\mathcal{S}$ be the simulator and $\mathcal{I}_{II}$ be the impersonator.

1. (Initialization Phase) The initialization phase is the same as that of **Game Unforgeability**.
2. (Preparation Phase) $\mathcal{I}_{II}$ is invoked on input $(1^k, pk_A, pk_B)$ and permitted to issue queries to all the oracles defined in Section 3.1. At some point, $\mathcal{I}_{II}$ submits the challenge (m^*, pk_A, pk_B, $role$) to the Signing Oracle. Then $\mathcal{S}$ (acting as nominee) will carry out a run of the **Signing Protocol** with $\mathcal{I}_{II}$ (acting as nominator) and return the signature σ^* generated by $\mathcal{S}$ at the end of the protocol to $\mathcal{I}_{II}$.
3. (Impersonation Phase) $\mathcal{I}_{II}$ outputs a valid selective proof $P^{m^*,\sigma^*}_{pk_A,pk_B}$ for a message-signature pair (m^*, σ^*).

$\mathcal{I}_{II}$ *wins* the game if $P^{m^*,\sigma^*}_{pk_A,pk_B}$ satisfies the **Selectively Verify** algorithm but: (1) $\mathcal{I}_{II}$ has never submitted pk_B to the Corrupt Oracle; (2) $(m^*, \sigma^*, pk_A, pk_B)$ has never queries the Selectively Convert Oracle. $\mathcal{I}_{II}$'s advantage in this game is defined to be $Adv(\mathcal{I}_{II})$ =Pr[$\mathcal{I}_{II}$ wins].

Game Impersonation of Universally Convert Algorithm: Let $\mathcal{S}$ be the simulator and $\mathcal{I}_{III}$ be the impersonator.

1. (Initialization Phase) The initialization phase is the same as that of **Game Unforgeability**.

2. (Preparation Phase) $\mathcal{I}_{III}$ is invoked on input $(1^k, pk_A, pk_B)$ and permitted to issue queries to all the oracles defined in Section 3.1.
3. (Impersonation Phase) $\mathcal{I}_{III}$ outputs a valid universal proof P_{pk_A,pk_B}.

$\mathcal{I}_{III}$ *wins* the game if P_{pk_A,pk_B} satisfies the **Universally Verify** algorithm but: (1) pk_B has never been submitted to the Corrupt Oracle; (2) (pk_1, pk_B) has never been queried to the Universally Convert Oracle where pk_1 is generated by the CreateUser Oracle. $\mathcal{I}_{III}$'s advantage in this game is defined to be $Adv(\mathcal{I}_{III}) = \Pr[\mathcal{I}_{III} \text{ wins}]$.

Definition 3. *A convertible nominative signature scheme is said to be secure against impersonation if no PPT impersonators $\mathcal{I}_I$, $\mathcal{I}_{II}$ and $\mathcal{I}_{III}$ have a non-negligible advantage in the above games.*

3.5 Non-repudiation

Non-repudiation requires that the nominee B cannot convince a verifier C that a valid (invalid) convertible nominative signature is invalid (valid).

Game Non-repudiation: Let $\mathcal{S}$ be the simulator and $\mathcal{B}$ be the cheating nominee.

1. (Initialization Phase) The initialization phase is the same as that of **Game Unforgeability**.
2. (Preparation Phase) $\mathcal{B}$ prepares (m^*, σ^*, μ) where m^* is some message and σ^* is a nomnative signature. $\mu = 1$ if $\mathbf{Ver}^{\mathbf{nominee}}(m^*, \sigma^*, pk_A, sk_B) = valid$; otherwise, $\mu = 0$.
3. (Repudiation Phase) If $\mu = 1$, $\mathcal{B}$ executes the **Disavowal Protocol** with $\mathcal{S}$ (acting as a verifier) on $(m^*, \sigma^*, pk_A, pk_B)$ but the first bit sent to $\mathcal{S}$ is 0. If $\mu = 0$, $\mathcal{B}$ executes the **Confirmation Protocol** with $\mathcal{S}$ but the first bit sent to $\mathcal{S}$ is 1.

$\mathcal{B}$ *wins* the game if $\mathcal{S}$ acting as the verifier outputs *accept*. $\mathcal{B}$'s advantage in this game is defined to be $Adv(\mathcal{B}) = \Pr[\mathcal{B} \text{ wins}]$.

Definition 4. *A convertible nominator signature scheme is said to be secure against repudiation by nominee if no PPT cheating nominee $\mathcal{B}$ has a non-negligible advantage in the above game.*

4 Proposed Scheme and Security Analysis

Our selectively and universally convertible nominative signature scheme adopts some construction technique of undeniable signature scheme [6]. It consists of the following algorithms and protocols:

System Setup: Given the system parameter $k \in \mathbb{N}$, the algorithm first generates two cyclic groups $\mathbb{G}$, $\mathbb{G}_1$ of prime order $p \geq 2^k$, a generator g of $\mathbb{G}$ and a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ with properties described in Section 2. Then it generates two different hash functions $H_1, H_2 : \{0,1\}^* \rightarrow \mathbb{G}$. The common parameters are $cp = (p, \mathbb{G}, \mathbb{G}_1, e, g, H_1, H_2)$.

Key Generation: On input cp, the algorithm generates the secret key $sk_A = (x_A, y_A)$ where $x_A, y_A \in_R \mathbb{Z}_p^*$ and public key $pk_A = (X_A, Y_A) = (g^{x_A}, g^{y_A})$ for nominator A. Similarly, let (sk_B, pk_B) be the public/secret key pair of nominee B.

Signing Protocol: On input a message $m \in \{0,1\}^*$, the convertible nominative signature is generated by carrying out the following protocol between nominator A and nominee B.

1. A first computes the value $s = H_1(m\|pk_A\|pk_B)^{x_A}$ and sends (m, s) to B.
2. B checks
$$e(s, g) \overset{?}{=} e(H_1(m\|pk_A\|pk_B), X_A)$$
If the above equation is incorrect, B outputs $\perp$ for failure; otherwise, B computes the convertible nominative signature
$$\sigma = s^{x_B y_B} \cdot H_2(m\|pk_A\|pk_B)^{y_B}$$

Signature Space : We say σ is a convertible nominative signature if $\sigma \in \mathbb{G}$.

Ver$^{\textbf{nominee}}$: On input (m, σ, pk_A, pk_B), the nominee B checks
$$e(\sigma, g) \overset{?}{=} e(H_1(m\|pk_A\|pk_B), X_A)^{x_B y_B} \cdot e(H_2(m\|pk_A\|pk_B), Y_B)$$
If the above equation is correct, outputs *valid*; otherwise, outputs *invalid.*

Confirmation/Disavowal Protocol: On input (m, σ, pk_A, pk_B),

1. Nominee B first runs **Ver$^{\textbf{nominee}}$**(m, σ, pk_A, sk_B). If the output is *valid*, B sends $\mu = 1$ and $t = X_A^{x_B}$ to a verifier C. Otherwise, B sends $\mu = 0$ to C.
2. For the case $\mu = 1$, C checks $e(t, g) \overset{?}{=} e(X_A, X_B)$. If the above equation is correct, C sends $\nu = 1$ to B; otherwise, aborts. For another case $\mu = 0$, C sends $\nu = 0$ to B.
3. Upon receiving ν, if $\nu = 1$, B proves to C that the following tuple
$$(e(g, g), e(Y_B, g), e(H_1(m\|pk_A\|pk_B), t), e(\sigma, g)/e(H_2(m\|pk_A\|pk_B), Y_B))$$
is a DH-tuple using ZKIP protocol for DH-tuple described in Section 2.1; otherwise, B proves to C that the above tuple is a non-DH-tuple using ZKIP protocol for non-DH-tuple described in Section 2.1.

Selectively Convert: B computes the selective proof $P_{pk_A, pk_B}^{m,\sigma}$ of message-signature pair (m, σ) where
$$P_{pk_A, pk_B}^{m,\sigma} = (H_1(m\|pk_A\|pk_B)^{y_B}, X_A^{x_B})$$

Selectively Verify: On input a message-signature pair (m, σ) with respective to A and B, and the selective proof $P_{pk_A, pk_B}^{m,\sigma} = (P_1^{SC}, P_2^{SC})$,

1. anyone can verify $e(P_1^{SC}, g) \overset{?}{=} e(H_1(m\|pk_A\|pk_B), Y_B)$ and $e(P_2^{SC}, g) \overset{?}{=} e(X_A, X_B)$. If both equalities are satisfied, continue to the next step. Otherwise, $P_{pk_A, pk_B}^{m,\sigma}$ is invalid.

2. verify $e(\sigma, g) \stackrel{?}{=} e(P_1^{SC}, P_2^{SC}) \cdot e(H_2(m\|pk_A\|pk_B), Y_B)$. If this equality is satisfied as well, one can accept σ as a *valid* nominative signature. Otherwise, it is *invalid.*

Universally Convert: B computes the universal proof

$$P_{pk_A,pk_B} = (X_A^{x_B}, X_A^{x_B y_B})$$

Universally Verify: For any message-signature pair (m, σ) with respect to A and B and the universal proof $P_{pk_A,pk_B} = (P_1^{UC}, P_2^{UC})$,

1. anyone can verify $e(P_1^{UC}, g) \stackrel{?}{=} e(X_A, X_B)$ and $e(P_2^{UC}, g) \stackrel{?}{=} e(P_1^{UC}, Y_B)$. If both equalities are satisfied, continue to the next step. Otherwise, P_{pk_A,pk_B} is *invalid.*
2. verify $e(\sigma, g) \stackrel{?}{=} e(H_1(m\|pk_A\|pk_B), P_2^{UC}) \cdot e(H_2(m\|pk_A\|pk_B, Y_B)$. If this equality is satisfied as well, one can accept σ as a *valid* nominative signature. Otherwise, it is *invalid.*

Remark: In Zhao et al.'s work [16], the authors proposed a universally and selectively convertible nominative signature scheme by using WI protocols proposed in Kurosawa and Heng's work [8] as the confirmation/disavowal protocol. However, Ogata et al. [13] stated that Kurosawa et al.'s 3-move undeniable signature scheme which used WI protocols as the confirmation/disavowal protocol does not satisfy non-impersonation property against active attack. Thus, Zhao et al.'s scheme also does not satisfy non-impersonation of confirmation/disavowal protocol against active attack. Therefore, we in our scheme employ Chaum's perfect zero-knowledge interactive protocol described in Section 2.1 as the building block. As stated in Ogata et al.'s work, Chaum's ZKIP is secure against active attack.

4.1 Discussion

Our scheme can be seen as a construction based on two building blocks: standard digital signature and undeniable signature. It just needs one-round communication between the nominator and the nominee to generate nominative signature, that is, the nominator first computes the standard digital signature s on $m\|pk_A\|pk_B$ and then the nominee computes the "undeniable signature" [6] on s (Although our construction has similar form with Zhao et al.'s scheme [16], the construction ideas of these two schemes are different and our scheme has shorter signature length). However, we find that our method does not have generality, that is, our method can not be extended to a generic construction of nominative signatures.

Note that in the work [12], Liu et al. proposed a method to construct nominative signature based on undeniable signature and strongly unforgeable standard digital signature, and stated that their method can be generalized to a generic construction of nominative signatures. However, their construction needs multi-round communications between the nominator and the nominee and does not satisfy the invisibility (Note that given a challenge $(m^*, \sigma^* = (\sigma^{undeni}, \sigma^{standard}))$ in Liu et al.'s nominative signature scheme, an adversary with the secret key of nominator can generate another nominative signature $\sigma' = (\sigma^{undeni}, (\sigma^{standard})')$ of m^*. Based on the security model for invisibility of nominative signatures defined in Liu et al.'s work , the adversary can decide whether (m^*, σ^*) is valid according to whether (m^*, σ') is valid. Therefore, the adversary can always break the invisibility of Liu et al.'s scheme).

From above discussions, we can see that although undeniable signatures is the dual concept of nominative signatures, it is not trivial to construct nominative signatures based on undeniable signatures. Therefore, whether there exists a generic construction of nominative signatures which is based on undeniable signatures and just needs one-round communications (or not) between the nominator and the nominee to generate nominative signature deserves further investigation. The answer to this problem will decide whether the dual property in the construction of nominative signatures and undeniable signatures has generality.

4.2 Security Analysis

In this section, we give a formal security analysis of our proposed scheme in the random oracle model. Due to page limitation, we leave all the security proofs in the full version of this paper.

Theorem 1 (Unforgeability). *The proposed convertible nominative signature scheme is existential unforgeable if CDH problem is hard.*

Theorem 2 (Invisibility). *The proposed convertible nominative signature scheme has the property of invisibility if 3-DDH problem is hard.*

Theorem 3 (Non-impersonation). *The proposed convertible nominative signature is secure against impersonation if CDH problem is hard.*

Theorem 4 (Non-repudiation). *The proposed convertible nominative signature scheme is secure against repudiation by nominee.*

4.3 Efficiency Analysis and Comparison

Now we make an efficiency analysis of our scheme and a comparison between our scheme and Zhao et al's scheme [16].

Since the introduction of nominative signatures, there are only several secure scheme [10,11,12,5,16]. Among these schemes, only Zhao et al's scheme [16] is constructed from bilinear pairings and owns both selectively and universally convertible property. Their scheme requires that the nominator computes 1 hash mapping to $\mathbb{G}$ and 1 exponentiation in $\mathbb{G}$, and the nominee computes 2 pairings, 2 exponentiations in $\mathbb{G}$ and 1 multiplication in $\mathbb{Z}_p^*$ in the signature generation. In our scheme, to generate a signature, the nominator needs to compute 1 hash mapping to $\mathbb{G}$ and 1 exponentiation in $\mathbb{G}$, while the nominee needs to compute 2 pairings, 1 multiplication in $\mathbb{Z}_p^*$, 1 hash mapping to $\mathbb{G}$, 2 exponentiation in $\mathbb{G}$ and 1 multiplication in $\mathbb{G}$. In addition, the nominator and nominee in Zhao et al's scheme needs one pair of keys respectively, and the nominator and nominee in

Table 1. Comparison of pairing-based schemes

Scheme	L_s	Unforgeability	Invisibility	Non-impersonation	N_{key}
Scheme in [16]	$2\lvert\mathbb{G}\rvert$	WCDH-I WCDH-II	WDDH	WDLOG WCDH-III WCDH-IV	1
Our Scheme	$\lvert\mathbb{G}\rvert$	CDH	3-DDH	CDH	2

Notations. $|\cdot|$ means that the bit length of an element in the group $\mathbb{G}$, L_s denotes the signature length, and N_{key} means the number of key pair employed by the user in the system.

our scheme needs double key pairs respectively. Hence, compared with Zhao et al's scheme, our scheme is slightly less efficient. However, our scheme is provably secure under some conventional assumptions and enjoys short signature length which is better than Zhao et al's scheme. We give the detailed comparison in the **Table 1**.

In addition, the nominee in our scheme can precompute some pairings, such as $e(H_1(m\|pk_A\|pk_B), X_A)$, $e(H_2(m\|pk_A\|pk_B), Y_B)$, $e(H_1(m\|pk_A\|pk_B), X_A^{x_B})$, $e(g,g)$ and $e(Y_B, g)$ to execute the **Signing Protocol**, the $\mathbf{Ver}^{\mathbf{nominee}}$ algorithm and the **Confirmation/Disavowal Protocol** efficiently; the verifier can percompute $e(H_1(m\|pk_A\|pk_B), Y_B)$, $e(H_2(m\|pk_A\|pk_B), Y_B)$, $e(X_A, X_B)$, $e(g,g)$ and $e(Y_B, g)$ to reduce the computation overhead when he verifies a nominative signature with respect to A and B.

5 Conclusion

In this paper, we proposed a selectively and universally convertible nominative signatures from bilinear pairings which is provably secure under several conventional assumptions in the random oracle model. The signature length of our scheme is short; meanwhile, our scheme is efficient which requires only one-move message transfer from the nominator to the nominee for signature generation. Based on our construction and further discussion, we think the nominative signatures are just the dual form of undeniable signatures in the concept; whether the dual property in the construction of these two types of signatures has generality needs further investigation.

Acknowledgements. The authors wish to thank the anonymous reviewers for the valuable suggestions to improve the paper. The authors also thank Dr. Xinyi Huang for providing the full version of his paper and answering our questions.

References

1. Chaum, D., Antwerpen, H.V.: Udeniable signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 212–216. Springer, Heidelberg (1990)
2. Chaum, D.: Zero-knowledge udeniable signature. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 458–464. Springer, Heidelberg (1991)

3. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithm. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
4. Guo, L., Wang, G., Wong, D., Hu, L.: Further discussions on the security of a nominative signature scheme. In: The 2007 International Conference on Security & Management - SAM 2007, pp. 566–572. CSREA Press (2007), Cryptology ePrint Archive, Report 2006/007
5. Huang, Q., Liu, D.Y.W., Wong, D.S.: An efficient one-move nominative signature scheme. International Journal of Applied Cryptography (IJACT) 1(2), 133–143 (2008)
6. Huang, X., Mu, Y., Susilo, W., Wu, W.: Provably secure pairing-based convertible undeniable signature with short signature length. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 367–391. Springer, Heidelberg (2007)
7. Huang, Z., Wang, Y.: Convertible nominative signatures. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 348–357. Springer, Heidelberg (2004)
8. Kurosawa, K., Heng, S.: 3-Move undeniable signature scheme. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 181–197. Springer, Heidelberg (2005)
9. Kim, S.J., Park, S.J., Won, D.H.: Zero-knowledge nominative sinatures. In: Pragocrypt 1996, International Conference on the Theory and Applications of Cryptology, pp. 380–392 (1996)
10. Liu, D.Y.W., Chang, S., Wong, D.S.: A more efficient convertible nominative signature. In: International Conference on Security and Cryptography - SECRYPT 2007, pp. 214–221 (2007)
11. Liu, D.Y.W., Chang, S., Wong, D.S., Mu, Y.: Nominative signature from ring signature. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) IWSEC 2007. LNCS, vol. 4752, pp. 396–411. Springer, Heidelberg (2007)
12. Liu, D.Y.W., Wong, D.S., Huang, X., Wang, G., Huang, Q., Mu, Y., Susilo, W.: Formal definition and construction of nominative signature. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 57–68. Springer, Heidelberg (2007)
13. Ogata, W., Kurosawa, K., Heng, S.H.: The secutity of the FDH variant of Chaum's undeniable signature scheme. IEEE Tansactions on Information Theory 52(5), 2006–2017 (2006)
14. Susilo, W., Mu, Y.: On the security of nominative signatures. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 329–335. Springer, Heidelberg (2005)
15. Wang, G., Bao, F.: Security remarks on a convertible nominative signature scheme. In: Venter, H., Eloff, M., Labuschagne, L., Eloff, J., von Solms, R. (eds.) IFIP International Federation for Information Processing. New Approaches for Security, Privacy and Trust in Complex Environments, vol. 232, pp. 265–275. Springer, Boston (2007)
16. Zhao, W., Lin, C.L., Ye, D.F.: Provably secure convertible nominative signature scheme. In: Yung, M., Liu, P., Lin, D. (eds.) Inscrypt 2008. LNCS, vol. 5487, pp. 23–40. Springer, Heidelberg (2009)
17. Zhao, W., Ye, D.: Modified Huang-Wang's convertible nominative signature scheme. In: Proc. of the 9th International Conference for Young Computer Scientists - ICYCS 2008, pp. 2090–2095. IEEE Computer Society, Los Alamitos (2008)

Cryptanalysis of Certificateless Signcryption Schemes and an Efficient Construction without Pairing

S. Sharmila Deva Selvi, S. Sree Vivek*, and C. Pandu Rangan*

Theoretical Computer Science Lab,
Department of Computer Science and Engineering,
Indian Institute of Technology Madras, India
{sharmila,svivek,prangan}@cse.iitm.ac.in

Abstract. Certificateless cryptography introduced by Al-Riyami and Paterson eliminates the key escrow problem inherent in identity based cryptosystems. Even though building practical identity based signcryption schemes without bilinear pairing are considered to be almost impossible, it will be interesting to explore possibilities of constructing such systems in other settings like certificateless cryptography. Often for practical systems, bilinear pairings are considered to induce computational overhead. Signcryption is a powerful primitive that offers both confidentiality and authenticity to noteworthy messages. Though some prior attempts were made for designing certificateless signcryption schemes, almost all the known ones have security weaknesses. Specifically, in this paper we demonstrate the security weakness of the schemes in [2], [1] and [6]. We also present the first provably secure certificateless signcryption scheme without bilinear pairing and prove it in the random oracle model.

Keywords: Certificateless Signcryption, Provable Security, Pairing-free Cryptosystem, Random Oracle model, Cryptanalysis.

1 Introduction

To the best of our knowledge, there exist four ([2], [1], [6] and [3]) certificateless signcryption schemes (CLSC) in the literature. Among these four, [2], [1] and [6] are pairing based and [3] uses pairing for public key verification alone. In this paper, we show the security weaknesses in [2], [1] and [6]. We also present a provably secure certificateless signcryption scheme without pairing. Our scheme is the first provably secure certificateless signcryption scheme without pairing. The newly proposed CLSC scheme uses a key construct similar to that of [5] but uses a completely different approach for encryption. Any signcryption scheme is strongly secure if attacks by the insider is considered. Our security model

* Work supported by Project No. CSE/05-06/076/DITX/CPAN on Protocols for Secure Communication and Computation sponsored by Department of Information Technology, Government of India.

considers insider security and we have proved the security of our scheme in the random oracle model. It is to be noted that signcryption schemes are not directly obtained by combining a digital signature scheme and an encryption schemes. The security requirements for signcryption schemes are entirely different from encryption and digital signatures. The notion of insider security comes into picture when we talk about signcryption. This is because the private information of the sender and the public information of the receiver is involved in signcryption schemes. Thus, the certificateless signcryption scheme presented here is not a trivial extension of a signature scheme clubbed with an encryption scheme.

2 Preliminaries

We refer to [4] for a brief review of Discrete Logarithm Problem (DLP) and Computational Diffie-Hellman Problem (CDHP). We refer [2] for the framework of certificateless signcryption scheme.

2.1 Security Model of CLSC

The confidentiality proof of any CLSC scheme can be viewed as an interactive game, namely IND-CLSC-CCA2 between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. Similarly, the unforgeability proof of CLSC can be viewed as an interactive game namely EUF-CLSC-CMA, between a challenger $\mathcal{C}$ and a forger $\mathcal{F}$. In both the IND-CLSC-CCA2 and EUF-CLSC-CMA games, $\mathcal{A}$ and $\mathcal{F}$ are given access to some or all of the following six oracles (depending on their type). These oracles are simulated by $\mathcal{C}$:

- **Partial Private Key Extract of ID_A:** $\mathcal{C}$ responds by returning the partial private key d_A of the user U_A to $\mathcal{A}$.
- **Request Secret Value of ID_A:** If U_A's public key has not been replaced by $\mathcal{A}$ then $\mathcal{C}$ returns the user secret value y_A to $\mathcal{A}$. If U_A's public key was replaced by $\mathcal{A}$, then $\mathcal{C}$ returns nothing to $\mathcal{A}$.
- **Request Public Key of ID_A:** $\mathcal{C}$ responds by returning the current public key PK_A of user U_A to $\mathcal{A}$. (Because public keys are viable to change, $\mathcal{C}$ returns the current public key it has stored.)
- **Replace Public Key of ID_A:** The public key PK_A for a user U_A can be replaced with any value PK'_A provided by $\mathcal{A}$. On getting PK'_A from $\mathcal{A}$, $\mathcal{C}$ replaces the public key PK_A of ID_A with PK'_A. At any given time the current value of the user's public key is used by $\mathcal{C}$ in its computations or responses.
- **Signcryption of message m with ID_A as sender and ID_B as receiver:** $\mathcal{C}$ responds with the signcryption c on message m with ID_A as the sender and ID_B as the receiver. Note that even if $\mathcal{C}$ does not know the sender's private key, $\mathcal{C}$ should be able to produce a valid ciphertext and this is a strong property of the security model also $\mathcal{C}$ uses the current public keys of ID_A as well as ID_B to perform the signcryption.

– **Unsigncryption of ciphertext c with ID_A as sender and ID_B as receiver:** An unsigncryption query for ciphertext c and user U_A as the sender and U_B as the receiver is answered by $\mathcal{C}$, by first decrypting c and then returning the corresponding message m. $\mathcal{C}$ should be able to properly unsigncrypt ciphertexts, even for those users whose public keys have been replaced or if the receiver private key is not known to $\mathcal{C}$. This is a strong requirement of the security model. (Note that, $\mathcal{C}$ may not know the correct private key of the user whose public key is replaced. Still $\mathcal{C}$ can unsigncypt c by getting the corresponding secret value from $\mathcal{A}$.)

For any certificateless signcryption scheme two types of attacks are possible. They are referred as Type-I and Type-II attacks in the literature. Under each type of attack, it is required to establish the confidentiality and unforgeability of the scheme. The attack by a third party, (i.e. anyone except the legitimate receiver or the KGC) who is trying to break the security of the system is modeled by Type-I attack.

Summary of Constraints: In summary, the security model distinguishes the two types of adversaries (resp. forgers), namely Type-I and Type-II with the following constraints.

– Type-I adversary $\mathcal{A}_I$ (resp. forger $\mathcal{F}_I$) is allowed to replace the public keys of users at will but does not have access to the master private key msk.
– Type-II adversary $\mathcal{A}_{II}$ (resp. forger $\mathcal{F}_{II}$) is equipped with the master private key msk but is not allowed to replace public keys of any of the users.

Confidentiality: The security model to prove the confidentiality of a CLSC scheme with respect to Type-I adversary $\mathcal{A}_I$ (IND-CLSC-CCA2-I) and Type-II adversary $\mathcal{A}_{II}$ (IND-CLSC-CCA2-II) are given below:

IND-CLSC-CCA2-I game for Type-I Adversary: A certificateless signcryption (CLSC) scheme is IND-CLSC-CCA2-I secure if no probabilistic polynomial time adversary $\mathcal{A}_I$ has non-negligible advantage in winning the IND-CLSC-CCA2-I game. $\mathcal{A}_I$ is given access to all the six oracles defined above. It is to be noted that $\mathcal{A}_I$ does not have access to the master private key msk. IND-CLSC-CCA2-I game played between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}_I$ is defined below:

Setup: The challenger $\mathcal{C}$ runs the setup algorithm to generate the system public parameters $params$ and the master private key msk. $\mathcal{C}$ gives $params$ to $\mathcal{A}_I$ while keeping msk secret. $\mathcal{A}_I$ interacts with $\mathcal{C}$ in two phases:

Phase I: $\mathcal{A}_I$ is given access to all the six oracles described above. $\mathcal{A}_I$ adaptively queries (adaptively means the current query may depend on the responses to the previous queries) the oracles consistent with the conditions for Type-I adversary (Described in the ***Summary of Constraints*** above).

Challenge: $\mathcal{A}_I$ generates two messages m_0, m_1 of equal length, an arbitrary sender identity ID_A and a receiver identity ID_B, which satisfies the following constraints.

- $\mathcal{A}_I$ can access the full private key of the sender ID_A.
- $\mathcal{A}_I$ has not queried the **Partial Private Key** corresponding to the receiver ID_B.

$\mathcal{A}_I$ sends m_0, m_1, ID_A and ID_B to $\mathcal{C}$. $\mathcal{C}$ randomly chooses a bit $b \in_R \{0,1\}$ and computes a signcryption c^* with ID_A as the sender and ID_B as the receiver. Now, c^* is sent to $\mathcal{A}_I$ as the challenge signcryption.

Phase II: $\mathcal{A}_I$ adaptively queries the oracles consistent with the constraints that $\mathcal{A}_I$ should not query the partial private key of ID_B and $\mathcal{A}_I$ should not query for the *Unsigncryption* on c^* with ID_A as sender and ID_B as receiver.

Guess: $\mathcal{A}_I$ outputs a bit b' at the end of the game. $\mathcal{A}_I$ wins the IND-CLSC-CCA2-I game if $b' = b$. The advantage of $\mathcal{A}_I$ is defined as:

$$Adv_{\mathcal{A}_I}^{IND-CLSC-CCA2-I} = |2Pr\left[b = b'\right] - 1|$$

IND-CLSC-CCA2-II game for Type-II Adversary: A certificateless signcryption scheme (CLSC) is IND-CLSC-CCA2-II secure if no probabilistic polynomial time adversary $\mathcal{A}_{II}$ has non-negligible advantage in winning the IND-CLSC-CCA2-II game. $\mathcal{A}_{II}$ is given access to all the six oracles. The IND-CLSC-CCA2-II game played between $\mathcal{C}$ and the adversary $\mathcal{A}_{II}$ is defined below:

Setup: The challenger $\mathcal{C}$ runs the setup algorithm to generate the system public parameters *params* and the master private key *msk*. $\mathcal{C}$ gives both *params* and *msk* to $\mathcal{A}_{II}$. $\mathcal{C}$ interacts with $\mathcal{A}_{II}$ in two phases:

Phase I: This phase is similar to Type-I confidentiality game IND-CLSC-CCA2-I.
Challenge: Same as Type-I but with the restrictions that:

1. $\mathcal{A}_{II}$ should not have queried the private key of the receiver ID_B in Phase I.
2. $\mathcal{A}_{II}$ has not replaced public key of ID_B in Phase I.

Phase II: Same as Type-I but with the restrictions that,

- $\mathcal{A}_{II}$ cannot extract the private key of ID_B.
- $\mathcal{A}_{II}$ should not replace the receiver ID_B's public key.
- Unsigncryption query on $\langle c^*, ID_A, ID_B \rangle$ is not allowed.

Guess: Same as Type-I confidentiality game IND-CLSC-CCA2-I.
The advantage of $\mathcal{A}_{II}$ is defined as:

$$Adv_{\mathcal{A}_{II}}^{IND-CLSC-CCA2-II} = |2Pr\left[b = b'\right] - 1|$$

Unforgeability: The security model to prove the unforgeability of a CLSC scheme with respect to Type-I forger $\mathcal{F}_I$ (EUF-CLSC-CMA-I) and Type-II forger $\mathcal{F}_{II}$ (EUF-CLSC-CMA-II) are given below:

EUF-CLSC-CMA-I game for Type-I Forger: A certificateless signcryption scheme CLSC is Type-I, EUF-CLSC-CMA secure if no probabilistic polynomial-time forger $\mathcal{F}_I$ has non-negligible advantage in winning the EUF-CLSC-CMA-I game. A Type-I forger $\mathcal{F}_I$ is given access to all the six oracles defined above. The EUF-CLSC-CMA-I game played between the challenger $\mathcal{C}$ and the forger $\mathcal{F}_I$ is defined below:

Setup: $\mathcal{C}$ runs the setup algorithm to generate the master private key msk and public parameters $params$. $\mathcal{C}$ gives $params$ to $\mathcal{F}_I$ while keeping msk secret.

Training Phase: $\mathcal{F}_I$ is given access to all the six oracles. $\mathcal{F}_I$ adaptively queries the oracles consistent with the constraints for Type-I forger (Stated in the ***Summary of Constraints***).

Forgery: $\mathcal{F}_I$ outputs a signcryption c^* and a sender identity ID_A, for which $\mathcal{F}_I$ has not queried the partial private key. $\mathcal{F}_I$ wins the EUF-CLSC-CMA-I game if c^* is a valid signcryption with ID_A as the sender and ID_B as the receiver, also c^* was not the output of any signcrypt query on the corresponding message m with ID_A as the sender and ID_B as the receiver.

EUF-CLSC-CMA-II game for Type-II Forger: A certificateless signcryption scheme is Type-II, EUF-CLSC-CMA secure if no probabilistic polynomial-time forger $\mathcal{F}_{II}$ has non-negligible advantage in winning the EUF-CLSC-CMA-II game. A Type-II forger is given access to all the six oracles. EUF-CLSC-CMA-II game played between the challenger $\mathcal{C}$ and the forger $\mathcal{F}_{II}$ is same as EUF-CLSC-CMA-I with the constraints for Type-II Forger (Stated in the ***Summary of Constraints***).

3 Certificateless Signcryption Scheme of Barbosa et al.

In this section, we give the review and attack of the certificateless signcryption scheme by Barbosa et al. [2].

3.1 Review of Barbosa et al. Certificateless Signcryption Scheme

This scheme uses a symmetric bilinear group description Γ which is defined with two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of same order q, a random generator $P \in \mathbb{G}_1$ and an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. The four cryptographic hash functions used in the scheme are : $H_1 : \{0,1\}^* \rightarrow G_1$, $H_2 : \{0,1\}^* \rightarrow \{0,1\}^n$, $H_3 : \{0,1\}^* \rightarrow G_1$, $H_4 : \{0,1\}^* \rightarrow G_1$. Here, n is the maximum number of bits in a message. The master secret key s is selected uniformly at random from Z_p, and the master public key $P_{pub} = sP$. The public parameters of the system are $params$= $\langle \Gamma, P, P_{pub}, q, n, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, H_1, H_2, H_3, H_4 \rangle$.

The partial private key extraction algorithm on input (ID, s) returns $D = sH_1(ID) = sQ$. The user key generation algorithm returns a random element $x \in Z_p$ as the secret value, and $PK = xP$ as the public key of user with identity ID. The full private key of user with identity ID is $S = (x, D)$. Message, ciphertext and randomness spaces are $\{0,1\}^\kappa$, $G_1 \times \{0,1\}^\kappa \times G_1$ and Z_p respectively.

Signcrypt$(m, S_S = (x_S, D_S), ID_S, PK_S, ID_R, PK_R, P_{pub})$

- Choose $r \in Z_p$, compute $U = rP$, $T = \hat{e}(P_{pub}, Q_R)^r$, $h = H_2(U, T, rPK_R, ID_R, PK_R)$, $V = m \oplus h$, $H = H_3(U, V, ID_S, PK_S)$ and $H' = H_4(U, V, ID_S, PK_S)$.
- Compute $W = D_S + rH + x_S H'$ and set $c = \langle U, V, W \rangle$
- Return the signcryption c of message m from ID_S to ID_R.

Unsigncrypt$(c, S_R = (x_R, D_R), ID_R, PK_R, ID_S, PK_S, P_{pub})$

- The ciphertext c is of the form $\langle U, V, W \rangle$.
- Compute $H = H_3(U, V, ID_S, PK_S)$ and $H' = H_4(U, V, ID_S, PK_S)$
- If the check $e(P_{pub}, Q_S)e(U, H)e(PK_S, H') \overset{?}{=} e(P, W)$ fails, return $''Invalid''$.
- Compute $T = \hat{e}(D_R, U)$, $h = H_2(U, T, x_R U, ID_R, PK_R)$ and retrieve $m = V \oplus h$.
- Return the message m.

Note: The certificateless signcryption scheme uses an Encrypt-then-Sign approach. A common randomness is shared between the signature and encryption components in the scheme to bind them together.

3.2 Attack on Barbosa et al. Certificateless Signcryption Scheme

The scheme proposed by Barbosa et al. in [2] is existentially forgeable. The scheme uses the Encrypt-then-Sign approach with public verifiability of ciphertext. The intuition behind the attack: for any signcryption scheme following the Encrypt-then-Sign approach, the identity of the sender should be bound to the encryption and the identity of the receiver should be bound to the signature. In [2], the authors have achieved this binding by using a common randomness for encryption and signature independently but they failed to bind the receiver to the signature. This led to the attack on existential unforgeability of [2]. The attack is shown below.

- During the unforgeability game (Both Type-I and type-II), the forger requests a signcryption on a message m from ID_S^* to a arbitrary user with identity ID_A.
- Let the signcryption of m from ID_S^* to ID_A be $c = (U, V, W)$.
- Now, the forger submits $c^* = (U, V, W)$ as a signcryption from user ID_S^* to ID_R^*, where ID_S^* is the target sender identity for which the forger is not allowed to know the private key (partial private key for Type-I and user private key for Type-II forgers respectively) and ID_R^* is the new receiver identity. Note that c^* is a valid signcryption of some random message $m^* = m \oplus h \oplus h^*$ where $h^* = H_2(U, T^*, x_R^* U, ID_R^*, PK_R^*)$ and $T^* = \hat{e}(D_R^*, U)$. Here $h = H_2(U, T, x_A U, ID_A, PK_A)$ is the key used for encrypting the message m from ID_S^* to ID_A during signcryption.
- The signature W will pass the verification because none of the components of the H and H' are altered. The correctness of the signcryption is straight forward as follows.

$$e(P_{pub}, Q_S)e(U, H)e(PK_S, H') = e(P, W)$$

where $H = H_3(U, V, ID_S, PK_S)$ and $H' = H_4(U, V, ID_S, PK_S)$

So the challenger will accept c^* as a valid forgery on message $m^* = m \oplus h \oplus h^*$.

4 Certificateless Signcryption Scheme of Diego et al.

In this section, we give the review and attack of the certificateless signcryption scheme by Diego et al. [1].

4.1 Overview of the Scheme

Diego et al.'s CLSC scheme [1] consists of five algorithms namely: *Setup*, *Extract*, *Keygen*, *Signcrypt* and *Unsigncrypt*, which we describe below.

- ***Setup.*** Let κ be the security parameter. The KGC performs the following to set up the system.
  - The KGC selects cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of same order q with generators $P \in_R \mathbb{G}_1$ and $Q \in_R \mathbb{G}_2$.
  - Selects the master secret key $s \in_R \mathbb{Z}_q^*$ and the master public key is set to be $P_{pub} = sP$.
  - Selects an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.
  - Computes $g = \hat{e}(P, Q)$.
  - Selects three hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*, H_2 : \mathbb{G}_T \rightarrow \{0,1\}^n, H_3 : \{0,1\}^n \times \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$, Here n is the length of the message.
  - The public parameters of the scheme are set to be $params = \langle q$, $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, $\hat{e}$, g, P, Q, P_{pub}, H_1, H_2, $H_3\rangle$.
- ***Extract.*** Here, ID_A is the identity of the user U_A, the KGC computes the partial private key of user U_A as follows.
  - Computes the hash value $y_A = H_1(ID_A)$ and the partial private key $D_A = (y_A + s)^{-1}Q \in \mathbb{G}_2$.
  - The KGC sends D_A to the user U_A via a secure authenticated channel.
- ***Keygen.*** User U_A computes the full private key by performing the following steps:
  - Chooses $x_A \in_R \mathbb{Z}_q^*$ as the secret value.
  - Computes the full private key $S_A = x_A^{-1} D_A \in \mathbb{G}_2$.
  - Computes the public key as $P_A = x_A(y_A P + P_{pub}) \in \mathbb{G}_1$.
  - It is to be noted that $\hat{e}(P_A, S_A) = g$.
- ***Signcrypt.*** Inorder to signcrypt the message m to the receiver U_B, the sender U_A does the following:
  - Chooses $r \in_R \mathbb{Z}_q^*$, computes $u = r^{-1}$ and $U = g^u$.
  - Computes $c = m \oplus H_2(U)$, $R = rP_A$, $S = uP_B$, $h = H_3(c, R, S)$ and $T = (r + h)^{-1} S_A$.

  Finally, the sender outputs the signcryption on message m as $\sigma = \langle c, R, S, T\rangle$.
- ***Unsigncrypt.*** Inorder to unsigncrypt a ciphertext σ, the receiver U_B does the following:

- Computes $h' = H_3(c, R, S)$ and $U' = \hat{e}(S, S_B)$.
- Recovers the message as $m' = c \oplus H_2(U')$.
- Checks whether $\hat{e}(R + h'P_A, T) \overset{?}{=} g$.

If the check holds, then accepts m' as the message, otherwise outputs "*Invalid*".

4.2 Attack on the CLSC Scheme by Diego et al.

Type-I Forgeability: The Type-I adversary who is capable of replacing the public keys of all users and is not allowed to know the master private key can forge a valid signcryption on any message m, from any legitimate user U_A to U_B by performing the following:

- Let ID_A be the identity of user U_A.
- The adversary chooses $r \in_R \mathbb{Z}_q^*$, computes $u = r^{-1}$.
- Computes $U = g^u$ and sets $c = m \oplus H_2(U)$.
- Set $T = uQ$, $R = rP - P$ and $S = uP_B$.
- Compute $h = H_3(c, R, S)$.
- Set $P_A = h^{-1}P$.

Finally, the forger outputs the signcryption on message m as $\sigma = (c, R, S, T)$ which is a valid signcryption on m from U_A to U_B.

Correctness: The signcryption σ, which is produced as forgery passes the verification test as shown below,

$$\hat{e}(R + hP_A, T) = \hat{e}(rP - P + hh^{-1}P, uQ) = \hat{e}(rP, r^{-1}Q)$$
$$\text{f}= \hat{e}(P, Q) = g$$

This proves that the forgery generated is valid.

Type-I and Type-II Attacks on Confidentiality

- Let $\sigma^* = (c^*, R^*, S^*, T^*)$ be the challenge signcryption on message m_b, $b \in \{0, 1\}$ with ID_A as the sender and ID_B as the receiver.
- The adversary is capable of generating a new signcryption σ' on the message m_b (The message is same as in σ^*) with ID_C as sender and ID_B as receiver (Note that the adversary knows the private key of ID_C).
- σ' is obtained by the adversary by performing the following:
  - Sets $c' = c^*$.
  - Computes $R' = r'P_C$, where $r' \in_R \mathbb{Z}_q^*$.
  - Set $S' = S^*$.
  - Computes $h' = H_3(c', R', S')$
  - Set $T' = (r' + h')^{-1}S_C$
  - The signcryption corresponding to this change is $\sigma' = \langle c', R', S', T' \rangle$.
- Now, the adversary can query the unsigncryption oracle for the unsigncryption of σ' (Note that this query is valid because σ' is different from the challenge signcryption σ^*).

– The unsigncryption oracle will give back the message m_b since the key used in both σ^* and σ' are the same i.e., $U' = \hat{e}(S', S_B) = \hat{e}(S^*, S_B) = U^*$ and note that $S' = S^*$. Hence , $c' \oplus H_2(U') = c^* \oplus H_2(U^*) = m_b$.
– Therefore, designcryption of σ' outputs the message m_b, which is used for generating the challenge ciphertext σ^*. Thus the adversary can determine whether $m_b \overset{?}{=} (m_0 \text{ or } m_1)$ breaking the indistinguishability of the scheme. This attack can be performed by both Type-I and Type-II adversaries because the adversary does not require the master private key or even does not want to replace the public key.

5 Certificateless Signcryption Scheme of Chen-Huang et al.

In this section, we present the review and attack of the certificateless signcryption scheme by Chen-Huang et al. [6].

5.1 Overview of the Scheme

The CLSC scheme of Chen-Huang et al. [6] consists of the following four algorithms.

– ***Setup.*** Given κ as the security parameter, the KGC does the following to setup the system parameters.
  - The KGC selects $\mathbb{G}_1$, $\mathbb{G}_2$ of same prime order q with a generator $P \in_R \mathbb{G}_1$.
  - Selects the master secret key $s \in_R \mathbb{Z}_q^*$ and the master public key is set to be $P_{pub} = sP$.
  - Selects an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
  - Selects three cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1, H_2 : \{0,1\}^* \to \mathbb{Z}_q^*, H_3 : \{0,1\}^* \to \{0,1\}^n$, where n is the size of the message.
  - Computes $T = \hat{e}(P, P)$.
  - The public parameters of the scheme are set to be $params = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, T, H_1, H_2, H_3 \rangle$.

– ***Keygen.*** Let, ID_A be the identity of the user U_A. The KGC computes the partial private key of user U_A as follows.
  - Computes $Q_A = H_1(ID_A)$ and the partial private key $D_A = sQ_A \in \mathbb{G}_2$.
  - The KGC sends D_A to the user U_i via a secure authenticated channel.

  On receiving the partial private key D_A, user U_A computes his full private key by performing the following steps:
  - U_A chooses $x_A \in_R \mathbb{Z}_q^*$ as the secret value.
  - Sets the full private key $S_A = \langle x_A, D_A \rangle$.
  - The corresponding public key is $P_A = T^{x_A} \in \mathbb{G}_2$.

– ***Signcrypt.*** Inorder to signcrypt the message m of length n to the receiver U_B, the sender U_A does the following:

- Chooses $r, r_1, r_2 \in_R \mathbb{Z}_q^*$, computes $R_1 = T^{r_1}$, $R_2 = T^{r_2}$, $h = H_2(m\| R_1\|R_2\|P_A\|P_B)$, $U = r_1P - hS_A$, $u = r_2 - x_Ah$, $K = \hat{e}(S_A, Q_B)^r T_B^{x_A}$, $W = rQ_A$and computes $c = H_3(K) \oplus m$

Finally, the sender outputs the signcryption on message m as$\sigma = (c, u, h, U, W)$.

– ***Unsigncrypt.*** Inorder to unsigncrypt a ciphertext σ, the receiver U_B does the following:
  - Computes $K' = \hat{e}(S_B, W)T_A^{x_B}$.
  - Retrieves the message as $m' = c \oplus H_3(K')$.
  - Checks whether $h \overset{?}{=} H_2(m'\|\hat{e}(U, P)\hat{e}(Q_A, P_{pub})^h)\|T^u P_A^h\|P_A\|P_B)$.

  If the check holds, then accepts m' as the message, otherwise outputs "*Invalid*".

5.2 Attack on the CLSC Scheme by Chen-Huang et al.

In this section, we show that the certificateless signcryption scheme by Chen-Huang et al. does not provide confidentiality as well as unforgeability with respect to both Type-I and Type-II attacks.

Attack on Type-I and Type-II Confidentiality: The following attack is possible because the adversary is capable of altering the challenge signcryption without altering the message in it and is allowed to obtain the unsigncryption of the newly formed signcryption, which yields the message signcrypted in σ^*. We explain the attack in detail now. On getting the challenge signcryption $\sigma^* = \langle c^*, u^*, h^*, U^*, W^*\rangle$, ($\sigma^*$ is the signcryption of either message m_0 or m_1 from user U_A to U_B) the adversary (Type-I and Type-II) is capable of generating a new ciphertext $\sigma' = \langle c', u', h', U', W'\rangle$ (signcryption of m_0 from user U_C to U_B) as follows:

– Replace the public key of user U_C with the public key of user U_A.
– Sets $c' = c^*$ and $W' = W^*$.
– Chooses $r_1, r_2 \in_R \mathbb{Z}_q^*$, computes $R_1 = T^{r_1}$ and $R_2 = T^{r_2}$.
– Computes $h' = H_2(m_0\|R_1\|R_2\|P_C\|P_B)$.
– Computes $U' = r_1P - h'S_C$ and $u = r_2 - x_Ch'$.
– Gets the unsigncryption of σ'.
– If *Unsigncrypt*(σ') = "m_0" then the adversary outputs that σ^* is the signcryption of m_0 (i.e. $b' = 0$).
– If *Unsigncrypt*(σ') = "*Invalid*" then the adversary outputs m_1 (i.e. $b' = 1$).

Note: This attack can be done by both Type-I and Type-II adversaries.

6 Certificateless Signcryption without Pairing

In this section, we present our new certificateless signcryption scheme which is efficient and does not use the costly bilinear pairing operation.

– **CLSC.Setup(1^κ):** The KGC takes the security parameter 1^κ as input and performs the following for setting up the system:

  - Chooses two big prime numbers p and q such that $q|(p-1)$.
  - Selects an element $g \in_R \mathbb{Z}_p^*$ with order q.
  - Chooses a master private key $s \in_R \mathbb{Z}_q^*$ and computes the master public key $g_{pub}{=}g^s$.
  - Chooses five cryptographic hash functions $H_1 : \{0,1\}^* \times \mathbb{Z}_p^* \rightarrow \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \times \mathbb{Z}_p^* \times \mathbb{Z}_p^* \rightarrow \mathbb{Z}_q^*$, $H_3 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_4 : \{0,1\}^* \rightarrow |\mathcal{M}| \times \mathbb{Z}_q^* \times \mathbb{Z}_q^*$, $H_5 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, here $\mathcal{M}$ is the message space.

  The public parameters of the system, $params = \langle p, q, g, g_{pub}, H_1, H_2, H_3, H_4, H_5 \rangle$.

- **CLSC.PartialPrivateKeyExtract:** Given an identity, say ID_A of a user U_A, the KGC performs the following to generate the partial private key corresponding to ID_A:
  - Chooses $x_{A0}, x_{A1} \in_R \mathbb{Z}_q^*$.
  - Computes $X_{A0} = g^{x_{A0}}$ and $X_{A1} = g^{x_{A1}}$.
  - Computes $q_{A0} = H_1(ID_A, X_{A0})$ and $q_{A1} = H_2(ID_A, X_{A0}, X_{A1})$.
  - Computes $d_{A0} = x_{A0} + sq_{A0}$ and $d_{A1} = x_{A1} + sq_{A1}$.

  Returns $d_A = \langle d_{A0}, d_{A1} \rangle$ and $X_A = \langle X_{A0}, X_{A1} \rangle$, the partial private keys securely to user U_A.

  Note: It should be noted that the partial private key of a user is a Schnorr signature on the user's identity, signed by the KGC using the master private key.

- **CLSC.SetSecretValue:** The user U_A chooses an element $y_A \in_R \mathbb{Z}_q^*$ and keeps it as his secret value.
- **CLSC.SetPrivateKey:** The user U_A sets his full private key $s_A{=}\langle y_A, d_{A0} \rangle$.
- **CLSC.SetPublicKey:** The user U_A computes $Y_A = g^{y_A}$ and sets his public key as $PK_A = \langle d_{A1},\ X_{A0},\ X_{A1}, Y_A \rangle$. The resulting public key is distributed widely and freely.
- **CLSC.Signcrypt:** The sender U_A signcrypts a message m to a receiver U_B by performing the following:
  - Chooses $r_1, r_2 \in_R \mathbb{Z}_q^*$, computes $c_1 = g^{r_1}$, $c_2 = g^{r_2}$, $k_1 = (Y_B)^{r_1}$, $k_2 = (X_{B0}(g_{pub})^{q_{B0}})^{r_1}$, $d = H_3(m, c_2, ID_A, ID_B, PK_A)$, $e = H_5(m, c_2, ID_A, ID_B, PK_A)$, $v = (d.d_{A0} + e.y_A) + r_2$ and $c_3 = H_4(k_1, k_2, ID_A, ID_B) \oplus (m \| r_1 \| v)$.

  Now $c = \langle c_1, c_2, c_3 \rangle$ is the signcryption on message m to user U_B

- **CLSC.Unsigncrypt:** To unsigncrypt a signcryption $c = \langle c_1, c_2, c_3 \rangle$ from sender U_A, the receiver U_B does the following:
  - Computes $k_1'{=}(c_1)^{y_B}$, $k_2'{=}(c_1)^{d_{B0}}$ and $(m' \| r_1' \| v'){=}c_3 \oplus H_4(k_1', k_2', ID_A, ID_B)$.
  - Checks whether $g^{r_1'} \stackrel{?}{=} c_1$.
  - If so computes $d'{=}H_3(m', c_2, ID_A, ID_B, PK_A)$ and $e'{=}H_5(m', c_2, ID_A, ID_B, PK_A)$.
  - Checks whether $g^{v'} \stackrel{?}{=} ((g_{pub})^{q_{A0}}.X_{A0})^{d'}.(Y_A)^{e'}.c_2$.

  If both the checks hold, m' is output as the unsigncrypted message else outputs "*Invalid*".

Correctness: The correctness of the verification test $g^{r'_1} \overset{?}{=} c_1$ is straight forward. The second check also passes the verification if the signcryption is formed in a legitimate way which is shown below.

$$g^{v'} = g^{(d_{A0}d'+y_A e')+r_2} = g^{(x_{A0}d'+sq_{A0}d'+y_A e')+r_2} = g^{x_{A0}d'}.g^{sq_{A0}d'}.g^{y_A e'}.g^{r_2}$$
$$= ((g_{pub})^{q_{A0}}.X_{A0})^{d'}.(Y_A)^{e'}.c_2$$

7 Security of CLSC Scheme

In this section, we provide the formal proof for the unforgeability and confidentiality of the CLSC scheme.

7.1 Type-I Unforgeability

Theorem 1. *The certificateless signcryption scheme CLSC is secure against any EUF-CLSC-CMA-I forger $\mathcal{F}_I$ in the random oracle model if DLP is hard in $\mathbb{Z}_P^*$.*

Proof: A challenger $\mathcal{C}$ is challenged with an instance of the DLP, say $\langle g, g^a \rangle$, the aim of $\mathcal{C}$ is to find the value a. Let $\mathcal{F}_I$ be a forger who is capable of breaking the EUF-CLSC-CMA-I security of the CLSC scheme. $\mathcal{C}$ can make use of $\mathcal{F}_I$ to compute the solution of the DLP instance by playing the following interactive game with $\mathcal{F}_I$.

Setup: $\mathcal{C}$ sets the master public key g_{pub} as g^s by choosing $s \in_R \mathbb{Z}_q^*$, designs the hash functions H_i (i =1 *to* 5) as random oracles $\mathcal{O}_{H_i}$ (i =1 *to* 5) respectively. In order to maintain the consistency between the responses to the hash queries, $\mathcal{C}$ maintains lists

- L_1 with tuples of the form $\langle ID_A, q_{A0}, q_{A1}, X_{A0}, X_{A1}, d_{A0}, d_{A1}, y_A, Y_A \rangle$.
- L_{H_1} with tuples of the form $\langle ID_A, X_{A0}, q_{A0} \rangle$,
- L_{H_2} having tuples of the form $\langle ID_A, X_{A0}, X_{A1}, q_{A1} \rangle$,
- L_{H_3} with tuples of the form $\langle m, c_2, ID_A, ID_B, PK_A, d \rangle$,
- L_{H_4} which has tuples of the form $\langle k_1, k_2, ID_A, ID_B, h_4 \rangle$ and
- L_{H_5} with tuples of the form $\langle m, c_2, ID_A, ID_B, PK_A, e \rangle$.

$\mathcal{C}$ gives the public parameters *params* to $\mathcal{F}_I$.

Training Phase: $\mathcal{F}_I$ performs a series of polynomially bounded number of queries in an adaptive fashion during the **Training phase**. Through out the game we assume that $\mathcal{F}_I$ makes the $\mathcal{O}_{RequestPublicKey}$ queries on ID before querying other oracles with ID as input. The description of the various oracles and the query responses by each oracles are described below.

$\mathcal{O}_{H_1}(ID_A, X_{A0})$: To respond to this query, $\mathcal{C}$ checks whether a tuple of the form $\langle ID_A, X_{A0}, q_{A0} \rangle$ exists in the list L_{H_1}. If a matching tuple exists in L_{H_1}, $\mathcal{C}$ returns q_{A0} to $\mathcal{F}_I$. Otherwise, $\mathcal{C}$ chooses randomly $q_{A0} \in \mathbb{Z}_q^*$, appends the tuple $\langle ID_A, X_{A0}, q_{A0} \rangle$ to L_{H_1} and returns q_{A0} to $\mathcal{F}_I$.

$\mathcal{O}_{H_2}(ID_A, X_{A0}, X_{A1})$: For responding to this query, $\mathcal{C}$ checks whether a tuple of the form $\langle ID_A,\ X_{A0}, X_{A1}, q_{A1} \rangle$ already exists in the list L_{H_2}. If so, returns q_{A1}

to $\mathcal{F}_I$, else chooses $q_{A1} \in_R \mathbb{Z}_q^*$, adds $\langle ID_A, X_{A0}, X_{A1}, q_{A1}\rangle$ to the list L_{H_2} and returns q_{A1} to $\mathcal{F}_I$.

$\mathcal{O}_{H_3}(m, c_2, ID_A, ID_B, PK_A)$: To respond to this query, $\mathcal{C}$ checks whether a tuple of the form $\langle m, c_2, ID_A, ID_B, PK_A, d\rangle$ exists in the list L_{H_3}. If so, returns d to $\mathcal{F}_I$ else chooses $d \in_R \mathbb{Z}_q^*$, adds the tuple $\langle m, c_2, ID_A, ID_B, PK_A, d\rangle$ to the list L_{H_3} and returns d to $\mathcal{F}_I$.

$\mathcal{O}_{H_4}(k_1, k_2, ID_A, ID_B)$: To respond to this query, $\mathcal{C}$ checks whether a tuple of the form $\langle k_1, k_2, ID_A, ID_B, h_4\rangle$ exists in the list L_{H_4}. If so, returns h_4 to $\mathcal{F}_I$ else, chooses $h_4 \in_R \mathbb{Z}_q^*$, adds the tuple $\langle k_1, k_2, ID_A, ID_B, h_4\rangle$ to the list L_{H_4} and returns h_4 to $\mathcal{F}_I$.

$\mathcal{O}_{H_5}(m, c_2, ID_A, ID_B, PK_A)$: To respond to this query, $\mathcal{C}$ checks whether a tuple of the form $\langle m, c_2, ID_A, ID_B, PK_A, e\rangle$ exists in the list L_{H_5}. If so, returns e to $\mathcal{F}_I$ else chooses $e \in_R \mathbb{Z}_q^*$, adds the tuple $\langle m, c_2, ID_A, ID_B, PK_A, e\rangle$ to the list L_{H_5} and returns e to $\mathcal{F}_I$.

$\mathcal{O}_{RequestPublicKey}(ID_A)$*:* During the forgery phase, the model requires the forger $\mathcal{F}_I$ and the challenger $\mathcal{C}$ to work with a signcryption from user U_A to U_B, where U_A and U_B satisfy the following conditions:

– The hash of the identity ID_A of the sender U_A should have been queried by $\mathcal{F}_I$ to $\mathcal{O}_{H_1}$ and $\mathcal{O}_{H_2}$.
– At the same time, during the **Training phase** $\mathcal{F}_I$ should not know which of the identity that $\mathcal{F}_I$ has queried is selected by $\mathcal{C}$ for the forgery phase.

In order to achieve the selection of identities satisfying the above conditions, we specify the Request Public Key oracle as follows:

$\mathcal{F}_I$ will generate a series of users and send each of them to $\mathcal{C}$ and ask their corresponding public key. One of them will be chosen by $\mathcal{C}$, say the γ^{th} distinct user, as target identity for the forgery phase. However, $\mathcal{C}$ will not reveal the value of γ or ID_γ to $\mathcal{F}_I$. Moreover, $\mathcal{C}$ will choose γ randomly for each game. Thus, while the target identity is one of the identities chosen by $\mathcal{F}_I$, $\mathcal{F}_I$ will not know which one is that. From now on, ID_γ denotes the specific target identity selected by $\mathcal{C}$ during the **Training phase**.

The oracle description and the responses by $\mathcal{C}$ are described now. $\mathcal{F}_I$ produces an identity ID_A to $\mathcal{C}$ and requests ID_A's public key. The response of $\mathcal{C}$ depends upon whether the query is γ^{th} query or not:

Case 1:

– If the query is not the γ^{th} query then $\mathcal{C}$ proceeds as follows:
– $\mathcal{C}$ checks whether a tuple of the form $\langle ID_A, q_{A0}, q_{A1}, X_{A0}, X_{A1}, d_{A0}, d_{A1}, y_A, Y_A\rangle$ already exists in the list L_1. If a tuple appears in the list L_1, $\mathcal{C}$ retrieves $(X_{A0}, X_{A1}, d_{A1}, Y_A)$ from the tuple corresponding to ID_A and returns them as the public key of ID_A to $\mathcal{F}_I$.
– If no matching tuple exists for ID_A, $\mathcal{C}$ responds as per the actual algorithm, generates the public key corresponding to ID_A and sends $(X_{A0}, X_{A1}, d_{A1}, Y_A)$ to $\mathcal{F}_I$.

Case 2:

- If the query is the γ^{th} query and a tuple of the form $\langle ID_A, q_{A0}, q_{A1}, X_{A0}, X_{A1}, \perp, d_{A1}, y_A, Y_A\rangle$, corresponding to ID_A exists in list L_1, $\mathcal{C}$ retrieves $(X_{A0}, X_{A1}, d_{A1}, Y_A)$ from the tuple and returns them as the public key of ID_A to $\mathcal{F}_I$.
- If the query is the γ^{th} query and no tuple corresponding to ID_A exists in list L_1, $\mathcal{C}$ secretly sets $ID_\gamma = ID_A$ (The target identity) and proceeds as follows:
  - Randomly chooses $x_{A1}, y_A \in \mathbb{Z}_q^*$, computes $X_{A0} = g^a$, $X_{A1} = g^{x_{A1}}$ and $Y_A = g^{y_A}$.
  - Chooses $q_{A0} \in_R \mathbb{Z}_q^*$, adds the tuple $\langle ID_A, X_{A0}, q_{A0}\rangle$ in the list L_{H_1}.
  - Chooses $q_{A1} \in_R \mathbb{Z}_q^*$, adds the tuple $\langle ID_A, X_{A0}, X_{A1}, q_{A1}\rangle$ in the list L_{H_2}.
  - Computes $d_{A1} = x_{A1} + s\, q_{A1}$. (Here, s is the master private key and is known to $\mathcal{C}$).
  - Adds $\langle ID_A, q_{A0}, q_{A1}, X_{A0}, X_{A1}, \perp, d_{A1}, y_A, Y_A\rangle$ to the list L_1.

$\mathcal{O}_{ReplacePublicKey}(ID_A, X'_{A0}, X'_{A1}, d'_{A1}, Y'_A)$: $\mathcal{F}_I$ sends $ID_A, X'_{A0}, X'_{A1}, d'_{A1}$ and Y'_A to $\mathcal{C}$. $\mathcal{C}$ checks if $g^{d'_{A1}} = X'_{A0}(g^s)^{q_{A0}}$ (This is to check the validity of the public key). If the check holds then $\mathcal{C}$ replaces the existing tuple in the list L_1 with $\langle ID_A, q'_{A0}, q'_{A1}, X'_{A0}, X'_{A1}, \perp, d'_{A1}, \perp, Y'_A\rangle$ (Since X_{A0} is replaced with X'_{A0}, $\mathcal{C}$ does not know the corresponding partial private key d'_{A0}. Thus, $\mathcal{C}$ sets $d'_{A0} = \perp$ in the list L_1).

$\mathcal{O}_{ExtractPartialPrivateKey}(ID_A)$: $\mathcal{F}_I$ chooses an identity ID_A and gives it to $\mathcal{C}$. $\mathcal{C}$ performs the following to answer this query:

- If $ID_A = ID_\gamma$ then $\mathcal{C}$ *aborts*.
- If $ID_A \neq ID_\gamma$ and a tuple corresponding to ID_A was not available in list L_1, then $\mathcal{C}$ performs the following:
  - Chooses $x_{A0}, x_{A1}, y_A \in_R \mathbb{Z}_q^*$, computes $X_{A0} = g^{x_{A0}}$ and $X_{A1} = g^{x_{A1}}$.
  - $\mathcal{C}$ also chooses $q_{A0}, q_{A1} \in_R \mathbb{Z}_q^*$.
  - Computes $d_{A0} = x_{A0} + sq_{A0}$ and $d_{A1} = x_{A1} + sq_{A1}$.
  - Stores the tuple $\langle ID_A, q_{A0}, q_{A1}, X_{A0}, X_{A1}, d_{A0}, d_{A1}, y_A, Y_A\rangle$ and returns d_{A0} to $\mathcal{F}_I$.
- If $ID_A \neq ID_\gamma$ and a tuple corresponding to ID_A exists in L_1, $\mathcal{C}$ retrieves $(d_{A0}, d_{A1}, X_{A0}, X_{A1})$.
- $\mathcal{C}$ sends $\langle d_{A0}, d_{A1}, X_{A0}, X_{A1}\rangle$ to $\mathcal{F}_I$.

$\mathcal{O}_{ExtractSecretValue}(ID_A)$: $\mathcal{F}_I$ produces an identity ID_A and requests the corresponding user private key. $\mathcal{C}$ performs the following to answer this query:

- If a tuple of the form $\langle ID_A, q_{A0}, q_{A1}, X_{A0}, X_{A1}, d_{A0}, d_{A1}, y_A, Y_A\rangle$ or $\langle ID_A, q_{A0}, q_{A1}, X_{A0}, X_{A1}, \perp, d_{A1}, y_A, Y_A\rangle$ already exists in the list L_1 then $\mathcal{C}$ retrieves the corresponding y_A from the tuple and returns the user secret value y_A to $\mathcal{F}_I$.
- If an entry for y_A is not found in the corresponding tuple of the list L_1, then $\mathcal{C}$ chooses $y_A \in_R \mathbb{Z}_q^*$, updates the tuple with y_A, Y_A values, where $Y_A = g^{y_A}$ and returns the user secret value y_A to $\mathcal{F}_I$.

- If $\mathcal{F}_I$ has already replaced ID_A's public key, i.e. the value of Y_A is set and the corresponding y_A is $\perp$ in the tuple, then $\mathcal{C}$ does not provide the corresponding user secret value to $\mathcal{F}_I$, since it is replaced by $\mathcal{F}_I$ and $\mathcal{F}_I$ knows y_A.

Note: No separate private key extract oracle is provided in the model because the private key of a user is the combination of his partial private key and the secret value, which can be obtained from the two oracles $\mathcal{O}_{ExtractPartialPrivateKey}$ and $\mathcal{O}_{ExtractSecretValue}$.

$\mathcal{O}_{Signcrypt}(m, ID_A, ID_B)$*:* To respond to the signcrypt query on a message m with ID_A as the sender and ID_B as the receiver, $\mathcal{C}$ does the following:

- If $ID_A \neq ID_\gamma$ and $ID_A \neq ID_B$, $\mathcal{C}$ proceeds as per the actual signcryption algorithm because $\mathcal{C}$ knows the private key corresponding to the sender ID_A.
- If $ID_A = ID_\gamma$ and $ID_A \neq ID_B$ then $\mathcal{C}$ forms the signcryption in the following way:
  - Chooses $r_1, r_2, d, e \in_R \mathbb{Z}_q^*$, computes $c_1 = g^{(r_1)}$ and $c_2 = g^{(r_2)}.(g^{sq_{A0}} X_{A0})^{-d}.(Y_A)^{-e}$.
  - Computes $k_1 = Y_B^{(r_1)}$ and $k_2 = (X_{B0}.g_{pub}^{(q_{B0})})^{r_1}$
  - Stores the tuple $\langle m, c_2, ID_A, ID_B, PK_A, d\rangle$ in list L_{H_3} and the tuple $\langle m, c_2, ID_A, ID_B, PK_A, e\rangle$ in list L_{H_5}.
  - Sets $v = r_2$.
  - Chooses $h_4 \in_R (\mathcal{M} \times \mathbb{Z}_q^* \times \mathbb{Z}_q^*)$, computes $c_3 = h_4 \oplus (m\|r_1\|v)$ and stores the tuple $\langle k_1,\ k_2,\ ID_A,\ ID_B,\ h_4\rangle$ in the list L_{H_4}.
- Returns the signcryption $c = \langle c_1, c_2, c_3\rangle$ to $\mathcal{F}_I$.

Correctness: The validity of the signcryption generated with out knowing the private key of the sender ID_A can be verified with the equality $g^{v'} \stackrel{?}{=} (g_{pub}^{q_{A0}}.X_{A0})^d.Y_A^e.c_2$. We show that the signcryption formed is indeed a valid one through the following equality:

$$\begin{aligned}(g_{pub}^{q_{A0}}.X_{A0})^d.Y_A^e.c_2 &= (g_{pub}^{q_{A0}}.X_{A0})^d.Y_A^e.g^{(r_2)}.(g^{sq_{A0}}X_{A0})^{-d}.(Y_A)^{-e}\\ &= (g_{pub}^{q_{A0}}.X_{A0})^d.Y_A^e.g^{(r_2)}.(Y_A)^{-e}(g_{pub}^{q_{A0}}.X_{A0})^{-d} = g^{r_2} = g^{v'}\end{aligned}$$

$\mathcal{O}_{Unsigncrypt}(c, ID_A, ID_B)$*:* To respond to the unsigncrypt query of a signcryption $c = \langle c_1, c_2, c_3\rangle$ with ID_A as the sender and ID_B as the receiver, $\mathcal{C}$ does the following:

- If $ID_B \neq ID_\gamma$ and $ID_A \neq ID_B$, $\mathcal{C}$ proceeds as per the actual unsigncryption algorithm because $\mathcal{C}$ knows the private key corresponding to the receiver ID_B.
- If $ID_B = ID_\gamma$ and $ID_A \neq ID_B$ then $\mathcal{C}$ performs the unsigncryption in the following way:
  - Computes $k_1 = c_1^{y_B}$.
  - Checks all entries in the list L_{H_4} for the tuples of the form $\langle k_1, k_2, ID_A, ID_B, h_4\rangle$, that contains the computed k_1 value and with the current identities (i.e. k_1, ID_A, ID_B)

- Retrieves all corresponding k_2, h_4 values from the tuples obtained from the previous step.
- For all such pair $\langle k_2, h_4\rangle$, $\mathcal{C}$ performs the following:
  - $*$ Computes $(m' \| r'_1 \| v') = c_3 \oplus h_4$.
  - $*$ Retrieves the tuple $\langle m', c_2, ID_A, ID_B, PK_A, d\rangle$ from list L_{H_3} and the tuple $\langle m', c_2, ID_A, ID_B, PK_A, e\rangle$ from list L_{H_5}, checks whether $g^{v'} \overset{?}{=} g^{d_{A0}d+y_A e}.c_2$ and $c_1 \overset{?}{=} g^{r'_1}$.
  - $*$ If both the aforementioned checks hold, then returns m' as the message corresponding to c.
- If for no pair of $\langle k_2, h_4\rangle$ the verification passes then $\mathcal{C}$ returns "*Invalid*".

Note: If $y_A = \perp$ (which means the public key is replaced), $\mathcal{C}$ may obtain y_A from $\mathcal{F}_I$ and perform the above steps.

Remark: Note that the public key of both the sender and the receiver may be changed by $\mathcal{F}_I$ during the **Training Phase**. The unsigncryption will work correctly only with the public key used at the time of generating the signcryption and if the public keys are replaced after signcryption, then the unsigncryption will return "*Invalid*"

Forgery: At the end of the *Training Phase* (which is decided by $\mathcal{F}_I$), $\mathcal{F}_I$ sends to $\mathcal{C}$ a forged signcryption $c^* = \langle c_1^*, c_2^*, c_3^*\rangle$, where ID_A is the sender identity and ID_B is the receiver identity. It is to be noted that the partial private key of the sender ID_A should not have been queried by $\mathcal{F}_I$ during the **Training Phase**. In addition, c^* should not be the response for any signcryption queries by $\mathcal{F}_I$ during the **Training Phase**. If c^* is generated with these restrictions, then $\mathcal{C}$ can obtain the solution for the DLP instance by performing the following steps.

- Checks if $ID_A = ID_\gamma$, if so $\mathcal{C}$ performs the following:
  - Retrieves $(m^* \| r_i^* \| v^*) = c_3 \oplus h_4$. (Since $\mathcal{C}$ knows the private key of A)
  - If v^* passes the verification during unsigncryption, $\mathcal{C}$ by using the oracle replay technique, with the same random tape and different hash oracle for $\mathcal{O}_{H_3}$ simulates the game again and obtains two valid components v^* and v'.
  - Using v^* and v', $\mathcal{C}$ computes $\hat{v} = \dfrac{v^* - v'}{d^* - d'} = d_{A0} = sq_{A0} + a$.
  - Since $\mathcal{C}$ knows both s and q_{A0}, it computes $a = \hat{v} - sq_{A0}$.
  - Thus, the solution to the discrete log problem is obtained by $\mathcal{C}$.
- IF $ID_A \neq ID_\gamma$ then $\mathcal{C}$ aborts the game.

7.2 Type-II Unforgeability

Theorem 2. *The certificateless signcryption scheme CLSC is secure against any EUF-CLSC-CMA-II forger $\mathcal{F}_{II}$ in the random oracle model if DLP is hard in $\mathbb{Z}_P^*$.*

The proof of this theorem is omitted here due to page limitation and will appear in the full version of the paper.

7.3 Type-I Confidentiality

Theorem 3. *The certificateless signcryption scheme CLSC is secure against any EUF-CLSC-CCA2-I adversary $\mathcal{A}_I$ in the random oracle model if CDHP is hard in $\mathbb{Z}_P^*$.*

Proof: Challenger $\mathcal{C}$ is challenged with an instance of the CDH problem say $\langle g, g^a, g^b \rangle \in \mathbb{Z}_p^*$. Let us consider, there exists an adversary $\mathcal{A}_I$ who is capable of breaking the IND-CLSC-CCA2-I security of the CLSC scheme. $\mathcal{C}$ can make use of $\mathcal{A}_I$ to compute g^{ab}, by playing the following interactive game.

Setup: $\mathcal{C}$ begins the game by setting up the system parameters as in the CLSC scheme. $\mathcal{C}$ sets g as in the instance of the CDH problem it has received. Chooses $s \in_R \mathbb{Z}_q^*$, computes $g_{pub} = g^s$ and sends the public parameters $params = \langle p, q, g, g_{pub} \rangle$ to $\mathcal{A}_I$. $\mathcal{C}$ also designs the hash functions H_i as random oracles $\mathcal{O}_{H_i}$, (for $i = 1$ to 5). $\mathcal{C}$ maintains six lists L_1, L_{H_1}, L_{H_2}, L_{H_3}, L_{H_4} and L_{H_5} in order to consistently respond to the queries to the various random oracles.

Phase I: In this phase $\mathcal{A}_I$ interacts with $\mathcal{C}$ by querying the various oracles provided by $\mathcal{C}$. The oracles for which $\mathcal{A}_I$ have access and their responses during the game are identical and exactly same to the oracle descriptions for $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$, $\mathcal{O}_{H_3}$, $\mathcal{O}_{H_4}$, $\mathcal{O}_{H_5}$, $\mathcal{O}_{RequestPublicKey}$, $\mathcal{O}_{ReplacePublicKey}$, $\mathcal{O}_{ExtractPartialPrivateKey}$, $\mathcal{O}_{Extract\ SecretValue}$, $\mathcal{O}_{Signcrypt}$ and $\mathcal{O}_{Unsigncrypt}$ in the EUF-CLSC-CMA-I game.

Challenge: $\mathcal{A}_I$ outputs two equal length messages m_0, m_1 and an arbitrary sender identity ID_A for which $\mathcal{A}_I$ knows the private key d_{A0}. If $ID_B = ID_\gamma$, $\mathcal{C}$ randomly chooses a bit $d \in_R \{0,1\}$ and signcrypts m_d as follows.

- Computes $c_1^* = g^b$, chooses $r_2 \in_R \mathbb{Z}_q^*$ and computes $c_2^* = g^{r_2}$.
- Computes $k_1 = (g^b)^{x_{B1}}$.
- Chooses $c_3^* \in_R (\mathcal{M} \times \mathbb{Z}_q^* \times \mathbb{Z}_q^*)$

Now, $c^* = \langle c_1^*, c_2^*, c_3^* \rangle$ is sent to $\mathcal{A}_I$ as the challenge ciphertext.

Phase II: $\mathcal{A}_I$ adaptively queries the oracles as in ***Phase I***, consistent with the constraints for Type-I adversary. Besides this it cannot query *Unsigncryption* on c^*.

Guess: $\mathcal{A}_I$ is capable of breaking the IND-CLSC-CCA2-I security of CLSC (which is assumed at the beginning of the proof) and hence can find c^* is "*Invalid*",

- $\mathcal{A}_I$ should have computed $k_1' = (c_1^*)^{y_B}$ and $k_2' = (c_1^*)^{d_{B0}}$.
- Also, $\mathcal{A}_I$ should have queried $\langle k_1', k_2', ID_A, ID_B \rangle$ to the oracle $\mathcal{O}_{H_4}$.
- Now, when unsigncrypting c^*, $\mathcal{A}_I$ finds that c^* is an invalid ciphertext and hence *aborts*.

Therefore, if the list L_{H_4} has q_{H_γ} queries corresponding to the sender ID_A and receiver ID_B, one of the q_{H_γ} values stored in the list L_{H_4} is the solution for the CDHP instance. Now, $\mathcal{C}$ chooses one k_2 value uniformly at random from the q_{H_γ} values and outputs it as the solution for the CDH instance.

7.4 Type-II Confidentiality

Theorem 4. *The certificateless signcryption scheme CLSC is secure against any EUF-CLSC-CCA2-II adversary $\mathcal{A}_{II}$ in the random oracle model if CDHP is hard in $\mathbb{Z}_P^*$.*

The proof of this theorem is omitted here due to page limitation and will appear in the full version of the paper.

8 Conclusion

In this work, we have showed the security weakness in three existing certificateless signcryption schemes that appear in [2], [1] and [6]. We have also presented the first pairing free certificateless signcryption scheme in the random oracle model. The proposed scheme is more efficient since the scheme evades bilinear pairing. We have proved the security of the scheme with the strongest security notion for signcryption schemes, namely insider security. We leave it as an open problem to construct certificateless signcryption scheme without pairing in the standard model. As a concluding remark we present the complexity figure of the new CLSC scheme in the following table.

Table 1. Complexity figure for CLSC

Scheme	Signcrypt	Unsigncrypt
CLSC	5 *EXP*	7 *EXP*

EXP - Exponentiation in group $\mathbb{G}$.

References

1. Aranha, D., Castro, R., Lopez, J., Dahab, R.: Efficient certificateless signcryption, http://sbseg2008.inf.ufrgs.br/proceedings/data/pdf/st03_01_resumo.pdf
2. Barbosa, M., Farshim, P.: Certificateless signcryption. In: ACM Symposium on Information, Computer and Communications Security - ASIACCS 2008, pp. 369–372. ACM, New York (2008)
3. Barreto, P.S.L.M., Deusajute, A.M., de Souza Cruz, E., Pereira, G.C.F., da Silva, R.R.: Toward efficient certificateless signcryption from (and without) bilinear pairings, http://sbseg2008.inf.ufrgs.br/proceedings/data/pdf/st03_03_artigo.pdf
4. Dutta, R., Barua, R., Sarkar, P.: Pairing-based cryptographic protocols: A survey (2004), Cryptology ePrint Archive, Report 2004/064, http://eprint.iacr.org/
5. Sun, Y., Zhang, F., Baek, J.: Strongly secure certificateless public key encryption without pairing. In: Bao, F., Ling, S., Okamoto, T., Wang, H., Xing, C. (eds.) CANS 2007. LNCS, vol. 4856, pp. 194–208. Springer, Heidelberg (2007)
6. Wu, C., Chen, Z.: A new efficient certificateless signcryption scheme. In: IEEE, International Symposium on Information Science and Engieering, ISISE 2008, vol. 1, pp. 661–664 (2008)

Sanitizable Signatures with Strong Transparency in the Standard Model

Shivank Agrawal, Swarun Kumar, Amjed Shareef, and C. Pandu Rangan

Theoretical Computer Science Lab,
Department of Computer Science and Engineering,
Indian Institute of Technology Madras, India
{shinku100,swarun.s,amjedshareef}@gmail.com, prangan@iitm.ac.in

Abstract. Sanitizable signatures provide several security features which are useful in many scenarios including military and medical applications. Sanitizable signatures allow a semi-trusted party to update some part of the digitally signed document without interacting with the original signer. Such schemes, where the verifier cannot identify whether the message has been sanitized, are said to possess strong transparency. In this paper, we have described the first efficient and provably secure sanitizable signature scheme having strong transparency under the standard model.

Keywords: sanitizable signatures, strong transparency, standard model.

1 Introduction

Applications like e-government and e-tax payment systems require appropriate alteration of digitally signed documents in order to hide personal information. Sanitizable signatures came into much attention when recently, government entities were forced to disclose documents owing to disclosure laws. In the past, when secret paper documents were made declassified, hiding of sensitive or personal information in the document was done by blackening-out (sanitizing) relevant sections of the documents. A digital signature, however, prohibits any alteration of the original message once it is signed. So, in the world of digital signatures, sanitization cannot be done.

A sanitizable signatures protects the confidentiality of a specified part of the document while ensuring the integrity of the document. A solution for this problem was proposed earlier in [1] as *content extract signatures*. In 2005, Ateniese et al. [2] introduced *sanitizable signatures* which can alter the signed document instead of hiding it. A sanitizable signature scheme is a signature scheme which allows a designated party, called the *sanitizer*, to hide certain parts of the original message after the message is signed, without interacting with the signer. The verifier confirms the integrity of disclosed parts of the sanitized document from the signature and sanitized document. In other words, a sanitizable signature scheme allows a semi-trusted *sanitizer* to modify designated portions of the document and produce a valid signature on the legitimately modified document without any interaction with the original signer. These designated portions of the document are blocks or segments explicitly indicated as mutable under prior agreement between the signer and the sanitizer. The sanitizer can produce a valid signature only if it modifies these portions and no other parts of the message. Following these works several authors [3,4,5,6,7,8] proposed various sanitizable signature schemes with different properties.

There are different types of sanitizable schemes present in literature. In some schemes, sanitization on any part of the message can be performed by the sanitizer, while in others, sanitization on some parts of the messages can be restricted by the signer. Transparency is a another property of sanitizable signature schemes [2,9]. If the verifier knows which part of the document is sanitized, then the scheme has no transparency. If he does not know whether the message is sanitized, then the scheme has weak transparency. If he also does not know whether the message can be sanitized, then the scheme is said to have *strong transparency.*

1.1 Our Contributions

In this paper, we have provided two protocols for strong transparency in the *standard model* using bilinear pairing. Our construction is based on Waters's scheme [10]. These are the *first* efficient and secure schemes which provide strong transparency under the standard model. In our first protocol we achieve strong transparency by providing some secret information to the sanitizer, where the portions of the message to be sanitized are specified by the signer. In our second protocol we remove the need to send secret information to the sanitizer on a per-message basis, provided that the blocks of message which need to be sanitized are fixed beforehand. This requirement holds in forms, databases, etc. The length of our sanitized signature is equivalent that of Waters' [10] signature, hence shorter than the signatures produced by other protocols. Our scheme uses techniques similar to the sanitizable signature scheme in [9] and provides additional properties. The scheme in [9] does not provide transparency. We also compare our scheme with other sanitizable signature schemes proposed in literature.

1.2 Applications

Sanitizable signatures are well-suited for customizing authenticated multicast transmissions. Consider a subscription-based multimedia database, where sponsors may wish to insert personalized commercials into authenticated broadcasted messages. One solution is for each vendor to sign the commercial once and allow the database administrator to customize the individual commercials by replacing the generic identity field with the actual subscriber's identity, at various points of the commercial. Hence, the subscriber can verify that the commercial is legitimate and the sponsors need not sign each customized broadcast. Detailed motivation and applications of sanitizable signatures including medical applications, secure routing, e-governance, etc. are available in [2].

2 Preliminaries

2.1 Bilinear Pairing

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be a multiplicative groups of prime order p. The elements $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow G_T$ with the following properties:

1. **Bilinear**: $e({g_1}^a, {g_2}^b) = e(g_1, g_2)^{ab}$ for all $g_1 \in \mathbb{G}_1$,$g_2 \in \mathbb{G}_2$, where $a, b \in \mathbb{Z}_p$.
2. **Non-degenerate**: There exists $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ such that $e(g_1, g_2) \neq 1$; in other words, the map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_2$ to the identity in $\mathbb{G}_T$.
3. **Computability**: There is an efficient algorithm to compute $e(g_1, g_2)$ for all $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.

2.2 Security Assumptions

Definition 1. The Computational Diffie-Hellman (CDH) problem is that, given g, g^x, $g^y \in \mathbb{G}$ for unknown x,$y \in Z_p*$, to compute g^{xy}.
We say that the ϵ-*CDH assumption* holds in $\mathbb{G}$ if no polynomial-time algorithm has non-negligible probability ϵ in solving the CDH problem.

2.3 Sanitizable Signature

A sanitizable signature scheme is a signature scheme that allows the sanitizer to sanitize certain portions of the document and to generate the valid signature of the resulting document with no interaction with the signer. A sanitizable signature is processed by three parties consisting of a signer, a sanitizer, and a verifier. The signer generates the signature assuring the authenticity of the document. The sanitizer receives the document and its signature from the signer. The sanitizer generates the sanitized document and its signature without any help of the signer. The verifier receives the sanitized document and its signature from the sanitizer. The verifier accepts the signature only if he verifies the authenticity of the disclosed document.

2.4 Transparency

A sanitizable signature scheme may have various levels of *transparency*, which we define below:

1. **No transparency.** The verifier knows which part of the document is sanitized.
2. **Weak transparency.** The verifier does not know if the message is sanitized. The verifier only knows if the message can be disclosed and sanitizing is prohibited or not.
3. **Strong transparency.** The verifier does not know if the message can be sanitized. In this model no extra information is sent to the verifier other than message and a signature.

2.5 State Information

In our first scheme, the signer can control the states of the bits of the document, i.e., whether sanitization is allowed or sanitization is prohibited. This state information is kept secret to achieve strong transparency. The signer generates the secret information for each message of the document. The secret information is necessary to generate the signature of the sanitized document. The signer sends the secret information of the message to the sanitizer if he allows the sanitizer to sanitize the message. Otherwise he does not send the secret information of the message to the sanitizer.

In our second scheme, the sanitizer is given the power to control certain bits of the message a priori in the set-up phase. In this case, no secret information needs to be sent to the sanitizer by the signer.

2.6 Scheme Outline

– **Key Generation.** Algorithm *KeyGen*, executed by the PKG, inputs a security parameter 1^k and outputs public parameters *param*, public and secret key pair for the signer (PK, SK) and secret key of the sanitizer SK', if any.
– **Signing.** Algorithm *Sign*, executed by the signer, takes as input a document M, public parameters *param* and secret key SK. Let $M = m_1 m_2 \cdots m_n \in \{0,1\}^n$, where m_i is defined as the bit at index i of message M. Let $I_{\mathcal{S}} \subseteq \{1, \cdots, n\}$ denote the set of indices that the sanitizer is allowed to modify. The algorithm outputs a document M, signatures(σ_1, σ_2) of M and secret information SI for the sanitizer, if any.
– **Sanitization.** Algorithm *Sanitize*, executed by the sanitizer, takes message M, public parameters *param*, signature σ on M, sanitizer's secret key SK', if any, secret information from the signer SI, if any, and outputs a message M' and sanitized signature σ'.
– **Verification.** Algorithm *Verify*, executed by the verifier, takes as input an unsanitized document and signature (M, σ) or a sanitized document and signature (M',σ'), public parameters *param*, and public key PK of signer, outputs *accept* or *reject*. The strong transparency property requires that the verifier not be able to find out whether the document is sanitized or not. Hence the verification procedure remains the same for both sanitized and unsanitized documents.

3 Security Model

3.1 Correctness

We require that $\mathit{Verify}(\sigma, M, PK, \mathit{param}) = \mathit{accept}$, for an unsanitized message M if :

1. $(PK, SK, \mathit{param}) \leftarrow \mathit{KeyGen}(1^k)$,
2. $\sigma \leftarrow \mathit{Sign}(M, SK, \mathit{param})$,

We additionally require that $\mathit{Verify}(\sigma', M', PK, \mathit{param}) = \mathit{accept}$, for an sanitized message M' if:

1. $(PK, SK, \mathit{param}) \leftarrow \mathit{KeyGen}(1^k)$,
2. $\sigma \leftarrow \mathit{Sign}(M, SK, \mathit{param})$,
3. $(M', \sigma') \leftarrow \mathit{Sanitize}(M, \sigma, PK, SI, \mathit{param})$

3.2 Unforgeability

We have the following game $\mathsf{Exp}_{\mathrm{unf}}$ for unforgeability:

1. The simulator S gives *param* and PK to the adversary A.
2. A is allowed to query the signing oracle q_s times adaptively. During the j^{th} query, on inputing a document $M_j = m_{j,1} \cdots m_{j,n}$, the oracle returns the corresponding signature σ_j on M_j.
3. Finally A outputs a document M^* , a signature σ^*.

A wins if $\mathit{Verify}(\sigma^*, M^* PK, \mathit{param}) = \mathit{accept}$ and the message M^* is not equal to any query message M_j for $1 \leq j \leq q_s$.

Note that the adversary is *not provided a sanitization oracle*, as a sanitized signature is indistinguishable from a normal signature by the signer on the same message. This follows from strong transparency. This security model for unforgeability is also present in [11].

Definition 2. A sanitizable signature scheme is (ϵ, q_s)- unforgeable if there is no randomized polynomial time adversary winning the above game with probability at least ϵ with at most q_s queries to the signing oracle.

3.3 Indistinguishability

We have the following game $\mathsf{Exp}_{\mathrm{ind}}$ for indistinguishability:

1. The simulator S gives *param* and PK to the adversary A.
2. A is allowed to query the signing oracle q_s times adaptively. The oracle is the same as the one in the game for unforgeability.
3. A sends two different signatures σ_0, σ_1 on M_0, M_1 respectively and a sanitized message M', where M' differs from M_0 and M_1 only at bits that are allowed to be sanitized.
4. S picks a random bit b and sends σ'_b to A which is the signature obtained from the sanitization of message M_b.
5. Finally, A outputs bit b'

A wins the game if $b = b'$. The advantage of A is $|\Pr[b = b'] - 1/2]|$.

Definition 3. A sanitizable signature scheme is said to be unconditionally indistinguishable if there is no adversary winning the above game with advantage greater than 0 with any number of queries to the signing oracle.

3.4 Immutability

We have the following game $\mathsf{Exp}_{\text{imm}}$ for immutability. Let $I_{\mathcal{S}}$ be the set of positions of the bits in the message that the sanitizer is allowed to modify. Here the adversary is a sanitizer who attempts to sanitize bits outside his permissible set $I_{\mathcal{S}}$.

1. A sends a challenge set $I_{\mathcal{S}}$, the set of bit positions where sanitization is allowed.
2. The simulator S gives the public parameters *param* and PK to the adversary A.
3. In scheme-2, the one-time secret information corresponding to the set $I_{\mathcal{S}}$ is also given to the adversary.
4. A is allowed to query the signing oracle q_s times adaptively. During the j^{th} query, on input a document $M_j = m_{j,1} \cdots m_{j,n}$, the oracle returns the corresponding signature σ_j on M_j.
5. In scheme-1, A additionally obtains secret values SK_j along with σ_j. This enables A to sanitize bits at positions $I_{\mathcal{S}}$.
6. Finally, A outputs a document $M^* = m_1^* \cdots m_n^*$, a signature σ^*, where $\forall j \in \{1, \cdots, q_s\}\ \exists i \notin I_{\mathcal{S}} : m_{j,i} \neq m_i^*$.

A wins the game if signature σ^* on M^* verifies successfully. The advantage of A is the probability that A succeeds. This security model is in accordance with [12]. Note that accountability is not required in our model as this property compromises the unconditional indistinguishability of our scheme.

Definition 4. A sanitizable signature scheme is ϵ- immutable if there is no randomized polynomial time adversary winning the above game with probability at least ϵ.

4 Scheme 1

4.1 Outline

This scheme provides a strong transparent sanitizable signature protocol where the signer is proactive in deciding which bits need to be sanitized and by which sanitizer The signer sends the indices which are permitted to be sanitized as well as one time secret information that enables sanitization of the relevant portions of the message to the sanitizer, in a secure fashion. The sanitizer may replace those portions of the message by an appropriate message of his choice.

4.2 Scheme Description

KeyGen. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of prime order p. Given a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. We denote by n, the number of bits of the message m. Let $g \in \mathbb{G}_1$ and $g_2, u', u_1, \cdots, u_n \in \mathbb{G}_2$.

Public parameters: $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, p, g_2, u', u_1, \cdots, u_n$. Public key of the signer is $g_1 = g^{\alpha}$, where $\alpha \in \mathbb{Z}_p^*$.

Private parameters: Private key of the signer is $\alpha \in_R \mathbb{Z}_p^*$.

Sign. Let m be the n-bit message $m_1 m_2 \cdots m_n \in \{0,1\}^n$. Signer randomly picks $r \in \mathbb{Z}_p^*$ and outputs the following values (σ_1, σ_2):

$$(\sigma_1 = g_2^{\alpha}(u' \prod_{i=1}^{n} u_i^{m_i})^r, \sigma_2 = g^r)$$

Let $I_{\mathcal{S}}$ be the set of indices that the sanitizer is permitted to modify. Then the signer sends the values $u_i^r \ \forall i \in I_{\mathcal{S}}$ to the sanitizer in a secure channel. Alternately, these values may be encrypted by the public key of the sanitizer and sent across.

Sanitize. The sanitizer obtains the values (σ_1, σ_2), and the secret information $u_i^r \ \forall i \in I_{\mathcal{S}}$ from the signer. It runs the verification protocol to check if the signature is valid. Let m' be the message whose signature is sought, which differs from m at positions $I \subseteq I_{\mathcal{S}}$. Define $I_1 = \{i \in I : m_i = 0, m_i' = 1\}$, $I_2 = \{i \in I : m_i = 1, m_i' = 0\}$. The sanitizer chooses $\tilde{r} \in_R \mathbb{Z}_p^*$. Then the required sanitized signature is:

$$(\sigma_1' = \sigma_1 \frac{\prod_{i \in I_1} u_i^r}{\prod_{i \in I_2} u_i^r} u'^{\tilde{r}} \prod_{i=1}^{n} u_i^{m_i' \tilde{r}}, \sigma_2' = \sigma_2 g^{\tilde{r}})$$

Verify. The verifier receives the tuple: (σ_1, σ_2) on a message m,
Verifier checks if the following relation holds from public parameters:

$$e(g, \sigma_1) \stackrel{?}{=} e(g_1, g_2) e(\sigma_2, u' \prod_{i=1}^{n} u_i^{m_i})$$

Note that the verification protocol is same for a sanitized and non-sanitized message.

4.3 Security

Correctness. To show correctness, we need to show that any valid normal signature, as well as sanitized signature verifies successfully.

Verification: The signature σ, on a given message m is given by the two-tuple $(\sigma_1 = g_2^{\alpha}(u' \prod_{i=1}^{n} u_i^{m_i})^r, \sigma_2 = g^r)$. If valid, then clearly:

$e(g_1, g_2)e(\sigma_2, u' \prod_{i=1}^n u_i^{m_i})$
$= e(g^a, g_2)e(g^r, u' \prod_{i=1}^n u_i^{m_i})$
$= e(g, g_2^a)e(g, (u' \prod_{i=1}^n u_i^{m_i})^r)$
$= e(g, g_2^a(u' \prod_{i=1}^n u_i^{m_i})^r)$
$= e(g, \sigma_1)$

Hence, a valid signature satisfies the verification equation.

Sanitization: A sanitized signature is obtained as:

$$(\sigma_1' = \sigma_1 \frac{\prod_{i \in I_1} u_i^r}{\prod_{i \in I_2} u_i^r} u'^{\tilde{r}} \prod_{i=1}^n u_i^{m_i' \tilde{r}}, \sigma_2' = \sigma_2 g^{\tilde{r}})$$

where $I_1 = \{i \in I_{\mathcal{S}} : m_i = 0, m_i' = 1\}$, $I_2 = \{i \in I_{\mathcal{S}} : m_i = 1, m_i' = 0\}$.
We note that $m_i' - m_i$ is 1 when $i \in I_1$, -1 when $i \in I_2$,and 0, otherwise. Hence, we can see that:

$\sigma_1' = \sigma_1 u'^{\tilde{r}} \prod_{i \in I_1} u_i^r / \prod_{i \in I_2} u_i^r \prod_{i=1}^n u_i^{m_i' \tilde{r}}$
$= \sigma_1 u'^{\tilde{r}} \prod_{i=1}^n u_i^{r(m_i' - m_i)} \prod_{i=1}^n u_i^{m_i' \tilde{r}}$
$= g_2^{\alpha} u'^r u'^{\tilde{r}} \prod_{i=1}^n u_i^{r(m_i)} \prod_{i=1}^n u_i^{r(m_i' - m_i)} \prod_{i=1}^n u_i^{m_i' \tilde{r}}$
$= g_2^{\alpha} u'^{(r+\tilde{r})} \prod_{i=1}^n u_i^{(r+\tilde{r})(m_i')}$

The sanitized signature is of the form $(\sigma_1' = g_2^{\alpha}(u' \prod_{i=1}^n u_i^{m_i'})^{(r+\tilde{r})}, \sigma_2' = g^{(r+\tilde{r})})$, whose distribution is identical to a regular signature on m' by the signer. Hence, a sanitized signature also satisfies the verification equation.

Unforgeability. We prove the following theorem about unforgeability.

Theorem 1. *The proposed sanitizable signature scheme in scheme-1 is* (ϵ, q_s)*-unforgeable under the* ϵ'*-CDH assumption where* $\epsilon \le (8q_s^2(n+1)^2 + 2)\epsilon' + 2/p$*, where* q_s *is the polynomial number of queries.*
Proof. Assume there is a (ϵ, q_s)-adversary A exists. We shall formulate another probabilistic polynomial time (PPT) algorithm B that uses A to solve the CDH problem with probability at least ϵ' and in time at most t'. B is given a problem instance as follow: Given a group $\mathbb{G}$, a generator $g \in \mathbb{G}$, two elements $g^a, g^b \in \mathbb{G}$. It is asked to output another element $g^{ab} \in G$. In order to use A to solve for the problem, B needs to simulates a challenger and the signing oracle for A. B does it in the following way (Recall here that g^a and g^b are the input for the CDH problem that B should solve).

Setup Phase. Let $l = 2q_s$. B randomly selects an integer k such that $0 \le k \le n$. Also assume that $l(n+1) < p$, for the given values of q_s and n. It randomly selects:

1. $x' \in_R \mathbb{Z}_l; y' \in_R \mathbb{Z}_p$
2. $\hat{x}_i \in_R \mathbb{Z}_l$, Let $\hat{X} = \{\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_n\}$.
3. $\hat{y}_i \in_R \mathbb{Z}_p$, Let $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \cdots, \hat{x}_n\}$.

We further define the following functions for binary string $M = (m_1, m_2, \cdots, m_n)$, where $m_i \in \{0,1\}$ $1 \le i \le n$, as follows:

$$F(M) = x' + \sum_{i=1}^{n} \hat{x}_i m_i - lk, \quad J(M) = y' + \sum_{i=1}^{n} \hat{y}_i m_i$$

B constructs a set of public parameters as follows: $g_2 = g^b$, $u' = g_2^{-lk+x'} g^{y'}$, $u_i = g_2^{\hat{x}_i} g^{\hat{y}_i}$, $i = 1, \cdots, n$. We have the following equation:

$$u' \prod_{i=1}^{n} u_i^{m_i} = g_2^{F(M)} g^{J(M)}$$

All the above public parameters and public key $g_1 = g^a$ are passed to A.

Simulation Phase. B simulates the signing oracle as follow. Upon receiving the j^{th} query for a document M_j, although B does not know the secret key, it can still construct the signature by assuming $F(M_j) \neq 0 \bmod p$. It randomly chooses $r_j \in_R \mathbb{Z}_p$ and computes the signature as

$$\sigma_{1,j} = g_1^{-J(M_j)/F(M_j)} (g_2^{F(M_j)} g^{J(M_j)})^{r_j}, \sigma_{2,j} = g_1^{-1/F(M_j)} g^{r_j}$$

By letting $\hat{r}_j = r_j - a/F(M_j)$, it can be verified that $(\sigma_{1,j}, \sigma_{2,j})$ is a valid signature on M_j as shown below:

$$\begin{aligned}
\sigma_{1,j} &= g_1^{-\frac{J(M_j)}{F(M_j)}} (g_2^{F(M_j)} g^{J(M_j)})^{r_j} \\
&= g^{-a\frac{J(M_j)}{F(M_j)}} (g_2^{F(M_j)} g^{J(M_j)})^{\frac{a}{F(M_j)}} (g_2^{F(M_j)} \ g^{J(M_j)})^{-\frac{a}{F(M_j)}} \ (g_2^{F(M_j)} g^{J(M_j)})^{r_j} \\
&= g_2^a (g_2^{F(M_j)} g^{J(M_j)})^{\hat{r}_j} \\
\sigma_{2,j} &= g_1^{-1/F(M_j)} g^{r_j} = g^{r_j - a/F(M_j)} = g^{\hat{r}_j}
\end{aligned}$$

If $F(M_j) = 0 \bmod p$, since the above computation cannot be performed (division by 0), the simulator aborts. To make it simple, the simulator will abort if $F(M_j) = 0 \bmod l$. The equivalence can be observed as follow. From the assumption that $l(n+1) < p$, it implies $0 \le lk < p$ and $0 \le x' + \sum_{i=1}^{n} \hat{x}_i m_i < p$ (as $x' < l, \hat{x}_i < l$). We have $-p < F(M_j) < p$ which implies if $F(M_j) = 0 \bmod p$ then $F(M_j) = 0 \bmod l$. Hence, $F(M_j) \neq 0 \bmod l$ implies $F(M_j) \neq 0 \bmod p$. Thus the former condition will be sufficient to ensure that a signature can be computed without aborting.

Challenge Phase. If B does not abort, A will return a document $M^* = m_1^* \cdots m_n^*$ with a forged signature $\sigma^* = (\sigma_1^*, \sigma_2^*)$. The algorithm B aborts if $x' + \sum_{i|m_i^*=1} \hat{x}_i - lk \neq 0 \bmod l$. From the verification equation, we can write:

$$\begin{aligned}
\sigma_1^* &= g_2^a (g_2^{F(M^*)} g^{J(M^*)})^{r^*} \\
&= g_2^a (g_2^{(x' + \sum_{i|m_i^*=1} \hat{x}_i - lk) r^*} \quad g^{(y' + \sum_{i|m_i^*=1} \hat{y}_i) r^*} \\
&= g_2^a g^{(y' + \sum_{i|m_i^*=1} \hat{y}_i) r^*}
\end{aligned}$$

Hence the algorithm successfully computes the solution to the CDH problem:

$$Z = \sigma_1^* \sigma_2^{* -y' - \sum_{i|m_i^*=1} \hat{y}_i} = g_2^a = g^{ab} \qquad \square$$

Probability Analysis. The probability that the simulation does not abort is characterized by the events A_j, A^* where:

1. A_j is the event that $F(M_j) \neq 0 \bmod l$ where $j = 1, \cdots, q_s$.
2. A^* is the event that $x' + \sum_{i|m_i^*=1} \hat{x}_i - lk = 0 \bmod p$.

The probability that B does not abort: $\Pr[\text{not abort}] \geq \Pr[\bigwedge_{j=1}^{q_s} A_j \wedge A^*]$.
As the adversary can at most make B abort by randomly choosing M^*, we have $\Pr[A^*] = \frac{1}{l(n+1)}$. Also noting that A_j is independent of A^* we have:

$$\begin{aligned}\Pr[\text{not abort}] &\geq \Pr[\textstyle\bigwedge_{j=1}^{q_s} A_j \wedge A^*] \\ &\geq \Pr[A^*]\Pr[\textstyle\bigwedge_{j=1}^{q_s} A_j | A^*] \\ &\geq \tfrac{1}{(l(n+1))^2}(1 - \textstyle\sum_{j=1}^{q_s} \Pr[\neg A_j | A^*]) \\ &\geq \tfrac{1}{8(n+1)^2 q_s^2}\end{aligned}$$

Indistinguishability. As shown in the correctness section, a valid signature σ_s produced by a signer on a message m' has a distribution identical to a valid sanitization of a message m_1 to result in message m' and signature σ_a' produced by the sanitizer. Similarly, the distribution is also identical to a valid sanitization of another message m_2 to result in message m' and signature σ_b', produced by the sanitizer. Hence the signatures σ_a' and σ_b' are indistinguishable as their distributions are identical.

Immutability. We prove the following theorem to show immutability.

Theorem 2. The proposed sanitizable signature scheme in scheme-1 is ϵ-immutable under the ϵ'-CDH assumption, where there exists constant $l : \epsilon < l\epsilon'$.

Proof: We prove that the sanitizer cannot modify any bit other than bits at positions $I_{\mathcal{S}} \subseteq \{1, \cdots, n\}$ for which the values $\{u_i^r : i \in I_{\mathcal{S}}\}$ are known to the sanitizer. We will prove the following lemma on immutability in order to prove the above theorem.

Lemma 1. For any randomized polynomial time algorithm algorithm B with an advantage ϵ_b in the immutability game $\mathsf{Exp}_{\text{imm}}$ on a message of length n with access to sanitize m bits at positions $I_{\mathcal{S}}$, there exists a randomized polynomial time algorithm A with an advantage $\epsilon_a \geq \epsilon_b$ in the unforgeability game $\mathsf{Exp}_{\text{unf}}$ on a message of length $n - m$.

Proof: Assume that there exists a randomized polynomial time algorithm B which plays the immutability game $\mathsf{Exp}_{\text{imm}}$ with advantage ϵ_b with access to sanitize m bits at positions $I_{\mathcal{S}}$. Consider a randomized polynomial time algorithm A which plays the unforgeability game $\mathsf{Exp}_{\text{unf}}$ on messages of length $n-m$.

Then we show that the algorithm A can simulate the challenger interacting with algorithm B, and thereby obtain an advantage $\epsilon_a \geq \epsilon_b$ in $\mathsf{Exp}_{\text{unf}}$. In the setup phase, A interacts with B and the challenger in $\mathsf{Exp}_{\text{unf}}$, denoted by C, as follows:

1. B provides A the set, $I_{\mathcal{S}}$, of bit positions where sanitization is allowed. In general, we have $I_{\mathcal{S}} \subseteq \{1, \cdots, n\}$. However for the ease of exposition we assume $I_{\mathcal{S}} = \{n-m+1, \cdots, n\}$, where $m = |I_{\mathcal{S}}|$. Note that the argument can be easily extended for the general form of $I_{\mathcal{S}}$.
2. C provides A the public parameters $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, p, g_2, u', u_1, \cdots, u_{n-m}$.
3. A chooses $t_i \in_R \mathbb{Z}_p^*$, $i = n-m+1, \cdots, n$. A sets $u_i' = g^{t_i'}$, for $i = n-m, \cdots, n$
4. A provides B the public parameters $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, p, g_2, u', u_1, \cdots, u_{n-m}$, $u'_{n-m+1} \cdots, u'_n$.

In the simulation phase, for every message M_j, $j = 1, \cdots, q_s$, requested by B, A interacts with B and C as follows:

1. B requests signature for a message $M_j = m_{j,1} \cdots m_{j,n}$ from A.
2. A requests a signature for message $M_j = m_{j,1} \cdots m_{j,n-m}$ from C.
3. A obtains $(\sigma_{j,1}, \sigma_{j,2})$ from C, and sets $\sigma'_{j,1} = \sigma_{j,1} \prod_{i=n-m+1}^{n} \sigma_{j,2}^{t_i' m_{j,i}}$ and $\sigma'_{j,2} = \sigma_{j,2}$.
4. A sends the signature $(\sigma'_{j,1}, \sigma'_{j,2})$ to B and the secret information $\{\sigma_{j,2}^{t_i' m_{j,i}} | i = n-m+1, \cdots, n\}$ to B.

In the challenge phase, if B is successful in obtaining a valid message signature pair $(M^{*\prime}, \sigma^{*\prime})$, then A obtains a valid signature tuple as follows:

1. B sends A a valid message-signature tuple $(M^{*\prime} = m_1^{*\prime} \cdots m_n^{*\prime}, \sigma^{*\prime} = {\sigma^*}'_1, \sigma_2^{*\prime})$. Clearly $\forall j \in \{1, \cdots, q_s\}\ \exists i \notin \{n-m+1, \cdots, n\} : m_{j,i} \neq m_i^{*\prime}$.
2. A sets $M^* = m_1^* \cdots m_{n-m}^*$, where $m_i^* = m_i^{*\prime}$ for all $i = 1, \cdots, n-m$. A sets

$$\sigma_1^* = \frac{\sigma_1^{*\prime}}{\prod_{i=n-m+1}^{n} \sigma_2^{t_i' m_{j,i}^{*}{}'}}, \sigma_2^* = \sigma_2^{*\prime}$$

3. A sends C a valid message-signature pair $(M^*, \sigma^* = (\sigma_1^*, \sigma_2^*))$. Clearly, it follows that $\forall j \in \{1, \cdots, q_s\}\ \exists i \in \{1, \cdots, n-m\} : m_{j,i} \neq m_i^*$.

It is easy to see that if B's signature tuple verifies, then A's signature tuple verifies as well. Hence the advantage of A winning the game $\mathsf{Exp}_{\text{unf}}$, $\epsilon_a \geq \epsilon_b$, where ϵ_b is the advantage of B in winning the immutability game $\mathsf{Exp}_{\text{imm}}$.

From theorem-1, the advantage of any probabilistic polynomial time algorithm in winning the unforgeability game $\mathsf{Exp}_{\text{unf}}$ is negligible under the CDH assumption. Applying lemma-1, clearly the advantage of any probabilistic polynomial time algorithm in winning the immutability game $\mathsf{Exp}_{\text{imm}}$ is also negligible under the CDH assumption. This proves theorem-2.

5 Scheme 2

5.1 Outline

This scheme provides a strong transparent sanitizable signature protocol where the sanitizer is provided private information in the key generation phase. Using this, he may modify certain fixed set of positions of the signature. The indices which are permitted to be sanitized are fixed at the time of key generation. The sanitizer may replace those portions of the message by an appropriate message of his choice.

5.2 Scheme Description

KeyGen. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of prime order p. Given a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Let $g \in \mathbb{G}_1$ and $g_2, u', u \in \mathbb{G}_2$. Let $\alpha_1, \cdots, \alpha_n \in \mathbb{Z}_p^*$. Compute $u_1 = u^{\alpha_1}, \cdots, u_n = u^{\alpha_n}$.

Public parameters: $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, p, g_2, u', u, u_1, \cdots, u_n$ Public key of the signer is $g_1 = g^{\alpha}$, where $\alpha \in \mathbb{Z}_p^*$.

Private parameters: Private key of the signer is $\alpha \in_R \mathbb{Z}_p^*$.
Private key of a sanitizer j with access to modify bit positions $I_{\mathcal{S}_j} \subseteq \{1, \cdots, n\}$ is $\alpha_i \ \forall i \in I_{\mathcal{S}_j}$.

Sign. Let m be the n-bit message $m_1 m_2 \cdots m_n \in \{0,1\}^n$. Signer randomly picks $r \in \mathbb{Z}_p^*$ and outputs the following values $(\sigma_1, \sigma_2, \sigma_3)$:

$$(\sigma_1 = g_2^{\alpha}(u' \prod_{i=1}^{n} u_i^{m_i})^r, \sigma_2 = g^r, \sigma_3 = u^r)$$

Sanitize. The sanitizer $\mathcal{S}_j$ obtains the values $(\sigma_1, \sigma_2, \sigma_3)$, from the signer. It runs the verification protocol to check if the signature is valid. It then computes the values $u_i^r \leftarrow \sigma_3^{\alpha_i} \ \forall i \in I_{\mathcal{S}_j}$. Let m' be the message whose signature is sought, which differs from m at positions $I \subseteq I_{\mathcal{S}_j}$. Define $I_1 = \{i \in I : m_i = 0, m_i' = 1\}$, $I_2 = \{i \in I : m_i = 1, m_i' = 0\}$. The sanitizer chooses $\tilde{r} \in_R \mathbb{Z}_p^*$. Then the required sanitized signature is:

$$(\sigma_1' = \sigma_1 \frac{\prod_{i \in I_1} u_i^r}{\prod_{i \in I_2} u_i^r} u'^{\tilde{r}} \prod_{i=1}^{n} u_i^{m_i' \tilde{r}}, \sigma_2' = \sigma_2 g^{\tilde{r}}, \sigma_3' = \sigma_3 u^{\tilde{r}})$$

Verify. Receives the tuple: $(\sigma_1, \sigma_2, \sigma_3)$ on a message m,
Verifier checks if the following relations hold from public parameters:

$$e(g, \sigma_1) \stackrel{?}{=} e(g_1, g_2) e(\sigma_2, u' \prod_{i=1}^{n} u_i^{m_i})$$

$$e(g, \sigma_3) \stackrel{?}{=} e(\sigma_2, u) \tag{1}$$

Note that the verification protocol is same for a sanitized and non-sanitized message. Note that this scheme provides the right to sanitize the same positions across multiple users. If this is to be avoided, the parameters $u_1, \cdots, u_n$ must be part of the public key. Details are present in the full version of the paper [13].

5.3 Security

We provide the sketches for proof of correctness, unforgeability, indistinguishability, and immutability for scheme-2. The complete proofs may be found in the full version of this paper [13].

Correctness. The proof of correctness of the above scheme is similar to that of scheme-1. Additionally, we note that (1) holds since:
$e(g, \sigma_3) = e(g, u^r) = e(g^r, u) = e(\sigma_2, u)$

Unforgeability. We prove the following theorem about unforgeability.

Theorem 3. *The proposed sanitizable signature scheme in scheme-2 is (ϵ, q_s)-unforgeable under the ϵ'-CDH assumption where $\epsilon \leq (8q_s^2(n+1)^2 + 2)\epsilon' + 2/p$.*

Proof Sketch: The proof of unforgeability of scheme-2 is on the same lines as that of scheme-1. We highlight the important differences below:

1. B sets public parameter $u = g^v$ where $v \in_R \mathbb{Z}_l$.
2. In the simulation phase, B simulates $\sigma_{3,j}$ on receiving the j^{th} query for a message M_j as follows:

$$\sigma_{3,j} = g_1^{-v/F(M_j)} g^{vr_j}$$

   It is easy to verify that $(\sigma_{1,j}, \sigma_{2,j}, \sigma_{3,j})$ is indeed a valid signature tuple.

Indistinguishability. Similar to scheme-1, one can easily observe that the distributions of a signature produced by the signer and the sanitizer are identical.

Immutability. We prove the following theorem to show immutability.

Theorem 4. The proposed sanitizable signature scheme in scheme-2 is ϵ-immutable under the ϵ'-CDH assumption, where there exists constant $l : \epsilon < l\epsilon'$.

Proof Sketch: The proof is on the same lines as the proof of Theorem-2. The differences in the proof due to the additional signature component σ_3 are given below:

1. In the simulation phase, the adversary A obtains a 3-tuple $(\sigma_{j,1}, \sigma_{j,2}, \sigma_{j,3})$ in the j^{th} query from C, and sets $\sigma'_{j,1} = \sigma_{j,1} \prod_{i=n-m+1}^{n} \sigma_{j,3}^{t'_i m_{j,i}}$, $\sigma'_{j,2} = \sigma_{j,2}$ and $\sigma'_{j,3} = \sigma_{j,3}$. She sends $(\sigma'_{j,1}, \sigma'_{j,2}, \sigma'_{j,3})$ to B.
2. In the challenge phase, A obtains a valid signature tuple $(\sigma_1^*, \sigma_2^*, \sigma_3^*)$ from a valid tuple $({\sigma^*}'_1, {\sigma_2^*}', {\sigma_3^*}')$ sent by B as follows:

$$\sigma_1^* = \frac{\sigma_1^{*\prime}}{\prod_{i=n-m+1}^{n} \sigma_3^{t_i' m_{j,i}^{*}{}'}}, \sigma_2^* = \sigma_2^{*\prime}, \sigma_3^* = \sigma_3^{*\prime}$$

6 Comparison

We compare our scheme against previous schemes on sanitizable signatures.

Scheme	Transparency	Security	Model
[1]	No Transparency	RSA	ROM
[14]	No Transparency	underlying signature	standard
[15]	No Transparency	underlying signature	standard
[5]	No Transparency	underlying signature and commitment	standard
[6]	No Transparency	co-GDH	ROM
[8]	No Transparency	co-GDH	ROM
[7]	No Transparency	strong RSA	standard
[16]	No Transparency	underlying signature commitment and pseudo random generator	standard
[9]	No Transparency	CDH + XDH	standard
[2]	Weak Transparency	underlying signature and chameleon hash	standard
[4]	Weak Transparency	CDH	ROM
[3]	Strong Transparency	-	-
Our Scheme	Strong Transparency	CDH	standard

7 Conclusion and Open Problems

In this paper, we proposed the first provably secure sanitizable signature protocol having strong transparency property under standard model. These signatures are of constant length, and shorter than most other protocols. In earlier schemes, such as [3], which claim strong transparency, either there is no formal proof provided or the proof is under the random oracle model. An interesting open problem is to devise a protocol which can achieve strong transparency without dividing the message into bits or blocks. The problem of using more traditional techniques such as RSA, rather than pairings to provide more efficient sanitizable signatures with strong transparency is open. Accountability is a property of sanitizable signatures by which the signer can prove that a particular signature is his, and not by the sanitizer. In our schemes, accountability is not provided as this compromises unconditional indistinguishabilty. However, an interesting open problem would be to formulate a sanitizable signature scheme with strong transparency that offers polynomial time indistinguishability as well as accountability.

References

1. Bull, L., Stański, P., Squire, D.: Content extraction signatures using xml digital signatures and custom transforms on-demand. In: WWW, pp. 170–177 (2003)

2. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
3. Klonowski, M., Lauks, A.: Extended sanitizable signatures. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 343–355. Springer, Heidelberg (2006)
4. Miyazaki, K., Hanaoka, G., Imai, H.: Digitally signed document sanitizing scheme based on bilinear maps. In: ASIACCS, pp. 343–354 (2006)
5. Miyazaki, K., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H., Tezuka, S., Imai, H.: Digitally signed document sanitizing scheme with disclosure condition control. IEICE Transactions 88-A(1), 239–246 (2005)
6. Suzuki, M., Isshiki, T., Tanaka, K.: Sanitizable signature with secret information. In: Research Reports on Mathematical and Computing Sciences, pp. 114–127. Springer, Heidelberg (2005)
7. Chang, E.C., Lim, C.L., Xu, J.: Short redactable signatures using random trees. In: CT-RSA, pp. 133–147 (2009)
8. Izu, T., Kunihiro, N., Ohta, K., Takenaka, M., Yoshioka, T.: A sanitizable signature scheme with aggregation. In: Dawson, E., Wong, D.S. (eds.) ISPEC 2007. LNCS, vol. 4464, pp. 51–64. Springer, Heidelberg (2007)
9. Yuen, T.H., Susilo, W., Liu, J.K., Mu, Y.: Sanitizable signatures revisited. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 80–97. Springer, Heidelberg (2008)
10. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
11. Klonowski, M., Lauks, A.: Extended sanitizable signatures. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 343–355. Springer, Heidelberg (2006)
12. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of sanitizable signatures revisited. In: Irvine: Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography, pp. 317–336. Springer, Heidelberg (2009)
13. Agrawal, S., Kumar, S., Shareef, A., Rangan, C.P.: Sanitizable signatures with strong transparency in the standard model (full version). Cryptology ePrint Archive (2010), `http://eprint.iacr.org/`.
14. Johnson, R., Molnar, D., Song, D.X., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
15. Kunihiko, M., Susaki Seiichi, I.M.: Digital document sanitizing problem. In: CIEIC Technical Report (Institute of Electronics, Information and Communication Engineers), pp. 61–67 (2003)
16. Haber, S., Hatano, Y., Honda, Y., Horne, W., Miyazaki, K., Sander, T., Tezoku, S., Yao, D.: Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In: ASIACCS 2008: Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, pp. 353–362. ACM, New York (2008)
17. 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)
18. Izu, T., Kanaya, N., Takenaka, M., Yoshioka, T.: Piats: A partially sanitizable signature scheme. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 72–83. Springer, Heidelberg (2005)

Breaking and Building of Threshold Signcryption Schemes

S. Sharmila Deva Selvi[1], S. Sree Vivek[1,⋆], Shilpi Nayak[2], and C. Pandu Rangan[1,*]

[1] Theoretical Computer Science Lab, Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India

[2] National Institute of Technology Bhopal, Bhopal, India

Abstract. Signcryption is a cryptographic primitive that fulfills both the functions of digital signature and public key encryption in a single step, at a cost significantly lower than that is required by the traditional signature-then-encrypt approach. Threshold involved with signcryption is used where the sender or receiver side has a group of members. Threshold Signcryption comes into picture when a message to be sent needs the authentication of a particular number of members in an organization, whereas threshold unsigncryption is used where until a given number of members join, a particular message cannot be decrypted. In this paper we show that three of the threshold signcryption schemes reported in the literature, lack confidentiality under the stronger notion of insider security. We also propose an improved scheme and give the formal proof of security in new stronger security model.

1 Introduction

Signcryption, introduced by Zheng in 1997 [14], is a cryptographic primitive that offers confidentiality and authentication simultaneously, but with lesser computational complexity and lower communication cost. This has made signcryption a suitable primitive for applications that require secure and authenticated message delivery, where devices have limited resources. After Zheng's work a number of signcryption schemes were proposed [4] [13] [12] [9] [5] [6]. The security notion for signcryption was first formally defined in 2002 by Baek et al. in [2]. This was similar to the notion of semantic security against adaptive chosen cipher text attack and existential unforgeability against adaptive chosen message attack.

Desmedt introduced the concept of threshold cryptography in 1987 [7]. Extending this concept, a (t, n) threshold signature scheme based on the RSA system [11] was first proposed by Desmedt and Frankel in 1991[8]. In such a

⋆ Work supported by the project PROTOCOLS FOR SECURE COMMUNICATION AND COMPUTATION- CSE/05-06/076/DITX/CPAN.

(t, n) threshold signature scheme, any t out of n signers in the group can collaboratively sign messages on behalf of the group by sharing the signing capability. This can be visualized in the situation where a company has n directors and if only at least t of them agree on a decision, then that decision is finalized. A threshold signcryption scheme blends the concept of threshold cryptosystem with the basic signcryption concept.

In this paper, we show three such schemes involving threshold lack confidentiality. The first one is the scheme by Peng et al. [10]. This scheme perfectly integrates digital signature with public key encryption along with the benefits of elliptic curve cryptosystem and takes care of the cheating of trusted dealer and the cheating of participants each other, but it lacks confidentiality under the stronger notion of insider security.

The second one is the scheme in [1] proposed by Aihong et al. which involves the concept of subliminal channel. A subliminal channel is used to send a private message to an authorized receiver, and the message cannot be discovered by any unauthorized receivers. We show the lack of confidentiality in this scheme.

The third one is the scheme by Zhang et al. [15] which involves threshold at the receiver's side. This scheme deals with the drawbacks of few of the previously known schemes, but still lacks confidentiality.

Finally, we propose an improvement for the scheme reported in [10] and prove formally that our new scheme is secure even in a stronger model.

2 PKI Based Threshold Signcryption

2.1 General Framework of PKI Based Threshold Signcryption

In a PKI based threshold signcryption scheme, a trusted CA generates the system parameters and each user generates his/her own private/public keys depending on the systems parameters. On behalf of every group, the CA generates a group public key and the group secret key. The CA generates the shares of the group secret key and distributes one share to each member of the group. For a given message, each group member generates the signcryption share using the secret key share that was given to him by the CA. In a t -threshold signcryption scheme, any t signcryption shares may be combined to generate the signcryption of the corresponding message. The details are discussed below. For a PKI based system the key generation is done by the user using the system public parameters and the secret value is chosen by the user. A PKI based threshold signcryption scheme consists of the following polynomial time algorithms.

- ***Initialization*** (κ): *Given a security parameter κ, the trusted center CA generates and publishes system's public parameters.*
- ***User Key generation***: This procedure is executed by each user taking the systems parameter as input, a public/private key pair is generated.
- ***Key Distribution*** (G_A, x_A, n_A, t_A): *Given the private key x_A of the group G_A, the number of group members n ($\{A_1, A_2, \ldots, A_n\}$) and t - the threshold value of the group G_A (i.e., the number of maximum collisions permitted in*

the group, beyond which the private key of the group is not secure), CA runs this algorithm to compute the private key shares x_{A_i} and the corresponding public keys y_{A_i} of all n members of the group. Then each (x_{A_i}, y_{A_i}) is sent to A_i. Note that A_i can have his own public key/private key pair besides the pair he receives as a member of a group.

Signcryption $(\ m, y_A, \{A_1, A_2, ..., A_t\}, y_B\ )$: *For generating a signcryption of a message m from sender group G_A with public key y_A to the receiver B with public key y_B, at least t of the group members must agree and run this algorithm to generate the signcryption shares. Then, one of the group members or a designated semi trusted authority called clerk verifies each share and generates the signcryption σ.*

Unsigncryption $(\ \sigma, y_A, y_B, x_B)$: *This algorithm is run by the receiver B with public key y_B and private key x_B by providing the signcryption σ and public key y_A of the sender G_A. The unsigncryption algorithm outputs m if σ is a valid signcryption from G_A to B. Otherwise outputs "Invalid". In case of threshold unsigncryption schemes, the members of the receiver group collaboratively perform the unsigncryption of σ.*

2.2 Security Model for PKI-Based Threshold Signcryption

The formal security of signcryption scheme was first proposed by *Baek* and *Zheng* in [3]. The security model includes two notions: the security against chosen ciphertext attacks which is also called semantic security and existential unforgeability against chosen-message attacks. In a signcryption,only one sender and one receiver is involved. However,in threshold schemes, either sender side or the receiving side may have more than one user and this calls for extending the definition of security models in signcryption to security models in threshold signcryption.

Confidentiality A PKI-based Threshold Signcryption Scheme (TSC) is semantically secure against adaptive chosen ciphertext attack (IND-TSC-aCCA2), if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage in the following game between Challenger $\mathcal{C}$ and Adversary $\mathcal{A}$.

Initialization : *The challenger $\mathcal{C}$ runs the **Initialization** algorithm with the security parameter κ as input, to generate the public parameters params of the system. $\mathcal{C}$ also generates a list of public, private key pairs and provides it to the adversary $\mathcal{A}$ along with params. $\mathcal{C}$ also provides $\mathcal{A}$ a target user A^* with public key y_{A^*} with which $\mathcal{A}$ will be challenged during the challengephase. $\mathcal{A}$ is not provided with the private key x_{A^*} of A^*.*

Phase 1: *In this phase $\mathcal{A}$ performs a series of queries in an adaptive fashion. The following are the oracles provided by $\mathcal{C}$ to $\mathcal{A}$:*

Signcrypt Oracle : *When $\mathcal{A}$ queries this oracle with message m, the sender group G_A and the receiver public key y_B, $\mathcal{C}$ returns the signcryption σ.*

Signcrypt Share Oracle : *When $\mathcal{A}$ queries this oracle with message m, the sender group G_A with public key y_A, t members of G_A and the receiver*

B with public key y_B, $\mathcal{C}$ returns the t signcryption shares to $\mathcal{A}$. To the best of our knowledge this oracle was not considered earlier in threshold signcryption schemes.

Unsigncrypt Oracle : *When $\mathcal{A}$ makes a query by submitting the signcryption σ, public key y_A of sender group G_A and public key y_B of receiver B, $\mathcal{C}$ responds with the message m, if σ is a valid signcryption of m from G_A to B. Otherwise returns "Invalid".*

Challenge : *At the end of Phase 1, $\mathcal{A}$ sends to $\mathcal{C}$, two plaintexts m_0 and m_1 of equal length and public key y_A (sender group G_A) and the public key y_{A^*} of the receiver A^*, on which $\mathcal{A}$ wishes to be challenged. The challenger $\mathcal{C}$ chooses a bit $b^* \in_R \{0,1\}$ and computes the challenge signcryption σ^* on the message m_{b^*} from G_A to y_{A^*} and returns σ^* to $\mathcal{A}$.*

Phase 2: *In this phase $\mathcal{A}$ can adaptively perform polynomially bounded number of queries again as in Phase 1 but it should not query for the unsigncryption of σ^* .*

The advantage of $\mathcal{A}$ is defined as $Adv(\mathcal{A}) = \mid 2P[b' = b^*] - 1 \mid$
Here, the adversary $\mathcal{A}$ is allowed to know the private key of the sender group used for challenge, i.e, A. This is to capture the notion of insider security.

Existential Unforgeability A PKI-based Threshold Signcryption Scheme (TSC) is said to be secure against an existential forgery for adaptive chosen messages attacks (EUF-TSC-aCMA) if no polynomially bounded adversary has a non-negligible advantage in the following game:

Initial : *The challenger $\mathcal{C}$ runs the **Initialization** with the security parameter κ as input, to generate the public parameters params. $\mathcal{C}$ generates a list of public, private key pairs and gives the public parameters and the list of public, private key pairs to $\mathcal{A}$. $\mathcal{C}$ also chooses a sender group G_{A^*} with public key y_{A^*} and gives y_{A^*} to $\mathcal{A}$. $\mathcal{A}$ is not provided with the private key x_{A^*} corresponding to the group G_{A^*}. But $\mathcal{A}$ is provided with $t-1$ secret key shares of A^*.*

Training Phase: *$\mathcal{A}$ makes polynomially bounded number of queries adaptively to the various oracles provided by $\mathcal{C}$, as described in Phase 1 of the confidentiality game.*

Forgery: *At the end of the Training Phase, $\mathcal{A}$ produces a signcryption σ^* on some message m^* with G_{A^*} with public key y_{A^*} as sender group and B with public key y_B as receiver, such that the triple (σ^*, y_A, y_B) was not the output of any previous queries to the Signcrypt Oracle. $\mathcal{A}$ wins the game if σ^* is a valid signcryption on message m^* with G_{A^*} as sender group and B as receiver.*

3 Threshold Signcryption Scheme by Peng et al.[10]

In this section, we review the threshold signcryption scheme based on elliptic curve cryptosystem proposed by Peng et al. [10]. We show that, [10] is not secure against CPA attack and is existentially forgeable. We also provide an improvement for [10] and provide the formal proof of security for the improved scheme.

3.1 Review of the Scheme[10]

The scheme by Peng et al. involves four roles : A trusted center CA who is responsible for generating system parameters, the sender group $G_A = \{A_1, A_2, ..., A_n\}$, the clerk C selected randomly from the group who collects the signcryption shares and generates the group signcryption and the receiver B. It consist of following algorithms:

- ***Parameters Choosing Phase :***
  CA chooses a secure elliptic curve $E(\mathbb{F}_p)$ over finite field $\mathbb{F}_p$ and a base point P on it whose order of q, where q is a large prime. G_A is a group having set of n member who can participate in the generation of the signcryption share for G_A. From the definition of threshold scheme, any t out of n signers $1 \leq t \leq n$ can represent the group G_A to generate the full signcryption of G_A. CA chooses a random integer $x_A \in [1, q-1]$ as a private key of group G_A, the corresponding public key is $Y_A = x_A P$.
- ***Verifiable Secret Key Split Phase :*** The private key x_A of group A will be distributed to A_i $(1 \leq i \leq n)$.
  - *Step 1:* The trusted dealer CA randomly generates a secret polynomial
  $$f(x) \ = \ a_0 + a_1 x + ... + a_{t-1} x^{t-1} \bmod q$$
  over Z_q of degree $t-1$ satisfying $a_0 = f(0) = x_A$. Then, CA computes $x_{A_i} = f(i)$ as private key of $A_i (1 \leq i \leq n)$, thus the corresponding public key is $Y_{A_i} = x_{A_i} P$. Finally, CA publishes Y_{A_i}.
  - *Step 2:* CA sends x_{A_i} secretly to A_i $(1 \leq i \leq n)$ and broadcasts $a_j P$ $(0 \leq j \leq t-1)$ to all n signers. This verification is given by.
  $$x_{A_i}.P \overset{?}{=} \sum_{j=0}^{t-1} i^j (a_j.P)$$
  That is, each signer A_i $(1 \leq i \leq n)$ may use above check to verify whether his private key x_{A_i} from CA is correct or not. If this check holds, the share x_{A_i} is accepted, otherwise rejected.
- ***Threshold Signcryption Phase*** : Suppose the message $m \in [1, p-1]$ will be signcrypted by any t participants from group A for the receiver B. Without loss of generality, let $A_1, A_2, ..., A_t$ are the t participants. In this phase, the group signcryption (c, s) will be generated. This phase includes four steps:
  - *Step 1 :* Each signer A_i $(1 \leq i \leq t)$ chooses a random integer $k_i \in [1, q-1]$, then computes $V_i = k_i P$ and sends it to clerk C and receiver B via public channel, computes $Z_i = k_i Y_B$ and sends it secretly to clerk C.

- *Step 2 :* Signcryption clerk C computes

  $$Z = \sum_{i=0}^{t} Z_i = \sum_{i=0}^{t} k_i.Y_B = k.Y_B \text{ and}$$
  $$c \ = \ m.(Z)_x \bmod p,$$

  where $(Z)_x$ is the x co-ordinate of Z, $k = \sum_{i=1}^{t} k_i$. Clerk broadcasts c to each signer A_i $(1 \leq i \leq t)$.

- *Step 3 :* Each signer A_i $(1 \leq i \leq t)$ computes
  - $*$ $l_i = \prod_{j=1, j\neq i}^{t} \frac{-j}{i-j} \bmod q$
  - $*$ $e_i \ = \ l_i.x_{A_i} \bmod q$,
  - $*$ $s_i \ = \ k_i - e_i.c \bmod q$

  and then sends the partial signcryption share s_i to clerk C.

- *Step 4 :* After receiving the partial signcryption shares s_i, $i = 1\,to\,t$, the clerk C computes $V_i' = c.l_i.Y_i + s_i.P$, and then verifies validity of partial signcryption share s_i using $V_i = V_i'$ If this equation holds, s_i is valid, otherwise, is invalid. If all the partial signcryption shares are valid, then C computes s by $s \ = \sum_{i=0}^{t} s_i \bmod q$

  Finally, $\sigma =(c, s)$ is the signcryption from group G_A to B.

– ***Verification and Message Recovery Phase :*** On receiving the signcryption (c, s), B can verify its validity using public key Y_A of group A and recover the message m using his private key x_B by following steps:
- *Step 1 :* Computes

  $$V = \sum_{i=1}^{t} Y_i = \sum_{i=1}^{t} k_i.P = k.P \ ; \ \ V' \ = \ c.Y_A + s.P$$

- *Step 2:* Verifies whether $V \overset{?}{=} V'$ is correct. If this holds, the signcryption (c, s) is valid, otherwise is invalid.
- *Step 3:* Recovers the message by $m \ = \ c.(Z')_x^{-1} \bmod p$, and checks its validity from redundant information of m.

3.2 Weakness of the Scheme

In the above scheme the clerk can ask signature on any message m since the signers cannot verify whether c is a valid signcryption of m as they do not have the key $(Z)_x$ which is used to encrypt $(c \ = \ m.(Z)_x \bmod p)$. The signers send their shares of key Z i.e,Z_i secretly to the clerk with which clerk generates the private key and encrypts the message. So no one else knows the key and hence cannot verify the validity of ciphertext.

3.3 Attack on the Scheme[10]

Attack on Unforgeability: The scheme in [10] is existentially forgeable. The attack is shown below:

- During the training phase of unforgeability game, $\mathcal{A}$ queries for the signcryption of m^* from sender group A^* to a receiver B.
- Let this signcryption be $\sigma = (c, s)$.
- $\mathcal{A}$ submits $\sigma^* = \sigma$ as a forgery from sender group A^* to any receiver B'.
- Here σ^* is a valid signcryption on $m' = cky_{B'}$.
- The signcryption σ^* is valid, since no components in σ are altered except the receiver of the signcryption.

Hence, the scheme [10] is existentially forgeable.
Attack on Confidentiality: During the confidentiality game, the challenger $\mathcal{C}$ provides the challenge signcryption $\sigma^* = (c = m_{b'}(Z)_x, s = k - x_A c)$ (signcryption of $m_{b'}$ from sender group G_A to receiver A^*) to the adversary $\mathcal{A}$. Here, $\mathcal{A}$ knows that σ^* is the signcryption of either m_0 or m_1. Also, $\mathcal{A}$ knows the private key x_A of the sender group G_A. Now $\mathcal{A}$ can decrypt the challenge signcryption σ^* by doing the computations given below:

- $k = s + x_A.c$, where x_A is the private key of the sender known to $\mathcal{A}$
- $Z = k.y_{A^*}$
- $m_{b^*} = c.(Z)_x$, where $(Z)_x$ is the x co-ordinate of Z.

Hence $\mathcal{A}$ can easily distinguish whether σ^* is the signcryption of the message m_0 or m_1 during the confidentiality game.

Remark : This attack is possible because unsigncryption key can be easily extracted using the sender's private key. Hence the ciphertext can be decrypted if the private key of sender is known. The above attack also holds in the signcryption scheme proposed by them in [10] where there is single sender.

3.4 The Improved Scheme

The improved threshold signcryption scheme involves the following roles : A trusted center CA who is responsible for generating parameters, the sender group $G_A = \{A_1, A_2, ..., A_n\}$, the clerk C selected randomly from the group who collects the signcrypted shares and generates the group signcryption and the message receiver B. It consist of following five algorithms:

1. ***Initialization*** :CA chooses a secure elliptic curve $E(\mathbb{F}_p)$ over finite field $\mathbb{F}_p$ and a base point P on it which has an order of q, where q is a large prime. CA chooses hash functions $H_1 : \{0,1\}^n \times \mathbb{F}_q \rightarrow \mathbb{F}_p, H_2 : \mathbb{F}_p \times \mathbb{F}_p \times \mathbb{F}_p \times \mathbb{F}_p \rightarrow \{0,1\}^n, H_3 : \{0,1\}^n \times \mathbb{F}_p \times \mathbb{F}_p \times \mathbb{F}_p \rightarrow \mathbb{F}_p$. Finally, CA publishes the system public parameters $p, q, E(\mathbb{F}_p), P, H_1, H_2, H_3$. CA chooses a random integer $x_A \in [1, q-1]$ as the private key of group G_A, thus the corresponding public key is $Y_A = x_A P$. Let B be any receiver whose public key $Y_B = x_B P$ and the corresponding private key is x_B(chosen by B). $\{A_i\}_{i=1\,to\,n}$ is a set of n signers of G_A. From the definition of threshold scheme, any t out of n

signers $1 \leq t \leq n$ can represent the group G_A and perform the signcryption by contributing their signcryption shares.

2. ***Key Distribution*** *:* This algorithm is same as the $Verifiable\, Secret\, Key\, Split\, Phase$ algorithm of the original scheme [10].
3. ***Signcryption*** *:* Suppose the message $m \in [1, p-1]$ will be signcrypted by any t participants from group G_A for the receiver B. Without loss of generality, let $A_1, A_2, ..., A_t$ are the t participants. In this phase, the signcryption (c, s, W, V) will be generated by performing the following steps:
   - *Step 1 :* Each signer A_i $(1 \leq i \leq t)$ chooses a random integer $x_i \in [1, q-1]$, then computes $W_i = x_i P$ and sends it to clerk C.
   - *Step 2 :* Signcryption clerk C chooses random integer $r \in [1, q-1]$ and computes,
     - $k = H_1(m, r)$
     - $V = k.P$
     - $Z = k.Y_B$
     - $W = \sum_{i=1}^{t} W_i = \sum_{i=1}^{t} x_i.P = x.P$, where $x = \sum_{i=1}^{t} x_i$
     - $\alpha = H_2(Z, V, W, Y_A, Y_B)$
     - $c = E_\alpha(m \| r)$
     - $h = H_3(c, V, W, Y_A, Y_B)$

     C then broadcasts W, r to each signer A_i $(1 \leq i \leq t)$.
   - *Step 3 :* Each signer A_i $(1 \leq i \leq t)$ computes,
     - $k = H_1(m, r)$
     - $V = k.P$; $Z = k.Y_B$
     - $\alpha = H_2(Z, V, W, Y_A, Y_B)$
     - $c = E_\alpha(m \| r)$
     - $h = H_3(c, V, W, Y_A, Y_B)$

     Each A_i then generates the signcryption share as follows:
     - $l_i = \prod_{j=1, j \neq i}^{t} \frac{-j}{i-j} \bmod q$
     - $e_i \;=\; x_{A_i}.l_i \bmod q$
     - $s_i \;=\; x_i - e_i.h \bmod q$

     and then sends the partial signcryption share s_i to clerk C.
   - *Step 4 :* After clerk C receives the partial signcryption share s_i, he first computes $W_i' = h.l_i.y_{A_i} + s_i.P$, and then verifies validity of partial signcryption share s_i by,

     $$W_i \stackrel{?}{=} W_i'$$

     If this check holds, then s_i is valid, otherwise s_i is invalid. If all the signcryption shares are valid, then C computes $s = \sum_{i=1}^{t} s_i \bmod q$ Finally, (c, s, W, V) is the signcryption of message m from group G_A to receiver B.
4. ***Sign Verification*** *:* On receiving the signcryption (c, s, V, W), B computes,
   - $h = H_3(c, V, W, Y_A, Y_B)$.
   - $W' = h.Y_A + s.P$

   Verifies whether $W \stackrel{?}{=} W'$ is correct. If this check holds, the signcryption (c, s, V, W) is valid, otherwise is invalid.

5. ***Unsigncryption:*** If **SignVerification**(c, s, V, W) = "$Valid$" then Receiver B computes,
   - $Z = x_B.V = x_B.(kP)$.
   - $\alpha = H_2(Z, V, W, Y_A, Y_B)$.
   - $m \| r = D_\alpha(c)$.

   and checks the validity of signcryption by verifying whether $V \overset{?}{=} H_1(m, r)P$.

Remark: In the improved scheme we have addressed all the drawbacks present in [10]. We have also proved the security of the improved scheme against the stronger notion of security for signcryption called the insider security (for both confidentiality and unforgeability).

3.5 Security Proof of the Improved Scheme

The improved scheme is proved in random oracle model for both confidentiality and unforgeability in the security model given in section 2.2.

Confidentiality:

Theorem 1. *The Improved Threshold Signcryption Scheme of [10] is secure against any (IND-TSC-aCCA2) adversary, if CDH problem is hard in the group* $E(\mathbb{F}_p)$

Proof : Let $\mathcal{C}$ be a challenger, who is challenged with an instance of CDH problem say, $(P, aP, bP) \in \mathbb{F}_p$ for unknown $a, b \in \mathbb{F}_q$. The aim of $\mathcal{C}$ is to calculate abP for the given CDH instance. $\mathcal{C}$ uses an adversary $\mathcal{A}$ who is capable of breaking the IND-TSC-CCA2 security of *ImprovedThreshold Signcryption* scheme to solve the CDH problem instance with non-negligible advantage in polynomial time as described below:
$\mathcal{A}$ may ask for the Signcryption oracle $\mathcal{O}_{Signcryption}$ or the Signcryption share oracle $\mathcal{O}_{Signcryption-Share}$ for any message.

Initial : Suppose there are u players in the system, the challenger $\mathcal{C}$ picks any one of them $i \in_R [1, 2, ..., u]$. It sets $y_{A^*} = aP$, $A^* \in \{1, \ldots, u\}$. Also, $\mathcal{C}$ generates public key, private key pairs $(y_i = x_iP, x_i)$ of the $(u-1)$ users other than A^*. $\mathcal{C}$ gives $\mathcal{A}$ the public parameters $p, q, E(\mathbb{F}_p), P, H_1, H_2, H_3, E$, D, y_{A^*}, $\{y_i, x_i\}_{(i=1\, to\, u and i \neq A^*)}$. Also, $\mathcal{C}$ generates a list of group public and private keys and provides it to $\mathcal{A}$.
Phase 1 : The adversary $\mathcal{A}$ can ask queries to the random oracles $H_1, H_2\; and\; H_3$. As these answers are randomly generated, $\mathcal{C}$ keeps the lists $L_1, L_2\; and\; L_3$ to maintain consistency and to avoid collision. The queries to the random oracles $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$, $\mathcal{O}_{H_3}$, $\mathcal{O}_{Signcrypt}$, $\mathcal{O}_{Unsigncrypt}$,
$\mathcal{O}_{Signcrypt-Share}$ are answered as follows :

- **Oracle** $\mathcal{O}_{H_1}(m, r)$ **:** For a query $H_1(m, r)$ by $\mathcal{A}$, if there exists a tuple (m, r, k) then $\mathcal{C}$ returns k as the answer to $\mathcal{A}$ else chooses $k \in_R \mathbb{F}_p$, returns k to $\mathcal{A}$ and updates the list L_1 with the tuple (m, r, k).

- **Oracle** $\mathcal{O}_{H_2}(Z, V, W, y_A, y_B)$ **:** For a query $H_2(Z, V, W, y_A, y_B)$, if there exists a tuple $(Z, V, W, y_A, y_B, \alpha)$, then $\mathcal{C}$ returns α as the response to $\mathcal{A}$, else chooses $\alpha \in_R \{0,1\}^n$ such that no other tuple contains the same α and returns to $\mathcal{A}$. It then updates the list L_2 with the tuple $(Z, V, W, y_A, y_B, \alpha)$.
- **Oracle** $\mathcal{O}_{H_3}(c, V, W, y_A, y_B)$ **:** For a query $H_3(c, V, W, Y_A, Y_B)$ by $\mathcal{A}$, if there exists a tuple (c, V, W, y_A, y_B, h) then $\mathcal{C}$ returns h as the answer to $\mathcal{A}$, else chooses $h \in_R \mathbb{F}_p$ such that no other tuple contains the same h and returns h to $\mathcal{A}$. It then updates the list L_3 with the tuple (c, V, W, Y_A, Y_B, h).
- **Oracle** $\mathcal{O}_{Signcrypt}(m, y_A, y_B)$ **:** For a signcryption query by $\mathcal{A}$ on message m from group G_A with public key y_A to user B with public key y_B, the challenger $\mathcal{C}$ computes the signcryption as follows :
  - If $y_A \neq y_{A^*}$, this also includes the case when $y_B = y_{A^*}$. In this case, $\mathcal{C}$ knows the private key of sender and can respond by running the algorithm **Signcrypt**(m, x_A, y_B). and updates the lists as
    - puts (m, r, k) into L_1, $(Z, V, W, y_A, y_B, \alpha)$ into L_2 and (c, V, W, y_A, y_B, h) into L_3
  - If $y_A = y_{A^*}$, $\mathcal{C}$ simulates Signcryption as:
    $\mathcal{C}$ randomly chooses s *and* $r \in_R \mathbb{F}_q$ and $h \in_R \mathbb{F}_p$, then computes:
    - $k = H_1(m, r)$
    - $V = k.P$
    - $W = h.y_B + s.P$
    - $Z = x_B.V$
    - $\alpha = H_2(Z_i, V_i, W_i, y_A, y_B)$
    - $c = E_\alpha(m \| r)$. Now $\mathcal{C}$ updates the entries in L_2 after ensuring that the tuple (c, V, W, y_A, y_B, h') does not exist such that $h' \neq h$, else repeat with different random values.

    $\mathcal{C}$ then returns to $\mathcal{A}$ ciphertext $\sigma = (c, s, W, V)$.
- **Oracle** $\mathcal{O}_{Signcrypt-Share}(m, y_A, y_B)$ **:** The signcryption share oracle is queried by the adversary when it needs the t shares of the members of G_A. Here the challenger knows the private key of all groups and can follow the protocol to provide the t signcryption shares.
- **Oracle** $\mathcal{O}_{Unsigncrypt}(\sigma, y_A, y_B)$ **:** For the unsigncryption query of (c, s, W, V) with G_A as sender and B as receiver, the challenger first verifies the signcryption by running the *Sign Verification* algorithm, if verification check passes then $\mathcal{C}$ unsigncrypts the message as follows :
  - If $y_B \neq y_{A^*}$, $\mathcal{C}$ knows the private key of the receiver, so $\mathcal{C}$ can unsigncrypt σ by running the *Unsigncrypt* algorithm and returns m as $m \parallel r = D_\alpha(c)$, only if $V = (H_1(m, r))P$, otherwise returns *Invalid* .
  - If $y_B = y_{A^*}$, then $\mathcal{C}$ searches L_2 for a tuple $(*, V, W, y_A, y_B, \alpha)$, if such a tuple exists then it retrieves corresponding α, and decrypts c to obtain $m \parallel r = D_\alpha(c)$, and checks if $V = (H_1(m, r))P$, if this does not hold then continue the above procedure with another such tuple in list L_2, if such a tuple passes the validity then return m, else return *Invalid.*

Challenge : After the end of phase one, $\mathcal{A}$ outputs two messages m_0 and m_1 of equal length, and a sender group G_A and the target user A^* with public key y_{A^*} as receiver. Given these values $\mathcal{C}$ gives the challenge ciphertext σ^* as :
$\mathcal{C}$ chooses $b^* \in_R \{0,1\}$, and returns the signcryption of m_{b^*} as follows:

- chooses $W \in_R \mathbb{Z}_p^*$, $r \in_R \mathbb{Z}_q^*$,
- sets $V = bP$ (From the CDH instance).
- chooses $c \in_R \{0,1\}^n$.
- $h = H_3(c, V, W, y_A, y_{A^*})$.
- $s = W - x_A.h$.

$\mathcal{C}$ then returns to $\mathcal{A}$ the challenge ciphertext $\sigma = (c, s, W, V)$
Phase 2 : Now $\mathcal{A}$ performs second series of queries to the oracles treated in the same way as in the first phase.
The challenge ciphertext given to the adversary is not a valid one. $\mathcal{A}$ can find that σ^* is invalid after unsigncrypting the signcryption. Particularly $\mathcal{A}$ will know that σ^* is invalid after querying H_3 oracle with $Z = abP$. This entry will get captured in list L_3. Now, $\mathcal{C}$ picks randomly a Z from list L_3 and submits as solution to the CDH problem instance. If ϵ is the advantage of the adversary then $\mathcal{C}$ wins the game with probability $\epsilon' \geq \frac{\epsilon}{q_3}$, where, q_3 is the number of tuples in the L_3 list. □

Existentially Unforgeability Proof of I-TSC

Theorem 2. *In the random oracle model, if there exists a forger $\mathcal{F}$ that can break the EUF-TSC-aCMA unforgeability of I-TSC, then there exists an algorithm $\mathcal{C}$ which can solve the DLP problem.*

The proof for this theorem is given in the full version of the paper.

4 Threshold Signcryption Scheme by Aihong et al. [1]:

In this section, we review Publicly verifiable Hybrid Signcryption(PVHS) scheme [1] proposed by Aihong et al. We show that the scheme in [1] is not CPA secure.

4.1 Review of the Scheme [1]

This scheme proposed by Aihong et al. in [1] involves the following algorithms:

1. ***System Initialization Phase :*** CA selects two large prime numbers p and q that satisfy $q|(p-1)$, a generator g with order q in $\mathbb{Z}_p^*$ which generates a subgroup $< g >$. CA also chooses a secure symmetric encryption system (E, D), hash functions $H_1 : \mathbb{Z}_p^* \rightarrow \{0,1\}^n, H_2 : \mathbb{Z}_p^* \times \{0,1\}^n \rightarrow \mathbb{Z}_q^*, H_3 : GF(P) \rightarrow \mathbb{Z}_q, H_4 : \{0,1\}^{(n+1)|q|} \rightarrow GF(P)$. Finally, CA publishes system's public parameters $\{p, q, g, H_1, H_2, E, D\}$. As in PKI based systems, any user A can choose a random integer $x_A \in \mathbb{Z}_q^*$ as private key and computes the public key $y_A = g^{x_A} \bmod p$. Similarly, for each group G_B with public key y_B an private key x_B. Group G_B has members $\{B_i\}_{i=1\,to\,n}$ and the private key of each member is $x_{B_i} \in \mathbb{Z}_q^*$ and the corresponding public key $y_i = g^{x_i}$ mod p.

2. ***Share Generation Phase :*** Let $m_s \in GF(P)$ be the subliminal message that the sender A would like to send to $B_1, B_2, ..., B_n$.
   - A choose a $(t-1)$ degree polynomial in $GF(P)$, and computes secret shadows $m_i = f(i)$ for all B_i's $(1 \leq i \leq n)$ $f(x) \ = \ m_s + a_1 x + ... + a_{t-1}x^{t-1} \bmod q$
   - A computes $v = f(1) \bmod q$ and $R = H_3(m_s \oplus v) \bmod q$, he publishes B.

3. ***Signcryption Generation Phase :*** For sending a message m from user A with public key y_A to a group G_B with public key y_B, the user A
   - A selects a random number $k_i \in \mathbb{Z}_p^*$ and computes the signcryption for every user B_i $(1 \leq i \leq n)$ through the following equations:
     - $K_i \ = \ g^{k_i} \bmod p$
     - $T_i \ = \ H_1(y_i^{k_i}) \bmod p$
     - $c \ = \ E_{T_i}(m_i)$
     - $r_i \ = \ H_2(K_i, c_i \bmod p)$
     - $s_i \ = \ k_i - x_a.r_i$

     The signcryption for B_i is (c_i, r_i, s_i)
   - A computes $c \ = \ H_4(R \parallel r_1 \parallel r_2 \parallel .... \parallel r_3)$
   - A publishes the signcryption $\sigma \ = \ (R, c, (c_1, r_1, s_1), \ (c_2, r_2, s_2)....., (c_n, r_n, s_n))$.

4. ***Signcryption Verification Phase :*** The signcryption is publicly verifiable because anyone can verify using the given equation $c \ = \ H_4(R \parallel H_2(g^{s_1}{y_A}^{r_1} mod p, c_1) \parallel H_2(g^{s_2}{y_A}^{r_2} mod p, c_2) \parallel .... \parallel H_2(g^{s_n}{y_A}^{r_n} mod p, c_n))$

5. ***Message Recovery and Verification Phase:*** Every user can get his partial signcryption share from σ which is (c_i, r_i, s_i) $(1 \leq i \leq n)$. They can get the share and reconstruct the subliminal message by co-operation Each user B_i retrieves the shadow m_i by the following equations
   - $K_i' = g^{s_i}{y_A}^{r_i} \ mod \ p$
   - $T_i' = H_1(K_i'^{x_i} \ mod \ p)$
   - $m_i = D_{T_i'}(c)$

     Any t users, say $B_1, B2, ..., B_t$ can cooperate to reconstruct the polynomial $f(x)$ in $GF(P)$ as follows:
   - $f(x) = \prod_{j=1, j \neq i}^{t} m_i \frac{x-j}{i-j} \ mod \ q$

     Recover the subliminal message $m_s = f(0)$. They computes $v = f(1)$, and verify whether $R = H_3(m_s \oplus v)$, if the equation is right, they accept the secret.

4.2 Attack on the Scheme citePing

The above scheme is CPA secure in the insider security model. The attack is as follows :

Attack on Confidentiality : During the confidentiality game, the adversary $\mathcal{A}$ is given ciphertext $\sigma = (R, c, (c_1, r_1, s_1), (c_2, r_2, s_2)....., (c_n, r_n, s_n))$ as a challenge. As per the insider security model, the adversary $\mathcal{A}$ knows the private key x_A of the sender. He can decrypt challenge ciphertext as follows:

– $\mathcal{A}$ decrypts the share of B_i $(1 \leq i \leq t)$ as
  - $k_i = s_i + r_i.x_A$
  - $K_i = g^{k_i} \bmod p$
  - $T_i = H_1(y_i^{k_i}) \bmod p$
  - $m_i = D_{T_i}(c_i)$

– Integrates each receiver's share $f(x) = \prod_{j=1,j\neq i}^{t} m_i \frac{x-j}{i-j}\ mod\ q$

– Recovers the subliminal message $m_s = f(0)$

Hence he easily distinguishes between the messages m_0, m_1.

Remark : This attack is possible because the randomness used in the signature part can be easily extracted using sender's private key. The same randomness is used in key by which signcryption is done. Hence the ciphertext can be easily decrypted if the private key of sender is known. Also, the identity of the receiver is not bound with the key which makes it CCA insecure. The above scheme attack also holds in the scheme without threshold when there is a only a single receiver.

5 Threshold Shared Unsigncryption Scheme Preventing Malicious Receivers by Zhang et al. [15]

In this section we review signcryption scheme with threshold shared unsigncryption preventing malicious receivers proposed by Zhang et al. We also show that it is not CPA secure.

5.1 Review of the Scheme [15]

The threshold scheme in [15] involves the following roles : A trusted center CA who is responsible for generating parameters, the sender A and the receiver group $B = \{B_1, B_2, ..., B_n\}$ and associated identity i $(1 \leq i \leq n)$. It consist of following algorithms:

1. ***Initialization :*** CA selects two large prime numbers p and q that satisfy $q|(p-1)$, a generator g with order q in $\mathbb{Z}_p^*$. It also chooses a secure symmetric cipher (E, D), one way hash function H and a keyed hash function $KH.CA$ randomly selects an integer $x_A \in \mathbb{Z}_q^*$ to be the private key for A and computes the corresponding public key as $y_A = g^{x_A} mod\ p$. Similarly, for the receiver group B, it computes private key x_B and public key y_B. After that, CA randomly generates a $(t-1)$ degree polynomial.

$$f(x) = x_B + a_1 x + ... + a_{t-1} x^{t-1} \bmod q$$

over Z_q satisfying $a_0 = f(0) = x_B$. Then, CA computes $x_i = f(i)$ as private key of B_i $(1 \leq i \leq n)$, thus the corresponding public key is $y_i = x_i P$. Finally, CA delivers x_A, x_B, x_i to A, B, B_i $(1 \leq i \leq t)$ respectively through secure channel and publishes system's public parameters $\{p, q, g, H, KH, E, D, y_A, y_B, y_i\}$.

2. ***Signcryption*** **:** To send a message m to the receivers group, A randomly picks $x \in \mathbb{Z}_q^*$ and computes (c, z, s) as follows :
   - $k = H(y_B^x)\ mod\ p$, split it into k_1 and k_2, e.g. let $(k_1 \parallel k_2) = H(k)$.
   - $r = KH_{k_1}(m, bind - info)$, where *bind-info* is information to identify receiver's group, such as the group's public key.
   - $s = x/(r + x_A)\ mod\ q$.
   - $z = g^r\ mod\ p$.
   - $c = E_{k_2}(m)$.
3. ***Unsigncryption*** **:** Without loss of generality, let $B = \{B_1, B_2, ..., B_t\}$ be the t receivers of the group B that want to cooperatively unsigncrypt the message. Firstly each user B_i uses his own private key to compute

$$F_i = (y_A z)^{sJ_i}\ mod\ p$$

where

$$J_i = x_i \prod_{j=1, j \neq i}^{t} \frac{-j}{i-j}\ mod\ q.$$

and presents it to the other participants in B. With the knowledge of $F = \prod_{i=1}^{t} F_i$, the message can be unsigncrypted as follows :
   - $k = H(F)$ and split it into k_1 and k_2.
   - $m = D_{k_1}(c)$.
   - check if $z = g^{KH_{k_2}(m, bind-info)}\ mod\ p$.

5.2 Attack on the Scheme [15]

The scheme in [15] is insecure from the point of view of confidentiality. If any member B_l of the receiver group B is malicious, B_l can decrypt any ciphertext (c, z, s) from the sender C to group B by claiming that he is decrypting a ciphertext (c', z', s) from sender A as follows:

- B_l calculates z' as $z' = y_A{}^{-1}.z.y_C$
- Each B_i computes its share as,

$$F_i = (y_A z')^{sJ_i}\ mod\ p = (y_A . y_A{}^{-1}.z.y_C)^{\left(\frac{x}{r + x_C}.J_i\right)}$$
$$= (y_C . z)^{\left(\frac{x}{r + x_C}.J_i\right)} = g^{x.J_i}$$

The other members cannot retrieve the correct message due to the wrong signcryption (c') given by B_l. The other members of group B other than B_l will get some junk message on unsigncryption of c', z', s, whereas B_l computes the message sent by C without knowledge of the other group members. The unsigncryption key and message are retrieved by B_l by performing,

- $F = \prod_{i=1}^{t} F_i mod\, p$
- $k = H(F)$
- $m = D_k(c)$

Remark : This attack is possible because the verification is carried out after the message recovery when all the shares are already produced and the signcryption key does not have binding to the identity of the sender.

6 Conclusion

In this paper, we showed the attacks in three PKI based signcryption schemes in the threshold settings. We proposed an improvement to the scheme in [10] and formal proved of security of the system in the newly proposed security model.

References

1. Li Aihong Ping, J., Zheng, M.: A threshold subliminal channel for manet using publicly verifiable hybrid signcryption. In: ISW, pp. 218–232 (1994)
2. Baek, J., Steinfeld, R., Zheng, Y.: Formal proofs for the security of signcryption. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 80–98. Springer, Heidelberg (2002)
3. Baek, J., Steinfeld, R., Zheng, Y.: Formal proofs for the security of signcryption. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 80–98. Springer, Heidelberg (2002)
4. Bao, F., Deng, R.H.: A signcryption scheme with signature directly verifiable by public key. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, pp. 55–59. Springer, Heidelberg (1998)
5. Boyen, X.: Multipurpose identity-based signcryption (a swiss army knife for identity-based cryptography). In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 383–399. Springer, Heidelberg (2003)
6. Chow, S.S.M., Yiu, S.-M., Hui, L.C.K., Chow, K.P.: Efficient forward and provably secure id-based signcryption scheme with public verifiability and public ciphertext authenticity. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 352–369. Springer, Heidelberg (2004)
7. Desmedt, Y.: Society and group oriented cryptography: A new concept. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 120–127. Springer, Heidelberg (1988)
8. Desmedt, Y., Frankel, Y.: Shared generation of authenticators and signatures (extended abstract). In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 457–469. Springer, Heidelberg (1992)

9. Libert, B., Quisquater, J.-J.: Efficient signcryption with key privacy from gap diffie-hellman groups. In: Bao, F., Deng, R.H., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 187–200. Springer, Heidelberg (2004)
10. Li Changgen, X.P.: Threshold signcryption scheme based on elliptic curve cryptosystem and verifiable secret sharing. IEEE, Los Alamitos (2005)
11. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. ACM Commun. 21(2), 120–126 (1978)
12. Steinfeld, R., Zheng, Y.: A signcryption scheme based on integer factorization. In: Okamoto, E., Pieprzyk, J.P., Seberry, J. (eds.) ISW 2000. LNCS, vol. 1975, pp. 308–322. Springer, Heidelberg (2000)
13. Yang, G., Wong, D.S., Deng, X.: Analysis and improvement of a signcryption scheme with key privacy. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 218–232. Springer, Heidelberg (2005)
14. Zheng, Y.: Signcryption and its applications in efficient public key solutions. In: Okamoto, E., Davida, G.I., Mambo, M. (eds.) ISW 1997. LNCS, vol. 1396, pp. 291–312. Springer, Heidelberg (1998)
15. Zhang, Z., Mian, C., Jin, Q.: Signcryption scheme with threshold shared unsigncryption preventing malicious receivers. In: Proceedings of IEEE TENCON 2002 (2002)

Provably Secure Password-Authenticated Group Key Exchange with Different Passwords under Standard Assumption*

Fengjiao Wang[1,2] and Yuqing Zhang[1,2]

[1] National Computer Network Intrusion Protection Center, GUCAS, Beijing, 100049, P.R. China

[2] State Key Laboratory of Information Security, GUCAS, Beijing, 100049, P.R. China

{wangfj,zhangyq}@gucas.ac.cn

Abstract. Although many password-authenticated group key exchange protocols have been proposed in recent years, it remains a non-trivial task to establish strong provable security guarantees without making idealized assumptions. In this paper, blending the best of a variant EKE scheme and Burmester-Desmedt (BD) group key exchange protocol, we present a provable secure and efficient different password-authenticated group key exchange (DPWA) protocol of three-round in the multicast setting. Based on the previous works, we first give a strengthened security model for this case, and then provide a security proof of our protocol in this model under the standard assumption.

Keywords: group key establishment, password-based authentication, provable security, dictionary attacks.

1 Introduction

An authenticated group key exchange protocol allows a pool of parties to agree on a common session key over an open network in a secure and authenticated manner which may later be used to achieve cryptographic goals like multicast message confidentiality or multicast data integrity. With the rapid development of internet, there is a great need for secure group key exchange schemes in many applications, such as electronic conferences, military operations and so on. Therefore, over the years, a lot of attempts have been made to design secure and efficient authenticated group key exchange protocols.

Password-authenticated Key Exchange. In distributed systems, password-authenticated key exchange (PAKE) is a practical and attracting way to achieve authentication and key establishment in view of its convenience, mobility and less hardware requirements, since secure session keys are established only by pre-sharing a human memorable password between (or among) communication

* This work was supported by the National Natural Science Foundation of China(60773135,90718007,60970140); The National High Technology Research and Development Program of China(863 Program) (2007AA01Z427,2007AA01Z450).

parties. However, a password is usually drawn from a relatively small dictionary and therefore it has a low-entropy, which makes PAKE schemes vulnerable to dictionary attacks. As is known, there are three ways in making use of the password to establish group-oriented communications. The first is by sharing a single password among group members to achieve authentication and session key establishment (SPWA)[1]. The second is in a pair-wise way, in which each client shares a password with his left and right neighbors, authenticates each other and establishes a common session key using these pair-wise keys [2]. The third is the different password-authenticated setting (DPWA) [4] in which authentication and session key establishment are implemented with the help of a trusted server, with who each party shared a different password.

The single password type schemes are used for the settings when the size of the group is small. It is difficult for a large number of parties to keep sharing a single password synchronously and securely for the following reasons. Exit of group members or compromise of passwords leads to breakdown of the entire system and an update of the shared password is needed, which could be a very expensive process. As far as the pair-wise setting is concerned, it is also not preferable in some applications, because how to share a password between previously unknown parties in a trusted way is a practical problem. In contrast, it is easy to see that the different password setting avoids the above problems, reflects more accurately what is happening in the real world and is more suitable for the large scale network, especially for the dynamic groups.

Related works. For the past decades, the design of two-party PAKE protocol has been explored intensively. Most PAKE schemes in the literature have focused on the shared password-authentication (SPWA) model which provides password-authenticated key exchange using a shared password between a client and a server [6], [13] etc. Recently, based on the previous work on 2PAKE and group key exchange (GKE), a number of group password-authenticated key exchange (GPAKE) schemes in SPWA setting [1],[5],[8],[9] were proposed and most of them are treated with formal security proof. For another branch, several GPAKE schemes in pair-wise way were also given in recent years [2],[7], in which a protocol compiler transforming a 2PAKE to group was presented by Abdalla et al.[2] based on the work of Bohli et al.[7].

In the meantime, with the extensive applicability in large scale and dynamic group setting being realized, research on schemes of DPWA-type has been receiving increasing attention. Remarkably, Abdalla et al.[11] first gave a formal security model for three-party PAKE scheme in DPWA setting, which became the basis for many later DPWA-type works[12],[3],[14] with formal security treatments. Independently, Byun and Lee [4] firstly considered DPWA-type schemes in group scenario and proposed two GPAKE protocols based on Diffie-Hellman key exchange, the essence of which is a key distribution. However, the schemes EKE-U and EKE-M in [4] were shown to be insecure against off-line dictionary attacks and undetectable on-line password guessing attacks [15], respectively, though they were proved to be secure in the random oracle model. Furthermore, their strengthened version in [16] as well as the extended application of

EKE-M in MANET network [18] were also pointed out to have security flaws [16]. Later, an efficient DPWA-type GPAKE protocol, as a combination of the hierarchical key tree structure and the password-based Diffie-Hellman exchange, was proposed by Wan et al.[3] with formal security proof in random oracle model. While the defect in this scheme is: the curious server can compute the group session key due to its key tree structure. Almost simultaneously, Wang et al.[14] gave a generic construction of DPWA-type GPAKE protocol in the multicast setting. They also provided a formal security proof in the standard model assuming that the underlying 2-party PAKE scheme is provable secure under the standard assumption and their scheme is of at least 5-round.

Our contributions. Based on the previous works on the security model for PAKE schemes of DPWA-type and group key exchange, we first present an enhanced security model for GPAKE protocols in DPWA setting, which takes semantic security in the ROR sense, forward secrecy, mutual authentication and key privacy against passive server in consideration. Then, by blending the best of a pretty-simple EKE protocol in [19] and Burmester-Desmedt (BD) group key exchange protocol [10],we propose a three-round and provably secure DPWA-type GPAKE protocol, which allows a group of parties to agree on a common session key concurrently in a multicast network environment with respective distinct passwords by the help of a trusted server. Finally, we provide a security proof in the strengthened model under the standard assumption.

Outline of the paper. In section 2, the security primitives that will be used in our scheme are briefly introduced. In section 3, we present the enhanced security model for DPWA-type GPAKE protocols based on previous works. In section 4, we describe our DPWA-type GPAKE protocol in detail and prove its security in the enhanced security model. Finally, the paper is concluded in section 5.

2 Security Assumptions

In this section, we briefly review the assumptions [14],[20] acting as building blocks to guarantee the security of our scheme.

Decisional Diffie-Hellman Assumption (DDH): Let $G =< g >$ be any finite cyclic group of prime order q. Informally, the DDH assumption is: it is difficult to distinguish the following real Diffie-Hellman distribution Γ_{real} and random Diffie-Hellman distribution Γ_{random}:

$$\Gamma_{real}=\{g^x, g^y, g^{xy} | x, y \in_R \mathbb{Z}_q\},\ \Gamma_{random}=\{g^x, g^y, g^z | x, y, z \in_R \mathbb{Z}_q\}$$

More formally, if we define the advantage function as

$$Adv_G^{ddh}(\mathcal{A})=|Pr[\mathcal{A}(X) = 1|X \in \Gamma_{real}]\text{-}|Pr[\mathcal{A}(Y) = 1|Y \in \Gamma_{real}]|,$$

We say the DDH assumption holds in group G if $Adv_G^{ddh}(\mathcal{A})$ is negligible for any probabilistic polynomial time adversary $\mathcal{A}$. We denote $Adv_G^{ddh}(t)$ the maximum value of over all adversaries running in time at most t.

Parallel Decisional Diffie-Hellman assumption (PDDH): Let us consider an extension of the DDH assumption, where there are the two distributions as follows:

$$PDDH_n^{*}=\{g^{x_1}, g^{x_2}, ..., g^{x_n}, g^{x_1x_2}, g^{x_2x_3}, ..., g^{x_nx_1} | x_1, x_2, ...x_n \in_R \mathbb{Z}_q\},$$
$$PDDH_n^{\#}=\{g^{x_1}, g^{x_2}, ..., g^{x_n}, g^{y_1}, g^{y_2}, ..., g^{y_n} | x_1, x_2, ...x_n, y_1, y_2, ..., y_n \in_R \mathbb{Z}_q\}$$

$\triangle$ is assumed to be a probabilistic polynomial (t, ε)-distinguisher for these two cases with the advantage$Adv_G^{pddh_n}(\triangle)$, so that the advantage function$Adv_G^{ddh}(t)$ is defined as the maximum value over all$(\triangle)$ with at most time complexity t.

Lemma 1. The $PDDH_n^{*}$is equivalent to the DDH for any prime order group G, any integer n and any time complexity T,

$$Adv_G^{ddh}(T) \leq Adv_G^{pddh_n}(T) \leq nAdv_G^{ddh}(T)$$

For more details about the proof of this lemma, please refer to [19].

Message authentication codes (MAC). A message authentication code scheme can be written as MAC=(Tag,Ver), where Tag is a MAC generation algorithm, possibly probabilistic, which produces a tag μ with the input of a message m and a secret key sk, and Ver is a MAC verification one, which takes a tag μ, a message m, and a secret key sk as the input, and then outputs 1 if μ is a valid tag for m under sk or 0 otherwise. A MAC scheme is existential unforgeability under chosen-message attacks (euf-cma) if the adversaries can not create a new valid message-tag pair, even after obtaining many valid message-tag pairs. Formally, let us consider the experiment, in which let l be a security parameter and sk be a secret key selected uniformly at random from $\{0, 1\}^l$, and let $\mathcal{A}$ be the adversary attacking the security of MAC, who is allowed to ask a MAC generation oracle Tag(sk; ·)and a MAC verification oracle Ver(sk; ·) and outputs a message-tag pair (m, μ). Let $Succ$ denote the event in which $\mathcal{A}$ generates a legal message-tag pair Tag(sk; ·) that was not outputted by the oracle on input m. The advantage of $\mathcal{A}$ in violating euf-cma is defined as $Adv_{\mathcal{A}}^{euf-cma} = Pr[Succ]$. We define $Adv_{\mathcal{A}}^{euf-cma}(t, q_g, q_v)$ as the maximal value of $Adv_{\mathcal{A}}^{euf-cma}$ over all $\mathcal{A}$ running in time at most t and asking at most q_g and q_v queries to its MAC generation and verification oracles, respectively.

3 Security Model

We describe below our enhanced security model following the Real-or-Random (ROR) model of Abdalla et al.[11] when considering semantic security of session keys, instead of the Find-then-Guess (FTG) model commonly used as in[3],[14], for the ROR model seems more suitable for the password-based setting. In the meantime, we consider the forward secrecy, key privacy against passive server and mutual authentication between user and server, based on the previous works in DPWA setting.

Protocol Participants. In a DPWA-type GPAKE protocol, there are two types of participants: clients and a trusted server. The total set of clients will be denoted by $\mathbb{C}$ and is assumed to be of polynomial size. By $U = \{u_1, u_2, ..., u_n\} \in \mathbb{C}$ we denote the set of protocol participants. We denote by S the server that is supposed to be always online. Each player participates in some distinct and possibly concurrent executions of the protocol, and each instance of their participation is modeled as an oracle. The l-th instance of the server is modeled as S^l and the s-th instance of u_i is modeled as u_i^s , where $1 \leq i \leq n, l, s \in N$

Long-lived keys. Each client obtains its distinct password p_i from a dictionary $\mathcal{D}$, and shares it with the server. The dictionary $\mathcal{D} \in \{0,1\}^*$ is assumed to be of constant or polynomial size and publicly available $\mathcal{D}$. The passwords are randomly chosen from with a uniform distribution.

Communication model. The adversary $\mathcal{A}$ is a probabilistic polynomial time machine that controls all communications and $\mathcal{A}$ can make queries to any instance. The list of queries that can make is as follows:

Execute(S^l,$\{u^{s_1}, u^{s_2}, ..., u^{s_n}\}$)**:** This query models passive attacks, in which the adversary $\mathcal{A}$ gets access to honest executions among the client instances $\{u^{s_1}, u^{s_2}, ..., u^{s_n}\}$ and the trusted server instance S^l by eavesdropping. The output of this query consists of the resulting transcript that was exchanged during the honest execution of the protocol π .

SendClient(u_i^s,m)**:** This query models an active attack against a client, in which $\mathcal{A}$ sends a message$m$ to the s-th instance of a client u_i and gets the output of oracle u_i^s after it processes m according to the protocol run. This query can be utilized by $\mathcal{A}$ to perform various active attacks such as impersonation attacks and man-in-the-middle attacks through modifying and inserting the messages of the protocol. A query *Send* initializes a new instance of the protocol π, and thus the adversary receives the initial flows sent out by the instance.

SendServer(S^l, m)**:** This query models an active attack against the trusted server, where the adversary $\mathcal{A}$ sends a message m to the server instance S^l and gets the output of oracle S^lin processing m according to the protocol run.

Reveal(u_i^s): This query models the misuse of group session keys by clients. Only if the session key of the client instance u_i^s is defined, the query is available and returns to the adversary the session key.

Test(u_i^s)**:** This query is used to measure the semantic security of the group session key. If the session key is not defined, it returns $\perp$. Otherwise, provided that the session key is defined and instance u_i^s is fresh, $\mathcal{A}$ can execute this oracle query at any time when being activated. Then, the session key is returned if $b = 0$ and a random number of the same size is returned if $b = 1$, where b is a random bit previously selected.

Corrupt(u_i)**:** These query models compromise of the long-term passwords p_i . The adversary $\mathcal{A}$ gets p_i by asking such a query, but he does not get any internal data of the instance being queried. Furthermore, with respect to the adversary $\mathcal{A}$, we assume that the server S is honest but curious, that is, S always receives and sends messages as the normal protocol requires, while he wants to get the session keys agreed on by the users.

Notions and Security Goals:

Freshness. An oracle is said to be fresh if none of the following conditions holds: (1) For some $u_j \in pid_i^s$, a query $Corrupt(u_j)$ was executed before a query of the form $Send(u_k^{s_k}, M)$ has taken place, for some message(or set of identities)M and some $u_k \in pid_i^s$).
(2) The adversary earlier queried Reveal(u_j^t) with u_i^s and u_j^tbeing partnered.

Forward secrecy. Forward secrecy is achieved if a weak corruption(corrupting a principal means only retrieving his long term password) does not give the adversary any information about previously agreed session keys.

Semantic Security in the ROR model. The security notion is defined in the context of executing a GPAKE protocol π in the presence of an adversary $\mathcal{A}$. During executing the protocol, the adversary $\mathcal{A}$ is allowed to send multiple queries to the *Execute*,*SendClient*, *SendServer*, *Corrupt* and *Test* oracles and asks at most one *Test* query to each fresh instance of each honest client, while it is no longer allowed to ask *Reveal* queries. Finally, $\mathcal{A}$ outputs its guess b' for the bit b hidden in the *Test* oracle. An adversary is said to be successful if $b' = b$. We denote this event by *Succ* . Provided that the passwords are drawn uniformly from a dictionary $\mathcal{D}$. We define the advantage of $\mathcal{A}$ in violating the semantic security of the protocol π and the advantage function of the protocol π, respectively, as follows:

$$Adv_{\pi,\mathcal{D}}^{ror-ake}(\mathcal{A}) = 2 \cdot Pr[Succ] - 1, Adv_{\pi,\mathcal{D}}^{ror-ake}(t, R) = max Adv_{\pi,\mathcal{D}}^{ror-ake}(\mathcal{A})$$

where the maximum is taken over all with time complexity at most t and using resources at most R (such as the number of oracle queries).

We say a GPAKE protocol π is semantically secure if the advantage $Adv_{\pi,\mathcal{D}}^{ror-ake}$ (t, R) is only negligibly larger than $kn/|\mathcal{D}|$, where n is the number of active sessions and k is a constant.

Mutual authentication. An adversary $\mathcal{A}_m$ against the mutual authentication of a correct GPAKE protocol π is allowed to ask *Execute*, *Send*, *Reveal* and *Corrupt* queries. $\mathcal{A}_m$ violates the mutual authentication property of the GPAKE protocol if at some point during the protocol run, there exists a fresh instance u_i^s that has accepted with a key $sk_{u_i}^s$ and another party $u_j \in pid_i^s$ that is uncorrupted at the time $sk_{u_i}^s$ accepts such that
(1) there is no instance u_j^t with $(pid_{u_j}^t, sid_{u_j}^s) = (pid_{u_i}^s, sid_{u_i}^s)$ or
(2) there is an instance u_j^t with $(pid_{u_j}^t, sid_{u_j}^s) = (pid_{u_i}^s, sid_{u_i}^s)$ that has accepted with $sk_{u_j}^t = sk_{u_i}^s$. The probability of the adversary $\mathcal{A}_m$ successfully winning the mutual authentication game is denoted as $Succ_{\mathcal{A}_{ma}}$. The protocol GPAKE is said to provide mutual authentication if $Succ_{\mathcal{A}_{ma}}$ is negligible in the security parameter k for any polynomial time $\mathcal{A}_{ma}$.

Key privacy against passive server. We require that no information about the session key is revealed to the server. Note that the server knows all passwords of the group members in the DPWA setting, so a malicious server is always able to impersonate one of its members and establish a common session key with the other group users by active attack. As a result, it is inevitable to learn the session key by a malicious server.

The passive server S could query two oracles: *Execute* and *Test*. We say S succeeds if he correctly guesses the value of the random bit b used in the *Test* query. Let $Succ^{kp}$ denote the event that the passive server succeeds. Let $\mathcal{D}$ be user's password dictionary. For any passive server S, we define his advantage $Adv_{\mathcal{D}}^{kp}(S)$ as

$$Adv_{\mathcal{D}}^{kp}(S)=2 \cdot Pr[Succ^{kp}] - 1, Adv_{\mathcal{D}}^{kp}(t,R)=max\{Adv_{\mathcal{D}}^{kp}(S)\}$$

where the maximum is over all adversaries with time-complexity at most t and querying oracles at most R times. We say the GPAKE protocol is key private against passive server if the advantage $Adv_{\mathcal{D}}^{kp}(t,R)$ is negligible.

4 A New GPAKE Scheme

4.1 Description of Our Scheme

In this section, we present a group PAKE protocol in the DPWA-setting. Let$\{u_1, u_2, ..., u_n\}$ be the users, who share a distinct password with the trusted server S, respectively and wish to establish a session key under the help of S. Simultaneously, we assume that these participants mentioned above are arranged in a ring with respect to the lexicographic order of their identities so that $u_1 = u_{n+1}$. Furthermore, the public information needed by the participants is as follows: A finite cyclic group G of order q in $\mathbb{Z}_p^*$ .Two large primes p and q with $p = 2q+1$, where p is a safe prime such that the DDH problem is hard to solve in G. g and h are generators of G both having order q, where g and h must be generated so that their discrete logarithmic relation is unknown. H is a hash function from $\{0,1\}^*$ to $\mathbb{Z}_p^*$.

Round 1: For $1 \le i \le n$, each user u_i chooses $x_i, y_i \in_R \mathbb{Z}_q^*$, computes $X_i = g^{X_i} \cdot h^{H(p_i||u_i)}$,$Y_i = g^{y_i}$ and broadcasts the message $u_i||1||X_i||Y_i$.

Round 2: upon receiving the message from each $u_i(1 \le i \le n)$, S decrypts X_i using u_i's p_i , chooses $s_i \in_R \mathbb{Z}_q^*$, computes $K_i = (g^{x_i})^{s_i}$,$Z_i = g^{s_i} \cdot h^{H(p_i||u_i)}$ and $MAC(K_i, Z_i, Y_{i-1}, Y_{i+1})$. Then, S broadcasts the message:$\{Z_i||2||Y_{i-1}||Y_{i+1}|| MAC(K_i, Z_i, Y_{i-1}, Y_{i+1})\}_{i=1}^n$.

Round 3: On receiving the message from S, each user u_i computes $K_i' = Z_i/h^{H(p_i||u_i)}$(and then checks the correctness of $Z_i, Y_{i-1}, Y_{i+1}, MAC(K_i, Z_i, Y_{i-1}, Y_{i+1})\}_{i=1}^n$ .If it is valid,u_i computes $T_i = (Y_{i-1})^{y_i}$,$T_{i+1} = (Y_{i+1})^{y_i}$,$\xi_i = T_{i+1}/T_i$ and $MAC(K_i, \xi_i)$,and then broadcasts the message:$u_i||\xi_i||3||MAC(K_i, \xi_i)$. S checks the correctness of Z_i and each u_i checks that $\xi_1 \cdot \xi_2, ..., \xi_n = 1$. If at least one of these checks fails, terminate the protocol execution. Or else, each client u_i computes the group session key $Gsk = (T_i)^n {\xi_i}^{n-1} {\xi_{i+1}}^{n-2} \cdots \xi_{i+n-2}$. This key is equal to $g^{y_1 y_2 + y_2 y_3 +, ..., + y_n y_1}$, which is same for all $1 \le i \le n$.

4.2 Security Analysis of Our Scheme

Due to the limitation of length, we omit the proof of our scheme here, the concrete proof will be given in the full version of this paper.

Theorem 1. *Let G be a group in which the DDH assumption holds and MAC is an existential unforgeability secure scheme under chosen-message attacks (EUF-CMA). Let q_{exe} and q_{Test} represent the number of queries to Execute and Test oracles, and let $q_{send}^{u_i^s}$ represent the number of queries to the SendClient and SendServer oracles between the user instance u_i^s and the corresponding server instance s^l, and assume that there are n honest users who participate $q_{session}$ honest executions of the scheme. Then, the protocol we proposed in this paper is a correct group password-authenticated key exchange protocol and we have*

$$\begin{gathered} Adv_{\pi,\mathcal{D}}^{ror-ake}(t, q_{exe}, q_{Test}, q_{send}, q_{session}) \leq \\ 2 \cdot n \cdot q_{session} \cdot (q_{exe} + q_{send} + n + 1) \cdot Adv_G^{ddh}(T) \\ +2 \cdot n \cdot q_{session} \cdot Adv_{MAC}^{euf-cma}(t, 2, 0) \\ +2(q_{send} + 1)/N + 2 \cdot (q_{exe} + q_{send})^2/q \end{gathered}$$

$$Succ_{GPAKE}^{ma}(t, q_{exe}, q_{Test}, q_{send}, q_{session}) \leq n \cdot q_{session} \cdot (q_{exe} + qsend + 1) \cdot Adv_G^{ddh}(T)$$

$$\begin{gathered} +n \cdot q_{session} \cdot Adv_{MAC}^{euf-cma}(t, 2, 0) \\ +(q_{send} + 1)/N + (q_{exe} + qsend)^2/q \end{gathered}$$

where T is the maximum time-complexity for an adversary to solve the DDH problem in group G, N is the size of the dictionary $\mathcal{D}$.

Theorem 2. *In our GPAKE protocol, a passive server cannot learn the group session key among group users as long as the PDDHn assumption holds in the group G. Formally,*

$$Adv_{\mathcal{D}}^{kp}(t, q_{exe}) \leq 2 \cdot n \cdot q_{exe} \cdot Adv_G^{ddh}(t + 8 q_{exe} \tau_e)$$

where q_{exe} represents the number of queries to the oracle Execute, τ_e denotes the exponentiation computational time in G.

Theorem 3. *Let $\mathcal{A}$ be the adversary against forward secrecy of GPAKE protocol within a time bound t, with q_{send} queries to SendClient and SendServer oracles, q_{exe} Execute queries and assume that there are n honest users who participate $q_{session}$ honest executions of the scheme. Let $Adv_G^{ddh}(T)$ be the success probability against the DDH problem of an adversary in time T. Then, we have*

$$\begin{gathered} Adv_{GPAKE}^{fs}(\mathcal{A}) \leq 2 \cdot n \cdot q_{session} \cdot Adv_{MAC}^{euf-cma}(t, 2, 0) \\ +2(2n + q_{session} + q_{send}) \cdot Adv_G^{ddh}(t) \\ +(q_{exe} + q_{send})^2/q + 2 \cdot q_{send}/N \end{gathered}$$

where t is the maximum time for the adversary $\mathcal{A}$, N is the size of dictionary $\mathcal{D}$.

4.3 Comparison with Existing Schemes

As far as security is concerned, the two main advantages of our scheme are as follows:
(1) Provably secure under standard assumption. As far as we know, aside from Wang et al.'s scheme, whose security assumption depends on the underlying two-party PAKE, security of the other existing DPWA-type PAKE schemes is under random oracle or ideal cipher assumption, while ours is provably secure under standard assumption.
(2) Semantic security in RoR model. The security model we adopt consider semantic security in RoR sense, which is stronger then that in FTG model as proved in [11]. Semantic security of the other existing schemes is in the FTG model. Simultaneously, efficiency comparison with existing schemes is given by table 1.

Table 1. Comparison on efficiency with existing schemes

schemes	EKE-U[4]	EKE-M[4]	Wan's scheme[3]	Wang's scheme[14]	Our scheme
Round	3	3	3	min:5	3
Exp of client	(n+3)/2	2	5+$\lfloor logn \rfloor$	min:5	7
Exp of server	(n+1)(n+2)/2	2n	3n	min:2n	3n

5 Conclusion

So far, it is still a non-trivial task to design efficient group password-authenticated key exchange schemes under the standard assumptions, especially in the DPWA setting. In this paper, we define an enhanced security model for GPAKE protocols in DPWA setting, through considering semantic security in the RoR sense and incorporating the formal treatments for key privacy and forward secrecy. On this basis, we propose a three-round GPAKE protocol of DPWA-type without ideal assumptions, and prove its security in the enhanced security model. Hitherto, to the best of our knowledge, our protocol is the most efficient DPWA-type GPAKE scheme without ideal assumption.

References

1. Abdalla, M., Pointcheval, D.: A scalable password-based group key exchange protocol in the standard model. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 332–347. Springer, Heidelberg (2006)
2. Abdalla, M., Bohli, J., Vasco, M., Steinwandt, R. (Password) authenticated key establishment from 2-party to group. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 499–514. Springer, Heidelberg (2007)
3. Wan, Z., Bart, P.: N-PAKE+A Hierarchical Group Password Authenticated key exchange protocol Using different passwords. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 31–43. Springer, Heidelberg (2007)
4. Byun, J.W., Lee, D.H.: N-party encrypted diffie-hellman key exchange using different passwords. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 75–90. Springer, Heidelberg (2005)

5. Abdalla, M., Bresson, E., Chevassut, O., Pointcheval, D.: Password-based group key exchange in a constant number of rounds. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 427–442. Springer, Heidelberg (2006)
6. 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)
7. Bohli, J.-M., Vasco, M.I.G., Steinwandt, R.: Password-authenticated constant round group key establishment with a common reference string. In: Cryptology ePrint Archive, Report 2006/214 (2006)
8. Lee, S.-M., Hwang, J.Y., Lee, D.H.: Efficient password-based group key exchange. In: Katsikas, S.K., López, J., Pernul, G. (eds.) TrustBus 2004. LNCS, vol. 3184, pp. 191–199. Springer, Heidelberg (2004)
9. Dutta, R., Barua, R.: Password-based encrypted group key agreement. International Journal of Network Security 3(1), 30–41 (2006)
10. Burmester, M., Desmedt, Y.: A secure and efficient conference key distribution system (extended abstract). In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995)
11. Abdalla, M., Fouque, P.-A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (2005)
12. Wang, W., Hu, L.: Efficient and provably secure generic construction of three-party password-based authenticated key exchange protocols. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 118–132. Springer, Heidelberg (2006)
13. Bellovin, S.M., Merritt, M.: Encrypted Key Exchange: Password Based Protocols Secure against Dictionary Attacks. In: Proceedings 1992 IEEE Symposium on Research in Security and Privacy, pp. 72–84. IEEE Computer Society Press, Los Alamitos (1992)
14. Wang, W., Hu, L.: Provably Secure N-Party Authenticated Key Exchange in the Multicast DPWA Setting. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) Inscrypt 2007. LNCS, vol. 4990, pp. 93–107. Springer, Heidelberg (2008)
15. Tang, Q., Chen, L.: Weaknesses in two group diffie-hellman key exchange protocols. In: Cryptology ePrint Archive, Report 2005/197 (2005)
16. Byun, J.W., Lee, D.H., Lim, J.: Password-based group key exchange secure against insider guessing attacks. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS (LNAI), vol. 3802, pp. 143–148. Springer, Heidelberg (2005)
17. Phan, R.C.-W., Goi, B.-M.: Cryptanalysis of the n-party encrypted diffie-hellman key exchange using different passwords. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 226–238. Springer, Heidelberg (2006)
18. Byun, J.W., Lee, S.-M., Lee, D.H., Hong, D.: Constant-round password-based group key generation for multi-layer ad-hoc networks. In: Clark, J.A., Paige, R.F., Polack, F.A.C., Brooke, P.J. (eds.) SPC 2006. LNCS, vol. 3934, pp. 3–17. Springer, Heidelberg (2006)
19. Kobara, K., Imai, H.: Pretty-simple password-authenticated key-exchange under standard assumptions. IE-ICE Transactions E85-A(10), 2229–2237 (2002)
20. Katz, J., Yung, M.: Scalable protocols for authenticated group key exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003)

An Enhanced Password Authenticated Key Agreement Protocol for Wireless Mobile Network[*]

Zhigang Gao[1,2] and Dengguo Feng[1]

[1] State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[2] National Engineering Research Center of Information Security, Beijing 100190, China
{Zhigang2005,Feng}@is.iscas.ac.cn

Abstract. Password-based Authenticated Key Agreement (PAKA) protocols are widely used in wireless mobile networks, however many existing PAKA protocols have security flaws. In the 3GPP2 network, there are several PAKA protocols proposed to enhance the security of the Authentication Key distribution mechanism which is subjected to the Man-In-The-Middle attack. We point out the security flaws of such protocols in [10,6] and give two practical attacks on them. Moreover we propose an enhanced PAKA protocol which can resist both undetectable on-line and off-line password guessing attacks, and formally analyze its security in the random oracle model. In addition, we consider a special version of Diffie-Hellman problem called Degenerate Diffie-Hellman problem and propose two assumptions called Computational and Decisional Degenerate Diffie-Hellman assumption which are as difficult as CDH assumption and DDH assumption respectively.

1 Introduction

With the rapid development of wireless technology and applications, wireless communications become more and more popular in people's life. At the same time, security problems become important issues to be considered. Unlike wired networks which can resist part of attacks by physical access restrictions, everyone in the valid areas where are covered by radio access points can access the network resources if there is no available entity authentication mechanism. In the 3GPP2 network, the OTASP [11,7] (Over the Air Service Provisioning) is designed to enable and expedite the authentication and authorization procedures, by which potential wireless service subscribers can activate (i.e., become authorized for) new wireless services or current subscribers can request changes

[*] Supported by the National Natural Science Foundation of China under Grant No. 60803129; The National High-Tech Research and Development Plan of China under Grant Nos. 2007AA120404, 2007AA120405; The Next Generation Internet Business and Equipment Industrialization Program under Grant No. CNGI-09-03-03.

in their existing services, without the intervention of a third party or parties. One of the primary objectives of OTASP is to provide the Mobile Stations with a secure authentication key to facilitate authentication.

However, the OTASP is not completely secure since it is subject to a Man-In-The-Middle attack. Recently some protocols [14,10,6] are developed to enhance the security of the Authentication Key distribution mechanism of the OTASP. But these schemes do not achieve their design goals. We review these protocols, point out the security flaws of them and give efficient attacks on them next.

Firstly, Sui et al. [14] found out that Seo and Sweeney's simple authenticated key agreement protocol [12] suffers a reflection attack and loses its authentication capability. Then they developed an improved authenticated key agreement protocol with perfect forward secrecy using elliptic curve cryptography. They claim that their protocol eliminates the disadvantages of SAKA [12] and provides identity authentication, key validation, and perfect forward secrecy. They also show how their proposed protocol can be included in the 3GPP2 specifications for OTASP to improve the Authentication Key, which is the master key in IS-95 and CDMA2000 mobile networks, distribution.

Then, Lu et al. [10] point out the security flaw of Sui et al.'s method. Their protocol can not resist the off-line password guessing attack. Based on Sui et al.'s protocol, Lu et al. propose a new password-based authenticated key agreement protocol, which solves the problems of protocol in [14] and also could be used in 3GPP2 networks.

Furthermore, Chang et al. [6] argue that Lu et al.'s protocol [10] can not resist the parallel guessing attack. According to their paper, they launch this attack by guessing Q_{B2}^*, which is a point value transformed in the second step of the protocol in [10]. The point value is shown as (a, b), where $a, b \in [0, n-1]$ and n is a secure large prime. Therefore, they can guess all cases of (a, b) in $O(n^2)$ time, which is polynomial time. But, we should note that the variable n is a secure large prime, and generally the variable n should have 160 bits lengths at least. So Chang's attack is a power exponent time algorithm and can not solve by polynomial time adversaries in fact. Beside this, the modified version protocol proposed by Chang et al. involved the off-line password guessing attack, and we will show the practical attack below.

1.1 Attack on Lu et al.'s Protocol

Here, we give a practical undetectable on-line password guessing attack. As proposed in [10] and cited in [6], the two participants in Sui et al.'s protocol are called as Alice and Bob, and they share a low-entropy password S which is selected from a uniformly distributed dictionary D of size $|D|$. A point P with large prime order q is selected randomly from an elliptic curve $\mathbb{E}$. The value t is derived from the password S in a predetermined way. H is a secure one-way function. The set $(\mathbb{E}, P, n, D, H)$ is sent to Alice and Bob as public parameters.

To guess the password, the adversary Eve first guesses (maybe Eve selects it from a prepared directory) a password S', and derives t' from S' through pre-defined methods. Then Eve selects a random number $d_A \in Z_n$, computes

$Q_{A1} = (d_A + t')P, Q_{A2} = d_A^2 P$, and sends (Q_{A1}, Q_{A2}) to Bob. When Eve receives the message responded by Bob, she computes $X = d_A Q_{B1}$, and checks whether the equation $H_B = H(ID_A \parallel ID_B \parallel Q_{A1} \parallel Q_{B1} \parallel X)$ holds or not. If it holds, Eve guesses the password correctly, and then she responds as Alice following the protocol. If it is not true, Eve repeats the processes until she gets the right password, then she could respond the former unfinished session.

1.2 Attack on Chang et al.'s Protocol

The protocol developed by Chang et al. [6] is a slim modified version of Lu et al.'s [10]. In this section, we will demonstrate that the modification which is being carried out by Chang et al. makes their protocol be involved in a more serious situation: it can not resist the off-line password guessing attack. In addition, we note that it can not resist the undetectable on-line password guessing attack too. We do not give the detail of the undetectable on-line password guessing attack because it is similar as the attack in section 1.1. The public parameters of Chang et al.'s protocol are the same as Lu et al.'s protocol which is described in Section 1.1.

We suppose that Eve listened in the communication channels, and obtained all of the messages transformed between Alice and Bob. So, Eve have the messages $(Q_{A1}, Q_{A2}, Q_{B1}, H_A, H_B)$. Eve first guesses a password S' from D in accordance with the way she wants (may be ordered), and computes the corresponding t'. Then she computes $Y' = Q_{A1} - t'P$ and $H'_B = H(ID_A \parallel ID_B \parallel Q_{A1} \parallel Q_{B1} \parallel Y')$. Eve compares H'_B with H_B which was sent by Bob in the second step of Chang et al.'s protocol. If they are equal, Eve has got the right password; else she guesses another password and repeats the procedure until her guess is correct. The upper bound of the off-line password guessing attack is $O(|D|)$.

1.3 Our Contribution

In this paper, we focus on the PAKA protocols for wireless mobile networks without certificates and the Public Key Infrastructure because the Public Key Infrastructure is expensive to construct and maintain such as [16,15,13]. Firstly, we point out that the attack proposed by Chang et al. [6] on Lu et al.'s [10] scheme does not hold, and present an off-line password guessing attack on Chang et al.'s protocol which is based on the Lu et al.'s scheme. Moreover, we show that Lu et al.'s scheme is not secure enough too and give an undetectable on-line password guessing attack [8] on their scheme. Secondly, we propose an improved PAKA protocol, which is efficient and secure against the off-line and undetectable on-line password guessing attacks, based on Lu et al.'s [10] scheme. Our protocol can be applied to CDMA2000 networks to enhance the authentication key distribution procedure, and can be used in Wireless Local Area Network under the EAP [1] framework too. Thirdly, we consider a special version of Diffie-Hellman problem called Degenerate Diffie-Hellman (DDH) problem, and prove that the Computational DDH assumption is equivalent to the Computational DH assumption.

2 Preliminaries

Our protocol is based on the Elliptic Curve Cryptology (ECC), however, it can be used in other algebraic structure in which the Decisional Diffie-Hellman assumption holds. In this section we list the assumptions which will be used in our security proof with style of elliptic curve cryptology. Let $\mathbb{E}$ be an elliptic curve over a finite field F_m and let P is a point in $\mathbb{E}$ with a large prime order q.

Assumption 1 (Computational Diffie-Hellman (CDH) Assumption). *For $a, b \leftarrow_R Z_q$, given (aP, bP), computing abP is hard.*

Assumption 2 (Decisional Diffie-Hellman (DDH) Assumption). *For $a, b, z \leftarrow_R Z_q$, given (aP, bP, zP), deciding whether $abP = zP$ or not is hard.*

The assumptions mentioned above are based on the Elliptic Curve Discrete Logarithm Problem (ECDLP). The best known methods to solve ECDLP are Pollard approach and Pohlig-Hellman method. They are both fully exponential, while the best known methods to solve the Integer Factorization Problem (IFP) and the Discrete Logarithm Problem (DLP), on which most of the non-ECC cryptosystems rely, are sub-exponential.

Based on the ECDLP, we propose a modified version of Diffie-Hellman problem called Degenerate Diffie-Hellman problem. In the new problem, the two variable a and b, which are defined in Assumption 1, are equal, and we need to compute the element a^2P instead of abP or distinguish a^2P from a random element in $\mathbb{E}$. Like the situation of Diffie-Hellman problem, we propose two assumptions which both based on the Degenerate Diffie-Hellman problem.

Assumption 3 (Computational Degenerate Diffie-Hellman (CDDH) Assumption). *For $a \leftarrow_R Z_q$, given aP, computing a^2P is hard.*

Assumption 4 (Decisional Degenerate Diffie-Hellman (DDDH) Assumption). *For $a, z \leftarrow_R Z_q$, given (aP, zP), deciding whether $a^2P = zP$ or not is hard.*

We emphasize that the Degenerate Diffie-Hellman problem is also hard in other algebraic structures which the Discrete Logarithm problem still exists, although we have adopted the Elliptic Curve arithmetic to describe it. Next, we discuss the difficulty of the CDDH assumption and the DDDH assumption.

Theorem 1. *The CDDH assumption is equivalent to the CDH assumption.*

Proof. Firstly, we prove that if the CDH assumption holds then the CDDH assumption holds. To achieve this, we only need to prove if the CDDH problem is easy then the CDH assumption is no longer tenable. Assuming that we can solve the CDDH problem in polynomial time, given a CDH instance $(\mathbb{E}, P, q, aP, bP)$, we compute abP by the following step.

1. Computes $(a+b)P = aP + bP$;
2. Computes a^2P and b^2P using CDDH;
3. Computes $(a+b)^2P$ using CDDH;
4. Computes $2abP = (a+b)^2P - a^2P - b^2P$;
5. Computes $abP = \frac{1}{2}2abP$.

Secondly, we prove that if the CDDH assumption holds then the CDH assumption holds. To prove this, we only need to prove that if the CDH problem is easy then the CDDH assumption is no longer tenable. Assuming that we can solve the CDH problem, we show that we can solve the CDDH problem. Given a CDDH instance $(\mathbb{E}, P, xP)$, let the values of both variables aP and bP,which is described in assumption 1, be xP,then we can compute x^2P using CDH assumption. □

From the intuitive point of view, the DDDH assumption is as difficult as the DDH assumption. While, like the situation of DDH, the formal proof is complex and we do not discuss here.

3 Security Model for Our PAKA Protocol

To prove the security of the new protocol proposed in this paper, we extend the formal security model which is introduced in [2]. In this model, there are three classes of participants: clients, servers and the adversary. Each principal is either a client or a server. Let U_i denote the i-th user in the client set, while S_j denote the j-th server in the server set. In the special k-th execution of the protocol, we use $U_{i,j}^k$ to denote the client involved in this communication, while $S_{j,i}^k$ to denote the server. Let b be a bit chosen uniformly which is used in the *Test* query.

We use SID, a session ID, to indicate an execution of the protocol, where SID is the conjunction of all messages the participant sent and received in this interaction.

Definition 1 (Partner). *We say that a client $U_{i,j}^k$ and a server $S_{j,i}^k$ are partnered if the following conditions are met: (1) They are both accepted (if exist); (2) they share the same SID; (3) No oracle besides them accepts with the same SID.*

The interactions between an adversary Eve and the participants of the protocol occur via oracle queries, which model the adversary's capabilities in the real attack. All oracle queries in our model are listed in the following:

– $H_i(x)$: We give the adversary the ability to access the hash functions. In the random oracle model, hash functions are formalized as random oracles and maybe there are more than one hash functions in a protocol.
– $Execute(U_i, S_j)$: This oracle is used to simulate the eavesdropping attack. This call carries out an honest execution of the protocol between oracle $U_{i,j}^k$ and $S_{j,i}^k$, where the variable k is the sequence number of this protocol execution and it is maintained by the simulator.

- $Send(U_{i,j}^k, S_{j,i}^k, x)$ This oracle query enables the adversary to perform an active attack on a client or a server. According to the input message x, the oracles $U_{i,j}^k$ or $S_{j,i}^k$ execute the operations defined in PAKA protocols. Finally the query returns the messages produced according the protocol. Message x can be λ in the query which causes an oracle to be generated as an initiator, otherwise as a responder.
- $Reveal(C_{i,j}^k/S_{j,i}^k)$: The adversary uses this query to gain the session key hold by $C_{i,j}^k$ or $S_{j,i}^k$. If the oracle has been accepted, the session key is returned; else a symbol $\perp$ is returned.
- $CorruptClient(U_i, S_j)$: This query models exposure of a client U_i's password shared with the server S_j.
- $CorruptServer(S_j)$: This query models an exposure of a server S_j's secret key t.

If the adversary decides to end the first phase, he chooses a *fresh* oracle and issues a $Test$query. In [2], there are two notions of freshness: with and without forward secrecy (fs) . Here, we only consider the former for that we want to prove that our protocol providers perfect forward secrecy. The definitions of *freshness* and *test* query are described below:

Definition 2 (Freshness with forward secrecy). *We say that an oracle is fresh if the following conditions hold: (1) he has accepted; (2) No* Reveal *queries have been made to him or his partner; (3) If he has been made a* Corrupt-Client/Server *query then he must not been made a* Send *query and vice versa.*

- $Test(U_{i,j}^k/S_{j,i}^k)$: This query is used to measure the semantic security of the authenticate key agreement protocol. If the oracle is not accepted, it return $\perp$. Otherwise, it return either the session key held by the oracle if $b = 0$ or a random key with the same distribution as the real session key. This query can be launched only once.

Now, we can define the advantage of the adversary in attacking the PAKA protocol. We say that the adversary Eve wins the game define above, in semantic security scene, if she asks a single $Test$ on a fresh oracle, outputs a single bit b', and $b' = b$ (where b is the random bit selected during the $Test$ query). The PAKA advantage of the adversary is twice the probability that Eve wins, minus one.

Definition 3. *We say that the protocol is a secure authenticated key agreement protocol if the following conditions hold:*

1. *In the presence of a benign adversary, which faithfully conveys messages, both oracles always accept holding the same session key, and this key is distributed uniformly on* $\{0, 1\}^k$*;*
2. *For any polynomial time adversary, The PAKA advantage of the adversary is negligible.*

4 Our Proposed Protocol

In this section, we describe the improved password authentication key agreement protocol in detail and show its security analysis. We are illumined by the method in Zhang et al.'s article [17] originally, and then we apply a similar idea to our protocol to make it resist the Undetectable on-line password guessing attack. Our protocol also provides a key confirmation procedure which is necessary for analogous protocols.

Our protocol is based on Lu et al.'s protocol [10] and uses elliptic curve cryptology which could be computed efficient in both of mobile devices and consumer computer. Especially in China, the elliptic curve cryptology has been adopted in the standard for Wireless Local Area Network and would be used in encryption and access control. We call the participants in our protocol Alice and Bob. Alice plays the role of clients and Bob of servers. Bob has an extra secret information than Alice besides a password shared with Alice. This is different from Lu et al.'s protocol. Our protocol is more suitable for an asymmetric authentication scene, such as the mobile network access authentication, than symmetric environments.

4.1 Construction

In the initial stage, Bob (the server) selects an elliptic curve $\mathbb{E}$ over a finite field F_m with large group order, and randomly selects a point P with large prime order q from $\mathbb{E}$. He also selects three collision resistant hash [4] functions $H_1 : D \rightarrow Z_q$, $H_2 : \{0,1\}^* \rightarrow \{0,1\}^l$ and $H_3 : \{0,1\}^* \rightarrow \{0,1\}^k$, where D is the set containing all the possible passwords, l is a secure parameter selected by Bob and k is the desirable length of session keys. Finally, the sever selects a random number $t \in Z_q$, set $(\mathbb{E}, q, P, Q = tP, D, H_1, H_2, H_3)$ as public parameters and keeps t secretly.

We assume that the client obtains the public parameters and a password S shared with the server in an extra register stage which can be realized in different methods and is not described here. Detail of the protocol is shown in Figure 1 and depicted below:

1. Alice first selects a random number $x \in Z_q$, and computes $T_{A1} = (x + H_1(S))P$, $T_{A2} = x^2P$ and $T_{A3} = H_2(xQ)$. Then Alice sends $(A, T_{A1}, T_{A2}, T_{A3})$ to Bob, where A denotes the identity of Alice.
2. After receiving $(A, T_{A1}, T_{A2}, T_{A3})$, Bob checks whether the equation $H_2(t(T_{A1} - H_1(S)P)) = T_{A3}$ holds or not. If it does not hold, Bob terminates the session and output $\perp$. Otherwise, Bob selects two random numbers $y_1, y_2 \in Z_Q$, and computes $Y = T_{A1} - H_1(S)P$, $T_{B1} = y_1P + y_2Y$, $T_{B2} = H_2(tY \parallel S)$ and $H_B = H_2(A \parallel B \parallel T_{A1} \parallel T_{A2} \parallel T_{A3} \parallel T_{B1} \parallel T_{B2})$. Then Bob sends (B, T_{B1}, H_B) to Alice, where B denotes the identity of Bob.
3. When Alice receives the message, she first verifies whether the equation $H_B = H_2(A \parallel B \parallel T_{A1} \parallel T_{A2} \parallel T_{A3} \parallel T_{B1} \parallel H_2(xQ \parallel S))$ holds or not. If it holds, A send $H_A = H_2(B \parallel A \parallel T_{B1} \parallel H_2(xQ \parallel S) \parallel T_{A1} \parallel T_{A2} \parallel T_{A3})$ to Bob. Then Alice accepts and sets the session key as $K_A = H_2(A \parallel B \parallel T_{A1} \parallel T_{A2} \parallel T_{A3} \parallel T_{B1} \parallel xT_{B1})$.

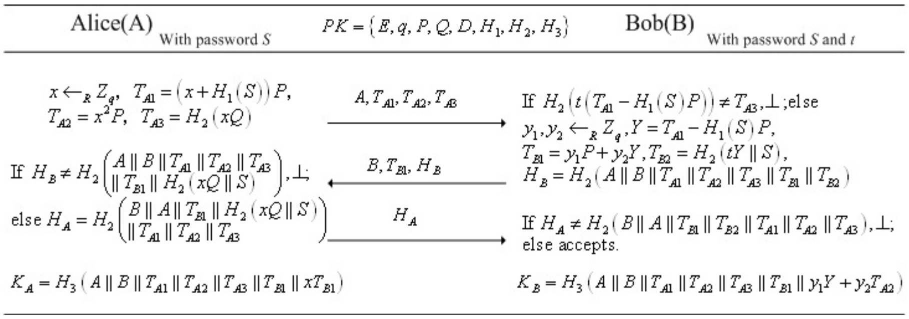

Alice(A) With password S
$PK = \{E, q, P, Q, D, H_1, H_2, H_3\}$
Bob(B) With password S and t
$x \leftarrow_R Z_q,\ T_{A1} = (x + H_1(S))P,$
$T_{A2} = x^2P,\ T_{A3} = H_2(xQ)$
$A, T_{A1}, T_{A2}, T_{A3}$
If $H_2(t(T_{A1} - H_1(S)P)) \neq T_{A3}, \perp$; else
$y_1, y_2 \leftarrow_R Z_q, Y = T_{A1} - H_1(S)P,$
$T_{B1} = y_1P + y_2Y, T_{B2} = H_2(tY \| S),$
$H_B = H_2(A \| B \| T_{A1} \| T_{A2} \| T_{A3} \| T_{B1} \| T_{B2})$
B, T_{B1}, H_B
If $H_B \neq H_2\begin{pmatrix}A \| B \| T_{A1} \| T_{A2} \| T_{A3} \\ \| T_{B1} \| H_2(xQ \| S)\end{pmatrix}, \perp;$
else $H_A = H_2\begin{pmatrix}B \| A \| T_{B1} \| H_2(xQ \| S) \\ \| T_{A1} \| T_{A2} \| T_{A3}\end{pmatrix}$
H_A
If $H_A \neq H_2(B \| A \| T_{B1} \| T_{B2} \| T_{A1} \| T_{A2} \| T_{A3}), \perp;$
else accepts.
$K_A = H_3(A \| B \| T_{A1} \| T_{A2} \| T_{A3} \| T_{B1} \| xT_{B1})$
$K_B = H_3(A \| B \| T_{A1} \| T_{A2} \| T_{A3} \| T_{B1} \| y_1Y + y_2T_{A2})$

Fig. 1. Our Proposed Protocol

4. When Bob receives the third message, he checks whether the equation $H_A = H_2(B \parallel A \parallel T_{B1} \parallel T_{B2} \parallel T_{A1} \parallel T_{A2} \parallel T_{A3})$ holds or not. If it holds, B sets the session key as $K_B = H_2(A \parallel B \parallel T_{A1} \parallel T_{A2} \parallel T_{A3} \parallel T_{B1} \parallel y_1Y + y_2T_{A2})$.

4.2 Security Analysis

In this section, we discuss the security of our protocol. First we prove that our protocol is a secure authenticated key agreement protocol with forward secrecy in the semantic security scene under the random oracle model [3], then we prove that our protocol can resist both undetectable on-line password guessing attack and the off-line password guessing attack.

Theorem 2. *The protocol is a secure AKA protocol with forward secrecy if the DDH assumption and DDDH assumption holds and the hash functions are modeled as random oracles.*

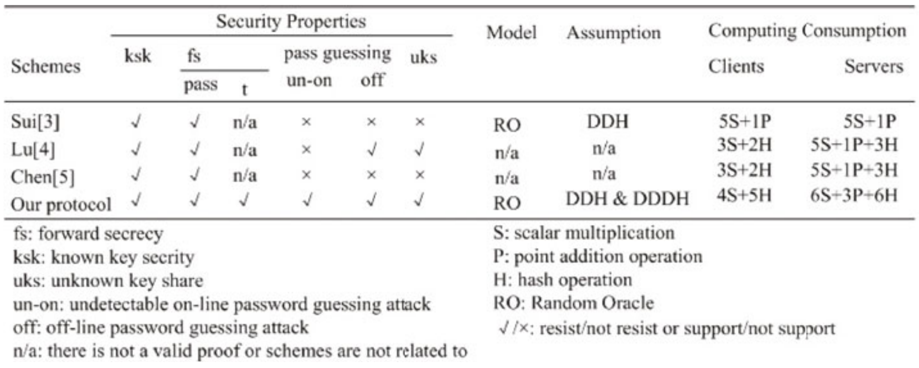
Moreover, our protocol is an password based authentication protocol combined with public key technologies. However, public key techniques are unavoidable for password protocols that resist off-line guessing attacks [9]. Our protocol not only resist the off-line and undetectable on-line guessing attack, but also keep the advantages such as convenience and efficiency of pure password protocols.

Theorem 3. *The protocol is secure against the off-line password guessing attack if the DDH assumption and DDDH assumption holds and the hash functions are modeled as random oracles.*

Theorem 4. *The protocol is secure against the undetectable on-line password guessing attack if the CDH assumption holds and the hash functions are modeled as random oracles.*

Due to the limitation of space, the proofs of Theorem 2, Theorem 3 and Theorem 4 are given in the full paper.

Table 1. Security Properties and Complexity Comparation



Schemes	Security Properties						Model	Assumption	Computing Consumption	
	ksk	fs		pass guessing		uks			Clients	Servers
		pass	t	un-on	off					
Sui[3]	√	√	n/a	×	×	×	RO	DDH	5S+1P	5S+1P
Lu[4]	√	√	n/a	×	√	√	n/a	n/a	3S+2H	5S+1P+3H
Chen[5]	√	√	n/a	×	×	×	n/a	n/a	3S+2H	5S+1P+3H
Our protocol	√	√	√	√	√	√	RO	DDH & DDDH	4S+5H	6S+3P+6H

fs: forward secrecy
ksk: known key secrity
uks: unknown key share
un-on: undetectable on-line password guessing attack
off: off-line password guessing attack
n/a: there is not a valid proof or schemes are not related to

S: scalar multiplication
P: point addition operation
H: hash operation
RO: Random Oracle
√/×: resist/not resist or support/not support

5 Efficiency and Application

As shown in Table 1, comparing to the scheme in [14] which five scalar multiplication and one point addition operations are needed both for Alice and Bob, our protocol is more suitable for mobile devices. Because of that, in our protocol, the client saves one scalar multiplication and one point addition operations. In contract with the scheme in [10] which needs five scalar multiplication, one point addition operations and three hash operations for the server and three scalar multiplication and two hash operations for the client, our protocol has to execute one scalar multiplication and two hash operations more both for the server and the client. But that is indispensable to resist the undetectable on-line password guessing attack. We list the detail information in the Table 1.

In [14], Sui et al. give a method to apply their protocol to improve the Authentication Key distribution in 3GPP2 networks. Our protocol can also be used in the 3GPP2 network with the same method. We note that our protocol also can be applied to the Wireless Local Area Network under the Extensible Authentication Protocol (EAP) framework [1]. The detail method to apply our protocol with EAP is similar as the method in [5], so we do not give a detail description here.

6 Conclusion

Our research focuses on the password-based authenticated key agreement protocol for wireless network. In this paper, we review the solutions which aim to improve the security of the Authentication Key distribution procedure in the OTASP, point out the security flaws of them [10,6] and give two practical attacks on them. By considering the security strength, computation efficiency and security properties, we proposed an enhanced PAKA protocol which can resist undetectable on-line and off-line password guessing attacks. Moreover, we prove the security of our protocol in the random oracle model.

References

1. Aboba, B., Blunk, L., Vollbrecht, J., Levkowetz, H., Carlson, J.: Extensible authentication protocol (eap). Technical report, The Internet Engineering Task Force. RFC 3748 (June 2004)
2. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
4. Bellare, M., Rogaway, P.: Collision-resistant hashing: Towards making uowhfs practical. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997)
5. Bersani, F., Tschofenig, H.: The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method. Technical report, The Internet Engineering Task Force. RFC 4764 (January 2007)
6. Chang, C.C., Chang, S.C.: An Improved Authentication Key Agreement Protocol Based on Elliptic Curve for Wireless Mobile Networks. In: IIHMSP 2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 1375–1378 (2008)
7. 3GPP2 C.S0016-B. Over-the-air service provisioning of mobile stations in spread spectrum standards. Technical report (October 2002), http://www.3gpp2.org
8. Ding, Y., Horster, P.: Undetectable on-line password guessing attacks. ACM SIGOPS Operating Systems Review 29(4), 77–86 (1995)
9. Halevi, S., Krawczyk, H.: Public-key cryptography and password protocols. ACM Trans. Inf. Syst. Secur. 2(3), 230–268 (1999)
10. Lu, R., Cao, Z., Zhu, H.: An enhanced authenticated key agreement protocol for wireless mobile communication. Computer Standards & Interfaces 29(6), 647–652 (2007)
11. 3GPP2 N.S0011-0. Otasp and otapa. Technical report, http://www.3gpp2.org (January 1999)
12. Seo, D.H., Sweeney, P.: Simple authenticated key agreement algorithm. Electronics Letters 35, 1073 (1999)
13. Singhal, A., Garg, V., Mathuria, A.: Analysis and Enhancement of Two Key Agreement Protocols for Wireless Networks. In: 2nd International Conference on Communication Systems Software and Middleware, COMSWARE 2007, pp. 1–7 (2007)
14. Sui, A., Hui, L.C.K., Yiu, S.M., Chow, K.P., Tsang, W.W., Chong, C.F., Pun, K.H., Chan, H.W.: An improved authenticated key agreement protocol with perfect forward secrecy for wireless mobile communication. In: 2005 IEEE Wireless Communications and Networking Conference, vol. 4 (2005)
15. Wan, Z., Zhu, B., Deng, R.H., Bao, F., Ananda, A.L.: Dos-resistant access control protocol with identity confidentiality for wireless networks. In: 2005 IEEE Wireless Communications and Networking Conference, vol. 3 (2005)
16. Wan, Z., Ananda, A.L., Deng, R.H., Bao, F.: Anonymous dos-resistant access control protocol using passwords for wireless networks. In: LCN, pp. 328–335. IEEE Computer Society, Los Alamitos (2005)
17. Zhang, Z., Feng, D.: On Password-based Key Exchange with Enhanced Security. In: The 4rd SKLOIS Workshop on Security Protocols, pp. 132–144 (2009)

Efficient Password-Based Authenticated Key Exchange Protocol in the UC Framework

Xuexian Hu and Wenfen Liu

Zhengzhou Information Science and Technology Institute,
Zhengzhou 450002, P. R. China
xuexian_hu@yahoo.com.cn

Abstract. In this paper, we propose a new password-based authenticated key exchange (PAKE) protocol and prove its security within the universal composability (UC) framework. The security proof of this protocol is based on standard number-theoretic assumptions, i.e., without random oracle or ideal cipher assumption. Comparisons show that, our protocol is more efficient than Canetti *et al.*'s protocol, which is the most efficient two party PAKE protocol proven secure in the UC framework and based on standard number-theoretic assumptions. More specifically, our protocol saves 1 round of communication and 5 modular exponentiations when the underlying cryptosystem is instantiated with Cramer-Shoup public key cryptosystem. Moreover, our protocol avoids the usage of the one-time signature, which saves the bandwidth for transmitting the message and saves the computation for signature and verification.

1 Introduction

Password-based authenticated key exchange (PAKE) protocols allow parties sharing only a low-entropy, human-memorizable password to securely establish a common session key over an insecure channel in authenticated manner. Since PAKE protocols do not require complex public-key infrastructure or trusted hardware of storing high entropy secrets, they have attracted many attentions since being introduced.

However, unlike a high entropy secret based protocol, PAKE protocols are susceptible to dictionary attacks, in which the adversary tries to conduct an attack by exhaustively trying all the values in the small set, called *dictionary*, for the correct password. Usually, dictionary attacks could be classified into two categories. In the *off-line dictionary attacks*, an adversary selects a password from a dictionary and verifies its guess in an off-line manner. In the *on-line dictionary attacks*, an adversary guessing a value for the target password must be present and interact with the system in order to verify whether its guess is correct. If the attack fails, the adversary can eliminate this value from the list of possible passwords. It is straightforward that one cannot actually prevent the adversary from the on-line dictionary attacks. However, we should expect that this attack is the only one that the adversary can mount, and invalidate the use of a password whenever a certain number of failed attempts occur. That is, the adversary's chance to defeat protocol goals should be restricted within a bound which depends on how many times it interacts with the system only, and it won't significantly depend on its off-line computing time.

Security models. The first formal models for analysis of PAKE protocols using password only were proposed in 2000 by Bellare, Pointcheval and Rogaway (BPR) [3] and by Boyko, MacKenzie and Patel (BMP) [4], independently. The security definition in the BPR model is indistinguishability-based, while the security definition in the BMP security model is a simulation-based one. Later, an improved security model was introduced by Abdalla *et al.* [2], which is proven to be strictly stronger than the BPR model. These security models provide quite reasonable security level and are used widely since being introduced.

Unfortunately, as pointed out by Canetti *et al.* [7], none of these models relates to the realistic setting where the password-based protocol is used as a component within a larger protocol. They fail to consider some scenarios such as protocol participants runs protocols with different but possibly related passwords. Rather, these security models assume that passwords being used are chosen from some pre-determined, known distribution and assume that passwords shared between different parties are independent. To overcome these deficiencies, Canetti *et al.* [7] proposed a new definition of security for PAKE within the UC framework, which captures these problems that were not adequately addressed by most prior notions and thus provides security guarantees under arbitrary composition with other protocols.

Related work. In 2001, a reasonable efficient and practical PAKE protocol without random oracle was proposed by Katz, Ostrovsky and Yung (KOY) [19] in the common reference string (CRS) model, in which all parties have access to a common string that is guaranteed to come from a pre-specified distribution. The security of KOY protocol relies only on standard number-theoretic building blocks, such as the Cramer-Shoup cryptosystem [10], one-time signature scheme [13] and the Decisional Diffie-Hellman (DDH) assumption.

Later, a framework for PAKE protocol, as an abstraction and generalization of the KOY protocol, was proposed by Gennaro and Lindell (GL) [16] in 2003, using generic building blocks instead of specific number-theoretic building blocks. More specifically, their construction uses non-malleable commitment [12], one-time signature scheme [13] and the smooth projective hash function family [11]. Very recently in 2008, Gennaro [15] slightly improved on the above framework by replacing one-time signature scheme with faster and shorter message authentication code (MAC). Nevertheless, all these protocols are 3-round and provide no explicit mutual authentication. Jiang and Gong [18] proposed yet another variant of the KOY protocol in 2004, which additionally supports mutual authentication while does not increase the round number.

In 2005, along with the security model, Canetti *et al.* also proposed a new protocol based on the KOY protocol [19] and on the GL framework, and proved its security in the UC framework against static corruption adversary. This two party protocol is the most efficient one which is proven UC secure in the standard model. Nevertheless, the resulting protocol is about 6 rounds and requires almost 30 modular exponentiations per party. Recently, Abadalla *et al.* [1] showed that the protocol by Bresson, Chevassut, and Pointcheval [5], which is more efficient than Canetti *et al.*'s protocol, is also secure in the model of Canetti *et al.* [7] even with the presence of adaptive corruption adversary. However, as an trade-off between the security and efficiency, the security proof of this protocol is based on the random oracle assumption and the ideal cipher assumption. Note that whether schemes proven secure in the random oracle model and the ideal cipher model can be instantiated securely in the real world is uncertain.

Contribution. In this paper, we propose a new PAKE protocol in the UC framework, which is based on standard number-theoretic assumption, i.e., without random oracle and ideal cipher assumption. We show that, by utilizing the projective hash value computed by the client as the random string used in computing a non-malleable ciphertext sent to the server, our protocol saves one round of communication and as much as 12 KBytes of bandwidth than Canetti *et al.*'s protocol [7]. Moreover, our protocol saves at least 5 modular exponentiations when instantiating the underlying public key encryption system with Cramer-Shoup public key cryptosystem [10].

We stress that our construction is inspired by the work of Jiang and Gong [18], in which a similar technique was used in constructing a PAKE protocol supporting mutual authentication. However, we also note that, Jiang *et al.* have not yet shown whether their protocol is secure or not without the last round, even in the BPR2000 model without explicit authentication, since that they use the CPA secure ElGamal encryption in computing the first message and their proof uses the last message for authentication in an essential way.

Organization. In Section 2 the UC framework and the PAKE functionality used for security definition are introduced. In Section 3 we recall the cryptographic tools that will be used in our protocol construction. The protocol is proposed in Section 4, in which comparisons with other protocols and intuitive security proof of our protocol are also presented. Finally, conclusions are given in Section 5.

2 Security Model

We assume the basic familiarity of the UC framework [6] and only briefly recall some related aspects here. Roughly speaking, the security of a given cryptographic task in the UC framework is captured via an `ideal functionality` $\mathcal{F}$, which is essentially a trust party that interacts with the parties in a secure way. More specifically, $\mathcal{F}$ receives inputs from the parties and sends back outputs in a private manner. Thus, in the ideal setting, security is inherently guaranteed. In order to test whether a candidate protocol π securely realizes an ideal functionality $\mathcal{F}$, an environment $\mathcal{Z}$ is involved, which provides inputs to and obtains outputs from all the participants and wants to distinguish the scenario where it is interacting with the real execution of the protocol π and a real adversary $\mathcal{A}$ from the scenario it is interacting with the ideal protocol (involving ideal functionality $\mathcal{F}$ and dummy parties) and an ideal adversary $\mathcal{S}$. We say that a protocol π securely realize the ideal functionality $\mathcal{F}$ if for every $\mathcal{A}$ there exists a $\mathcal{S}$, such that no environment $\mathcal{Z}$ can distinguish these two scenarios with non-negligible probability. The universal composition theorem then asserts that any larger protocol that uses multi-instances of $\mathcal{F}$ as its components behaves essentially the same when these instances are replaced with instances of protocol π.

Universal composition with joint state. For security analysis of our protocol, in which all executions use the same common reference string, we have to resort to the notion of universal composability with joint state (JUC) that was introduced by Canetti and Rabin [8], which provides a mean to deduce the security of the multi-instance case from the security of a single instance, even when these instances use some joint state, or joint subroutine. Specifically, this is done by defining a multi-session extension $\widehat{\mathcal{F}}$ of $\mathcal{F}$ as follows: $\widehat{\mathcal{F}}$ runs multiple independent copies of $\mathcal{F}$, where the copies are distinguished via sub-session identifiers (SSIDs). If $\widehat{\mathcal{F}}$ receives a message with SSID *ssid* it then hands the message to the copy of $\mathcal{F}$ having SSID ssid. If no such copy exists then a new one is invoked.

The PAKE functionality. We will use the definition of the PAKE functionality $\mathcal{F}_{pwKE}$ which is given by Canetti *et al.* [7] and incorporates the inherent “security defect” due to the use of low entropy passwords within standard key exchange. That is, the adversary is given the power to fully determine the resulting session key not only in case one of the participating parties is corrupted, but also in case that the adversary succeeds in guessing the parties’ shared password. An additonal important property of the definition of $\mathcal{F}_{pwKE}$ is that, password is chosen by the environment who then hands it to the parties as input. This formalization allows that the security of a protocol is preserved for all efficient password distribution, and even when the same passwords are used for other unintended purposed by the same environment.

3 Cryptographic Tools

In this section, we briefly recall the cryptographic tools needed in the construction of the protocol. Note that the encryption system and corresponding smooth projective hash function family could be instantiated with different assumptions, such as DDH assumption, Quadratic assumption and N-Residuosity assumption as specified in [16].

3.1 Labeled CCA2 Secure Public Key Encryption

A labeled CCA2 secure public key encryption system $(KeyGen, \mathcal{E}, \mathcal{D})$ is similar to the standard notion of CCA2 secure public key encryption system with the additional property that an arbitrary label can by bounded to the ciphertext in a non-malleable way. In more detail, for a pair of key (pk, sk) generated by the key generation algorithm $KeyGen$, the encryption algorithm $\mathcal{E}$ takes a public key pk, a plaintext m, a random string r, and additionally a optional label l, and returns a ciphertext $c = \mathcal{E}_{pk}(m; l; r)$. The decryption algorithm $\mathcal{D}$ takes a secret key sk, a ciphertext c and a label l, and returns $\mathcal{D}_{sk}(c; l)$, which is either the corresponding plaintext or a reject symbol $\perp$.

The standard CCA2 attack game is also modified in a similar way. The adversary $\mathcal{M}$ against the encryption system could query a pair of message m_0 and m_1 and a label l^* to an encryption oracle during the game, then a challenge ciphertext $c^* = \mathcal{E}_{pk}(m_b, l^*)$ with uniformly randomly chosen $b \in \{0, 1\}$ is returned to $\mathcal{M}$. During the game, the adversary is given access to the a decryption oracle which $\mathcal{M}$ could query on any ciphertext with any label except the challenge pair (c^*, l^*). The adversary's goal is to guess the hidden bit b. And we say that the encryption scheme is CCA2 secure if for any PPT adversary defined above, the probability that it rightly guess the bit is negligible.

3.2 Smooth Projective Hash Function Family

Informally, smooth projective hash function [11] is a family of hash functions which admit two keys. One key can be used to compute the hash values for all messages in the hash domain efficiently, the other key, called projection key, could be used to compute the hash values in some subset properly but given almost no information to the messages derived from the outside the specified subset, i.e., the value of the hash function on these messages is uniformly distributed.

As in [16], we present the notion of projective hash functions in the context of hard subset membership problems. Assume that X is set of superpolynomial size, L is a subset of X such that it is computationally hard to distinguish a random element in L from a random element in $X \backslash L$. Let G be finite, non-empty sets with prime order $q \approx 2^n$, where n is the security parameter. Let $\mathcal{H} = \{H_k\}_{k \in K}$ be a collection of hash functions from X to G, where K is called the key space of the hash family. Assume that $\alpha : K \times X \to S$ be a key projection function from $K \times X$ to the space of key projections S. Then the above system defines a projective hash function family, if for all $k \in K, x \in L$, it holds that the value $H_k(x)$ is uniquely determined by the key projection $s_x = \alpha(k, x)$ and a witness for x. The smooth property is defined as follows.

Definition 1 (Smooth projective hash function, [11]). *Let $(\mathcal{H}, K, X, L, G, S, \alpha)$ be a projective hash family defined as above. Denote by $V(x, \alpha(k, x), H_k(x))$ the random variable where x is some fixed value derived from the set $X \backslash L$, $k \in_R K$ is chosen uniformly at random. Similarly, define $V(x, \alpha(k, x), g)$ as the random variable where x is fixed and $k \in_R K, g \in_R G$ is chosen at random. This projective hash function family is said to be smooth if for every $x \in X \backslash L$,*

$$\{V(x, \alpha(k, x), H_k(x))\}_{n \in \mathbb{N}} \overset{s}{\equiv} \{V(x, \alpha(k, x), g)\}_{n \in \mathbb{N}} .$$

As noted by Gennaro *et al.* in [16], for $x \in_R D(L)$ of which corresponding witness is not known, the value $H_k(x)$ is computationally indistinguishable from random, even given the projection key s_x.

Lemma 1 (Pseudorandom property, [16]). *Assume that $\mathcal{E}$ is a CCA2 secure encryption algorithm, and pk is a randomly generated encryption key. Then for uniformly chosen hash key $k \in_R K$ and uniformly chosen coins r, it holds that for every message m, the ensemble $\{\mathcal{E}_{pk}(m;r), pk, m, \alpha(k,c), H_k(x)\}$ is computationally indistinguishable from the ensemble $\{\mathcal{E}_{pk}(m;r), pk, m, \alpha(k,c), g\}$, where g is uniformly derived from G.*

3.3 Simulation Sound Zero Knowledge Proof System

A zero knowledge proof system is said to be simulation sound if it has the property that an adversary cannot give a convincing proof for a false statement, even if it has oracle access to the zero knowledge simulator and can request simulated proofs for any statement. This notion was first formally defined by Garay *et al.* [14] in the context of interactive zero knowledge. We denote a proof protocol for the statement $x \in L$ in which the verifier V has input x and the prover P has input x, y by $\langle P(x,y), V(x) \rangle_{x \in L}$.

3.4 Universal Hash Function Family

A family of universal hash functions [9] is a family of hash functions with additional property that for a randomly selected function UH and any two distinct values $x \neq x'$ derived in the domain, the probability that the two hash values are equal, i.e., $\Pr[UH(x) = UH(x')]$, is less than some given bound ϵ, which usually is the value $1/2^{-n}$ when the range of the hash function is $\{0,1\}^n$. Note that as a consequence of the entropy smoothing theorem [17], if g is chosen uniformly at random in the domain, then the distribution of $UH(g)$ is statistically close to uniform random over its range.

4 The Protocol

In this section, we introduce our new PAKE protocol in the UC framework. Let n denote the security parameter. The common reference string of the protocol consists of a public key pk for the labeled CCA2 secure encryption scheme $\mathcal{E}$ and a reference string γ for the simulation sound zero knowledge proof system for the language $L_{pk} \overset{s}{=} \{(c_0, c_2) \,|\, \exists\, (pw, r_0, r_2) \text{ s.t. } c_0 = \mathcal{E}_{pk}(pw; r_0), c_2 = \mathcal{E}_{pk}(pw; r_2)\}$, where $\mathcal{E}_{pk}(m;r)$ denotes the encryption of the message m using random string r. Let C_{pk} denote the ciphertext set $\{\mathcal{E}_{pk}(m;r) | m \in M, r \in \{0,1\}^*\}$, where M is the message space. We also uses a family of smooth projective hash functions $\mathcal{H} = \{H_k\}$ such that for every k in key space K, H_k is a map from the set $C_{pk} \times M$ to a cycle group G with prime order $q \approx 2^n$. Let $\alpha : K \times (C_{pk}, M) \rightarrow S$ denote the key projection function. Denote by UH a hash function randomly selected from a family of universal hash function that takes elements from G and maps to the set $\{0,1\}^{2n}$. Denote by $UH_1(g)$ the first n bits of $UH(g)$ and by $UH_2(g)$ the remaining n bits of $UH(g)$.

Assume that P_i and P_j are two parties who want to establish a shared session key SK with each other. Then they involve in a protocol which proceeds as follows (see also in figure 1).

0. When P_i is activated with input ($\texttt{NewSession}, ssid, P_i, P_j, pw, role$), it does the following. If $role = server$, it simply waits for a `flow-zero` message as described below; If $role = client$, it chooses a random string r_0, computes the ciphertext $c_0 = \mathcal{E}_{pk}(pw; r_0)$, and sends the message ($\texttt{flow-zero}, c_0$) to party P_j. In the following, assume that P_i is a party activated as *client* and P_j is a party activated as *server*.
1. When the sever P_j receives a message ($\texttt{flow-zero}, c_0$), it first checks that the ciphertext is with the right form of a ciphertext, then chooses a random value r_1, computes $c_1 = \mathcal{E}_{pk}(pw; r_1)$, and sends ($\texttt{flow-one}, c_1$) to P_i.

2a. When P_i receives the message ($\texttt{flow-one}, c_1$), it first checks that the ciphertext is with the right form. Then it chooses a key k in the key space K of the smooth projective hash function family $\mathcal{H}$, computes the key projection $s = \alpha(k, c_1)$, then computes $\sigma = H_k(c_1, pw)$. He also computes $r_2 = UH_1(\sigma)$, sets $l_2 = i||j||ssid||c_0||c_1||s$, encrypts $c_2 = \mathcal{E}_{pk}(pw; l_2; r_2)$, and sends ($\texttt{flow-two}, s, c_2$) to party P_j.

2b. Party P_i also proves to P_j that c_0 in `flow-zero` and c_2 in `flow-two` are two ciphertexts of the same password, by engaging in a simulation sound zero knowledge proof system $\langle P(c_0, c_2, \gamma; pw, r_0, r_2), V(c_0, c_2, \gamma)\rangle_{(c_0,c_2)\in L_{pk}}$, which uses the witness pw, r_0, r_2 that it knows and also the reference string γ. P_i then computes the session key as $SK = UH_2(\sigma)$, outputs $(ssid, SK)$, and completes the session.

3. When P_j receives a message ($\texttt{flow-two}, s, c_2$), it checks the form of ciphertext. Then it involves in the simulation sound zero knowledge proof system as the verifier. If the proof is failed, it aborts and chooses a random session key. Otherwise, it then executes the following *ciphertext verifying procedure*: it computes $\sigma = H_k(c_1, pw)$ and $r_2 = UH_1(\sigma)$, then it sets $l_2 = i||j||ssid||c_0||c_1||s$ where c_1 is the ciphertext it computed and c_0, s are the messages it received. Then P_j re-ciphers the password with label l_2 and random string r_2 and verifies whether the equation $c_2{=}E_{pk}(pw; l_2; r_2)$ holds. If the verification is invalid then P_j also chooses a random value as the session key, else if it is valid it computes the session key as $SK = UH_2(\sigma)$. Either way, at last P_j outputs $(ssid, SK)$ and completes the session.

Protocol Comparisons. We compare our protocol with Canetti *et al.*'s protocol [7], which is the most efficient two party PAKE protocol constructed based on standard number-theoretic assumptions and proven secure in the UC framework, in terms of communication and computation efficiency. Firstly, Canetti *et al.*'s protocol [7] has 6 rounds when the zero knowledge proof system used is typically 3 rounds, while our protocol is only with 5 rounds. Further, in Canetti *et*

$$
\begin{array}{rcl}
 & \text{pub: } pk, \gamma & \\
P_i(pw) & & P_j(pw) \\
c_0 = \mathcal{E}_{pk}(pw; r_0) & \xrightarrow{(\texttt{flow-zero}, c_0)} & \\
 & \xleftarrow{(\texttt{flow-one}, c_1)} & c_1 = \mathcal{E}_{pk}(pw; r_1) \\
k \in_R K, s = \alpha(k, c_1) & & \\
\sigma = H_k(c_1, pw) & & \\
r_2 = UH_1(\sigma) & & \\
l_2 = i||j||ssid||c_0||c_1||s & & \\
c_2 = \mathcal{E}_{pk}(pw; l_2; r_2) & & \\
SK = UH_2(\sigma) & \xrightarrow{(\texttt{flow-two}, s, c_2)} & \\
 & \xrightarrow{\text{ZKP}\,\{(c_0, c_2) \in L_{pk}\}} & \\
 & & \text{If proof fails, } sk \in_R \{0,1\}^n \\
 & & \sigma = H_k(c_1, pw) \\
 & & r_2 = UH_1(\sigma) \\
 & & l_2 = i||j||ssid||c_0||c_1||s \\
 & & c_2 \overset{?}{=} \mathcal{E}_{pk}(pw; l_2; r_2) \\
 & & \text{If invalid, } SK \in_R \{0,1\}^n; \\
 & & \text{Otherwise, } SK = UH_2(\sigma)
\end{array}
$$

Fig. 1. The new universal composable PAKE protocol

al.'s protocol, assuming a security parameter of 128, we have that transmitting the key and the one-time signature requires about 12 KBytes [15]. However, our protocol, which avoids the usage of one-time signature and message authentication codes, will thus save as much as 12 KBytes of bandwidth. On the other hand, when instantiating the underlying public key encryption system with Cramer-Shoup public key cryptosystem [10] and instantiating smooth projective hash function family correspondingly, our protocol saves at least 5 modular exponentiations. Moreover, we stress that the remove of one-time signature provides improvement of computation efficiency too.

Security. The intuition behind the security of our protocol is somewhat similar to the security of Canetti *et al.*'s protocol. First note that if two completed sessions share the same password and accept each other as its intended partner, then they will generate the same session key, since they have agreed on the same c_1 and s. The pseudorandom property then guarantees that the adversary will be unable to distinguish this session key from random.

We then explain why a session which is affected by the adversary will generate a session key computationally close to uniform random. On one hand, if an adversary wants to impersonate the sever, then it has to send to the client the flow-one message. Due to non-malleability of the encryption scheme, the adversary could only send an invalid ciphertext or an old message derived from some past session. In the first case, the session key computed by the client session will be statistically close to uniform random due to the definition of smooth property. In the second case, the session key will be indistinguishable from random due to the pseudorandom property of smooth projective hash function. On the other hand, if the adversary wants to impersonate the client, it has to send the flow-two message to the server. Note that the ciphertext verifying procedure can only be successful when the ciphertext is an encryption of the right password held by the server, which is impossible due to the non-malleability of the underlying cryptosystem.

As a conclusion, we present the formal theorem for the security of the protocol in the following. The complete proof is omitted here due to the limitation of the paper length, which can be found in our full paper.

Theorem 1. *Assume that $\mathcal{E}$ is a labeled public key encryption scheme that is CCA2 secure, that $\mathcal{H}$ is a smooth projective hash function family as define above, that the proof system is simulation sound zero knowledge, and assume that UH is a hash function that randomly selected from a universal hash function family. Then, the protocol presented in Figure 1 securely realizes the multi-session extension $\widehat{\mathcal{F}}_{pwKE}$ of $\mathcal{F}_{pwKE}$ in the $\mathcal{F}_{CRS}$-hybrid model, in the presence of static corruption adversary.*

We stress that the protocol is proven secure in the static corruption model, where the adversary may corrupt some of the participants but only prior to the beginning of a protocol execution. Although this is a relatively weak assumption in the UC framework, Canetti *et al.* have proved that the weak corruption model of [3] is implied by this definition [7].

At last, we comment that our protocol in fact could be proven securely realizing the PAKE functionality with client authentication [1], since that the flow-two message sent to the server session includes the password shared between the two parties and is verified by the server session. In this case, we only need to modify the server session in such a way that it explicitly aborts and outputs the state information when the proof or the ciphertext verifying procedure is failed.

5 Conclusions

We proposed a new PAKE protocol and proved that it securely realize the PAKE functionality in the UC framework. Our protocol is based on standard number-theoretic assumptions and can be instantiated with different assumptions, such as DDH assumption, Quadratic assumption and N-Residuosity assumption as specified in [16]. By utilizing the projective hash value computed by the client as the random string used in computing a non-malleable ciphertext sent to the server, our protocol saves 1 round of communication and 5 modular exponentiations than Canetti *et al.*'s protocol. Moreover, our protocol omits the usage of the one-time signature, which saves one signature/verification computation and saves as much as 12 KBytes bandwidth when typical security parameter is used.

Acknowledgments . This work is supported by National Natural Science Foundation of China (No. 60873261) and National High Technology Research and Development Program of China (No. 2007AA01Z2a1, 2009AA01Z417).

References

1. Abdalla, M., Catalano, D., Chevalier, C., Pointcheval, D.: Efficient two-party password-based key exchange protocols in the UC framework. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 335–351. Springer, Heidelberg (2008)
2. Abadalla, M., Fouque, P., Pointcheval, D.: Password-based authenticatied key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (2005)
3. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attack. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
4. Boyko, V., MacKenzie, P., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
5. Bresson, E., Chevassut, O., Pointcheval, D.: Security proofs for an efficient password-based key exchange. In: Proc. of the 10th ACM Conference on Computer and Communications Security, pp. 241–250 (2003)
6. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: Proc. of 42nd IEEE Symposium on Foundations of Computer Science, pp. 136–145 (2001)
7. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
8. Canetti, R., Rabin, T.: Universal composition with joint state. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 265–281. Springer, Heidelberg (2003)
9. Carter, J.L., Wegman, M.N.: Universal classes of hash functions. Journal of Computer and System Sciences (18), 143–154 (1979)
10. Cramer, R., Shoup, V.: A practical public-key cryptosystem secure against adaptive chosen ciphertexts attacks. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
11. 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)
12. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. SIAM Journal of Computing 30(2), 391–437 (1999)
13. Even, S., Goldreich, O., Micali, S.: On-line/off-line digital signature. Journal of Gryptography (9), 35–67 (1996)
14. Garay, J., MacKenzie, P., Yang, K.: Strengthening zero-knowledge protocols using signatures. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 177–194. Springer, Heidelberg (2003)
15. Gennaro, R.: Faster and shorter password-authenticated key exchange. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 589–606. Springer, Heidelberg (2008)
16. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (2003)
17. Hastad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing 28(4), 1364–1396 (1999)
18. Jiang, S., Gong, G.: Password based key exchange with mutual authentication. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 267–279. Springer, Heidelberg (2004)
19. Katz, J., Ostrovsky, R., Yung, M.: Efficient password based authenticated key exchange using human memorable passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)

Efficient Generalized Selective Private Function Evaluation with Applications in Biometric Authentication

Helger Lipmaa[1,2] and Bingsheng Zhang[1,3]

[1] Cybernetica AS, Estonia
[2] Tallinn University, Estonia
[3] University of Tartu, Estonia

Abstract. In a selective private function evaluation (SPFE) protocol, the client privately computes some predefined function on his own input and on m out of server's n database elements. We propose two new generalized SPFE protocols that are based on the new cryptocomputing protocol by Ishai and Paskin and an efficient CPIR. The first protocol works only for constant values of m, but has 2 messages, and is most efficient when $m = 1$. The second SPFE protocol works for any m, has 4 messages, and is efficient for a large class of functionalities. We then propose an efficient protocol for private similarity test, where one can compute how similar client's input is to a specific element in server's database, without revealing any information to the server. The latter protocol has applications in biometric authentication.

Keywords: BDD, biometric authentication, CPIR, cryptocomputing, selective private function evaluation.

1 Introduction

In a selective private function evaluation (SPFE) protocol [CIK+01], the client privately computes some predefined function on m out of server's n database elements. It is required that the client will obtain only the function value, and the server will obtain no new information. More precisely, for client's indexes $x = (x_1, \dots, x_m)$, and server's database $f = (f_0, \dots, f_{n-1})$ and function g (either public or a private input of the server), the client obtains $g(f_{x_1}, \dots, f_{x_m})$. While SPFE has many different potential applications in privacy-preserving data mining, very few efficient solutions are known. In particular, all single-server SPFE protocols from [CIK+01] have communication complexity that is equal to at least the size of the Boolean circuit that is needed to implement g. In practice, g is often sufficiently complex, so that its circuit size is at least linear. (Here and in what follows, we measure linearity and polynomiality in n.) In such cases, the single-server SPFE protocols of [CIK+01] do not have sublinear communication complexity.

We first augment the definition of (n, m)-SPFE by letting the client to have another private input y, so that he will retrieve the value $g(f_{x_1}, \dots, f_{x_m}, y)$. This will enable us to use SPFE in more applications, including the private similarity test studied later in this paper. We call the result a generalized SPFE protocol. We then show how to implement SPFE for a large class of functionalities g in polynomial-time and sublinear (often low-degree polylogarithmic) communication. This will hold even if g has no sublinear-size circuits but has a binary decision diagram (BDD) with sublinear *length*. More precisely, we propose two different generalized SPFE protocols that are based on the recent PrivateBDD cryptocomputing protocol by Ishai and Paskin [IP07] and on an efficient $(n, 1)$-CPIR protocol. When using Lipmaa's $(2, 1)$-CPIR protocol from [Lip05], the communication of the PrivateBDD protocol is proportional to $\lambda \cdot (\lambda' + \mathsf{len}(g))$, where λ is the bitlength of client's private input, λ' is the length of client's private output and $\mathsf{len}(g)$ is an *upperbound* to the length of BDD that computes g. The online computation of PrivateBDD is $\mathsf{size}(g)$ public-key operations, where $\mathsf{size}(g)$ is the size of the BDD that corresponds to Bob's fixed input g.

The first protocol only works in the case of a constant m, and is especially efficient when $m = 1$. In this protocol, the server computes—by using the PrivateBDD protocol—a database of the answers $g(f_{i_1}, \ldots, f_{i_m}, y)$ for all values $(i_1, \ldots, i_m)$, and then the client and the server run an efficient two-message $(\binom{n}{m}, 1)$-CPIR protocol on this database so that the client retrieves $g(f_{x_1}, \ldots, f_{x_m}, y)$. This protocol requires us to compute g on n different inputs. However, it is still efficient in many cases where the evaluation of the BDD corresponding to g is efficient for known values of f_{i_j}.

The second protocol works for any value of m. It consists of an input selection protocol, after which the client obtains values $f_{x_1} \oplus r_1, \ldots, f_{x_m} \oplus r_m$ for random strings r_j that are chosen by the server. That is, the client and the server secret-share the values f_{x_j}. It is then followed by a PrivateBDD protocol, where client's inputs are $f_{x_j} \oplus r_j$ and y, server's inputs are g and r_j, and client's output is $g(f_{x_1}, \ldots, f_{x_m}, y)$. Due to the homomorphic properties of the underlying $(2, 1)$-CPIR protocol by Lipmaa [Lip05], it is straightforward to implement PrivateBDD on secret-shared inputs f_{x_j} based on a PrivateBDD protocol that uses non-shared input. The resulting generalized SPFE protocol has 4 messages, and requires to compute $(n, 1)$-CPIR m times, and then to execute PrivateBDD for g. This protocol has sublinear communication whenever g has a BDD with *length* that is sublinear in n.

We show how to apply the first generalized SPFE protocol in biometric authentication by presenting a new *private similarity test*. Consider the setting where the client (a fingerprint terminal) obtains a fingerprint y of some person, together with her claim that she is the xth employee (say, Alice). Then the client contacts the server who has a database of fingerprint templates of all employees. At the end of the protocol, the client gets to know whether fingerprint y is sufficiently close to the xth fingerprint template (in the sense of the Euclidean norm ℓ_2) in the template database to warrant access, without getting to know anything else. On the other hand, the server does not get any new information, including the fingerprint y, the value of x, or even whether the test succeeded. Moreover, the fingerprint templates at the server can be either unencrypted, or encrypted with the secret key that is known to the client.

Our private similarity test protocols works as follows. Assume that two fingerprints, represented as Boolean vectors of dimension $\lambda = |f_j| = |y|$ "match" if at least t of their coordinates match, that is, if $||f_j - y||_2^2 \leq \lambda - t$. By using the BDD for threshold function proposed in [ST97] and the CPIR of [GR05], we can constuct a private similarity test program with communication $\Theta(\lambda^2 \log^2 \lambda / \log\log\lambda \log\log\log\lambda + \log n)\kappa$ and server's online computation that is dominated by $\Theta(n \cdot \lambda \log^3 \lambda / \log\log\lambda \log\log\log\lambda)$

public-key operations. We are not aware of any previously proposed (provably secure) nontrivial private similarity tests.

2 Preliminaries

Client's input is $x \in \{0,1\}^\lambda$, server's input is a function $f : \{0,1\}^\lambda \to \{0,1\}^{\lambda'}$. κ is the security parameter. We denote $f(x)$ by f_x, that is, we think of f as of the characteristic function of the vector $f = (f_0, \dots, f_{2^\lambda - 1})$. n denotes the server's database size, and m is the number of queries to the database. All logarithms have base 2.

Public-Key Cryptosystems. Let $\mathsf{P} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a length-flexible additively-homomorphic public-key cryptosystem [DJ01], where Gen is a randomized key generation algorithm, Enc is a randomized encryption algorithm and Dec is a decryption algorithm. Here, both Enc and Dec receive an additional length parameter λ, so that $\mathsf{Enc}_{\mathsf{pk}}(\lambda, \cdot)$ encrypts plaintexts from some set $\{0,1\}^{\le\lambda}$. In the case of the DJ01 cryptosystem from [DJ01], for every integer $\lambda > 0$, $\mathsf{Enc}_{\mathsf{pk}}(\lambda, \cdot)$ is a valid plaintext of $\mathsf{Enc}_{\mathsf{pk}}(\lceil\lambda/\kappa\rceil \cdot \kappa + \kappa, \cdot)$, and therefore one can multiple-encrypt messages as say in $C \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\lambda + 2\kappa, \mathsf{Enc}_{\mathsf{pk}}(\lambda + \kappa, \mathsf{Enc}_{\mathsf{pk}}(\lambda, M)))$, and then recover M by multiple-decrypting, $M \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\lambda + 2\kappa, \mathsf{Dec}_{\mathsf{sk}}(\lambda + \kappa, \mathsf{Dec}_{\mathsf{sk}}(\lambda, C)))$. In practice, $2^\lambda < N$ where N is the public key of the DJ01 cryptosystem. Additionally, in any length-flexible additively-homomorphic cryptosystem, $\mathsf{Enc}_{\mathsf{pk}}(\lambda, M_1) \cdot \mathsf{Enc}_{\mathsf{pk}}(\lambda, M_2) = \mathsf{Enc}_{\mathsf{pk}}(\lambda, M_1 + M_2)$, where the addition is modulo the public key N. We will explicitly need the existence of a compression function C that, given pk, λ' and λ for $\lambda' \ge \lambda$, and $\mathsf{Enc}_{\mathsf{pk}}(\lambda', M)$ for $M \in \{0,1\}^\lambda$, returns $\mathsf{Enc}_{\mathsf{pk}}(\lambda, M) \in \{0,1\}^{\lceil\lambda/\kappa\rceil \cdot \kappa + \kappa}$.

In the LFCPA (*length-flexible chosen-plaintext attack*) game, the challenger first generates a random $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa)$, and sends pk to the attacker. Attacker chooses a polynomial number of message pairs (M_{j0}, M_{j1}) (such that $|M_{j0}| = |M_{j1}|$) and length parameters λ_j, and sends them to the challenger. Challenger picks a random bit b, and sends all ciphertexts $\mathsf{Enc}_{\mathsf{pk}}(\lambda_j, M_{jb})$ to attacker. Attacker outputs a bit b', and wins if $b = b'$. Because of the existence of the compress function, LFCPA security follows from the CPA security [Lip09]. Thus, the DJ01 cryptosystem [DJ01] is LFCPA-secure under the Decisional Composite Residuosity Assumption.

Cryptocomputing. Let λ and λ' be public parameters, and let $\mathcal{F}$ a class of functions $\{0,1\}^\lambda \to \{0,1\}^{\lambda'}$. In a cryptocomputing protocol for $\mathcal{F}$ between a client and a server, the client has an input $x \in \{0,1\}^\lambda$ and the server has an input $f \in \mathcal{F}$. The client obtains $f(x)$. Every cryptocomputing protocol $\Gamma = (\mathsf{Q}, \mathsf{R}, \mathsf{A})$ has two messages, where the client sends $\mathsf{Q}(\lambda', x)$ to the server, the server replies with $\mathsf{R} \leftarrow \mathsf{R}(\lambda', f, \mathsf{Q})$, and then finally the stateful client recovers f_x by computing $\mathsf{A}(\lambda', x, \mathsf{R})$. Here, Q, R and A are (probabilistic) polynomial-time algorithms.

Semisimulatable Privacy. Let $\Gamma = (\mathsf{Q}, \mathsf{R}, \mathsf{A})$ be a 2-message cryptocomputing protocol. Within this work we use the convention of many previous papers to only require (semisimulatable) privacy in the malicious model. More precisely, client's privacy is guaranteed in the sense of indistinguishability (CPA-security), while server's privacy is guaranteed (if at all) in the sense of simulatability. We now give an informal definition of privacy, see [IP07] for more. For the the privacy *of the client*, no malicious nonuniform probabilistic polynomial-time server should be able to distinguish, with non-negligible probability, between the distributions $\mathsf{Q}(\lambda', x_0)$ and $\mathsf{Q}(\lambda', x_1)$ that correspond to any two of client's inputs x_0 and x_1 that are chosen by herself. For *server-privacy*, we require the existence of an unbounded simulator that, given client's message Q^* and client's legitimate output corresponding to this message, generates server's message that is statistically indistinguishable from server's message R in the real protocol; here Q^* does not have to be correctly computed. A protocol is *private* if it is both client-private and server-private.

CPIR. A 2-message 1-out-of-n computationally-private information retrieval protocol, $(n, 1)$-CPIR is a special type of cryptocomputing protocol. In a $(n, 1)$-CPIR protocol for λ'-bit strings, the client has an index $x \in \{0, \dots, n-1\}$ and the server has a database $f = (f_0, \dots, f_{n-1})$ with $f_i \in \{0, 1\}^{\lambda'}$. The client obtains f_x. A $(n, 1)$-CPIR protocol $\Gamma = (\mathsf{Q}, \mathsf{R}, \mathsf{A}, \mathsf{C})$ is BDD-friendly if it satisfies the next four assumptions: (1) Γ has two messages, as any cryptocomputing protocol. (2) Γ is uniform in λ', that is, it can be easily modified to work on other values of λ'. (3) $|\mathsf{Q}(\lambda', \cdot)|, |\mathsf{R}(\lambda', \cdot, \cdot)| \leq \lambda' + \Theta(\kappa)$ (with possibly $\mathsf{Q}(\lambda', \cdot)$ being shorter). (4) The compress function C maps $\mathsf{Q}(\lambda'', x)$ to $\mathsf{Q}(\lambda', x)$ for any $\lambda'' \geq \lambda'$ and x. Here all 4 algorithms are (probabilistic) polynomial-time. The only known BDD-friendly $(2, 1)$-CPIR was proposed by Lipmaa in [Lip05], see [Lip09] for a compact description. Importantly for us, in this $(2, 1)$-CPIR protocol, $\mathsf{Q}(\lambda', x)$ consists of a public key and an additively homomorphic encryption of x under this key. Any $(n, 1)$-CPIR protocol Γ must be client-private, that is, CPA-secure. Lipmaa's $(2, 1)$-CPIR protocol [Lip05], when based on the DJ01 cryptosystem [DJ01], is CPA-secure and thus LFCPA-secure under the Decisional Composite Residuosity Assumption. A (semisimulatably) private $(n, 1)$-CPIR protocol is also known as an $(n, 1)$-*oblivious transfer protocol*.

Binary Decision Diagrams. A (multi-terminal) *binary decision diagram* (BDD, also known as branching program) is a fanout-2 directed acyclic graph (V, E), where the non-terminal (that is, non-sink) nodes are labeled by variables from some variable set $\{x_0, \dots, x_{\lambda-1}\}$, the sinks are labeled by λ'-bit strings and the two outgoing edges of every internal node are respectively labeled by 0 and 1. A BDD computes some function $f : \{0, 1\}^{\lambda} \to \{0, 1\}^{\lambda'}$. Every assignment of the variables selects one path from the source to some sink as follows. The path starts from the source. If the current version of path does not end at a sink, test the variable at the endpoint of the path. Select one of the outgoing edges depending on the value of this variable, and append this edge and its endpoint to the path. If the path ends at a sink, return the label of this sink as the value of the corresponding source. The BDD's value is then equal to the source value.

For a BDD P, let $\mathsf{len}(P)$ be its length (that is, the length of its longest path), $\mathsf{size}(P)$ be its size (that is, the number of non-terminal nodes). Let $\mathsf{BDD}(f)$ be the minimal size of any BDD computing f.

PrivateBDD Protocol. In [IP07], Ishai and Paskin proposed a new cryptocomputing method (PrivateBDD) that uses a BDD-representation of the target function in conjunction with a communication-efficient strong oblivious transfer. In [Lip09], the authors noted that the strong oblivious transfer protocol can be replaced by a BDD-friendly $(2, 1)$-CPIR protocol. We now briefly recall the main properties of PrivateBDD, as instantiated by Lipmaa's $(2, 1)$-CPIR from [Lip05], see [Lip09] for more.

Theorem 1. *Assume that the Decisional Composite Residuosity Assumption is true. Let $\mathcal{F}$ be a set of functions $f : \{0, 1\}^{\lambda} \to \{0, 1\}^{\lambda'}$, and for any $f \in \mathcal{F}$ let P_f be some (multi-terminal) BDD with λ'-bit sink labels that computes f. Let $\mathsf{len}(\mathcal{F}) := \max_{f \in \mathcal{F}} \mathsf{len}(f)$. Then $\mathcal{F}$ has a CPA-secure cryptocomputing protocol with communication upperbounded by $\kappa + \lambda \cdot (\lambda' + (\mathsf{len}(\mathcal{F}) + 2) \cdot \kappa)$, and server's online computation dominated by $\mathsf{size}(f)$ public-key operations.*

We omit internal details of the PrivateBDD. However, later we will use the fact that client's inputs to the PrivateBDD (when instantiated by Lipmaa's $(2, 1)$-CPIR from [Lip05]) are encrypted bitwise by using a length-flexible additively homomorphic public-key cryptosystem like [DJ01].

3 Generalized Selective Private Function Evaluation

In [CIK$^+$01], the authors consider the problem of m-out-of-n selective private function evaluation ((n, m)-SPFE), where the client obtains $g(f_{x_1}, \ldots, f_{x_m})$ of the database elements $f_{x_1}, \ldots, f_{x_m}$ for some function g. The authors proposed several ways to tackle this problem, but all their protocol needed an implementation of either a secure multi-party computation protocol or Yao's garbled circuits protocol.

In [BC09], the authors considered a related primitive that they called 1-out-of-n extended CPIR ($(n, 1)$-ECPIR). In such a protocol, the client obtains $g(f_x, y)$, where x, y are client's private inputs and f is server's private input. They proposed a number of protocols for a few functions g that could efficiently be implemented by using a homomorphic or a bilinear cryptosystem. Moreover, they only consider the case $m = 1$. On the other hand, the inclusion of the free input y makes ECPIR useful in a number of additional applications.

We define *generalized SPFE*, by letting the client to retrieve $g(f_{x_1}, \ldots, f_{x_m}, y)$, where y is client's auxiliary input and g may be a function that is private to the server. Generalized SPFE has a wider class of applications compared to vanilla SPFE, though it is also somewhat more difficult to implement.

In this section, we propose two protocols for generalized SPFE. The first one, presented in Sect. 3.1, works in the special case when m is a constant, has only 2 messages and is very efficient when $g(a, \cdot)$ has an efficient BDD representation for any constant a. The second, 4-message protocol (presented in Sect. 3.2) works for general m. In addition to having an augmented input y (which was not covered in [CIK$^+$01]), it is also usually more efficient than the protocols of [CIK$^+$01]. We note that one can construct more PrivateBDD-based SPFE protocols similarly to [CIK$^+$01] that proposed several different SPFE protocols based on generic multi-party computation. However, the protocol of Sect. 3.2 seems to be the most efficient one of the possible variations.

3.1 First Protocol: Efficient Generalized SPFE for Small m

Assume that m is constant, the client has inputs $(x_1, \ldots, x_m)$ and y, and the server has a database $f = (f_0, \ldots, f_{n-1})$. We construct the next protocol:

BDD-Based Generalized SPFE Protocol for Constant m

Client's inputs: $x_1, \ldots, x_m, y$.
Server's inputs: $f = (f_0, \ldots, f_{n-1})$, function g.
Client's output: $g(f_{x_1}, \ldots, f_{x_m}, y)$.

1. The client and the server run in parallel $\binom{n}{m}$ different PrivateBDD protocols to compute the values $g(f_{i_1}, \ldots, f_{i_m}, y)$ for every possible input $(i_1, \ldots, i_m)$. The server stores the database π, where $\pi_{(i_1,\ldots,i_m)} = g(f_{i_1}, \ldots, f_{i_m}, y)$.
2. In parallel to PrivateBDD protocols, the client and the server execute an $(\binom{n}{m}, 1)$-CPIR protocol to the database π, from which the client retrieves the correct element.

This generalized SPFE protocol takes 2 messages, computes n PrivateBDD protocols for g, and a single $(n, 1)$-CPIR protocol. Note that the client can "Q-encrypt" every bit of y once, since the server will reuse the same input in all n instances. Let $\mathsf{len}(P_g) := \max_i \mathsf{len}(P_{g,i})$ and $\mathsf{size}(P_g) := \max_i \mathsf{size}(P_{g,i})$ where $i = (i_1, \ldots, i_m)$ and $P_{g,x}$ is a BDD that implements $g(i_1, \ldots, i_m, \cdot)$. Thus, this generalized SPFE protocol has communication $\kappa + \lambda \cdot (\lambda' + (\mathsf{len}(P_g) + 2)\kappa)$ plus communication of a $(\binom{n}{m}, 1)$-CPIR on $(\lambda \cdot (\lambda' + (\mathsf{len}(P_g) + 2)\kappa))$-bit strings. In particular, when the Gentry-Ramzan CPIR [GR05] is used, this generalized SPFE protocol has communication $\kappa + \lambda \cdot (\lambda' + (\mathsf{len}(P_g) + 2)\kappa) + \Theta(m \cdot \log n + \lambda \cdot (\lambda' + (\mathsf{len}(P_g) + 2)\kappa) + \kappa) = \Theta(m \cdot \log n + \lambda \cdot (\lambda' + \mathsf{len}(P_g)\kappa))$. Server's online computation is dominated by $\Theta(\binom{n}{m} \cdot \mathsf{size}(P_g)) = \tilde{\Theta}(n^m \cdot \mathsf{size}(P_g))$ public-key operations.

Theorem 2. *Let the Decisional Composite Residuosity Assumption be true, and let the used $(n, 1)$-CPIR protocol be client-private. Then the BDD-based generalized SPFE protocol is private.*

Proof (Sketch.). One can prove client-privacy by a standard hybrid argument, by assuming that the PrivateBDD protocol and the $(n, 1)$-CPIR protocol are both client-private. For server-privacy, one can use any of the well-known transformations to transfer the $(n, 1)$-CPIR protocol and the $(2, 1)$-CPIR protocol used internally in the PrivateBDD protocol. The most efficient transformation in this case is most probably the one of [LL07]. In particular, that transformation only has to be applied to the instances of the $(2, 1)$-CPIR protocol that are on the very bottom of the BDD (that is, where the inputs to the CPIR protocol are the actual database elements). The resulting generalized SPFE protocol is clearly client-private. □

Example: Comparison Function. Let $m = 1$. As an example, suppose that the client wants to establish whether $y > f_x$, $y = f_x$ or $y < f_x$, where all values are λ'-bit long. One can construct a BDD for this with $\mathsf{len}(P) = \mathsf{size}(P) = \lambda'$. If one uses the Gentry-Ramzan $(n, 1)$-CPIR protocol [GR05], then this protocol has communication

$\Theta(\log n \cdot \lambda' \cdot \kappa)$, while server's online computation is dominated by $\Theta(n \cdot \lambda')$ public-key operations.

Another example, private similarity test, is studied more closely in Sect. 4.

3.2 Second Protocol

If m is not a constant, then we cannot use the same technique anymore because it could result in superpolynomial computation time. Moreover, the previous protocol requires to evaluate $g(a_1, \ldots, a_m, y)$, albeit for known values of a_j, $\binom{n}{m}$. Description of the second generalized (n, m)-SPFE protocol, that is more efficient for large values of m, follows:

BDD-Based Generalized SPFE Protocol

Client's inputs: $x_1, \ldots, x_m, y$.
Server's inputs: $f = (f_0, \ldots, f_{n-1})$, function g.
Client's output: $g(f_{x_1}, \ldots, f_{x_m}, y)$.

1. The client and the server execute an input selection protocol with the next inputs and outputs. The client has inputs $(x_1, \ldots, x_m)$. The server has inputs $(f, r_1, \ldots, r_m)$, where r_j is an λ-bit new random string generated by the server. The client obtains $(f_{x_1} \oplus r_1, \ldots, f_{x_m} \oplus r_m)$ while the server remembers $(r_1, \ldots, r_m)$. Note that after this protocol, the client and the server secret-share the inputs $f_{x_1}, \ldots, f_{x_m}$.
2. After the first step, the client and the server execute a PrivateBDD protocol with the next inputs: the client has inputs $(f_{x_1} \oplus r_1, \ldots, f_{x_m} \oplus r_m, y)$. The server has inputs $(g, r_1, \ldots, r_m)$. The client obtains $g(f_{x_1}, \ldots f_{x_m}, y)$.

To analyze the communication and computation complexity of the second generalized SPFE protocol, we have to specify both subprotocols. In the input selection protocol, the client and server execute an $(n, 1)$-CPIR protocol (for λ'-bit strings) m times in parallel, for $j \in \{1, \ldots, m\}$, with databases $(f_1 \oplus r_j, \ldots, f_m \oplus r_j)$ respectively. (This is similar to the input selection protocol of Sect. 3.3.1 of [CIK+01].)

Probably the simplest way to instantiate the PrivateBDD protocol on secret-shared inputs is to observe that when Lipmaa's $(2, 1)$-CPIR protocol [Lip05] is used, then client's inputs to the protocol are homomorphic encryptions of the bits $f_{x_i,j} \oplus r_{ij}$. Since the server knows the values r_{ij}, she can now easily obtain the encryptions of $f_{x_i,j}$. (For this, the server has to do expected $m\lambda/2$ additional divisions that use an encryption of 1.) After that, the server just has to compute PrivateBDD for the functionality that on inputs $(f_{x_1}, \ldots, f_{x_m}, y)$ outputs $g(f_{x_1}, \ldots, f_{x_m}, y)$.

This protocol requires 4 messages, and computes m parallel $(n, 1)$-CPIR protocols and a PrivateBDD protocol for $g(\cdots)$. As noted earlier, the server also has to execute 1 additional encryption and $m\lambda/2$ additional divisions. Thus, when the Gentry-Ramzan CPIR [GR05] protocol is used, this generalized SPFE protocol has communication $\Theta(m \cdot (\log n + \lambda' + \kappa) + \lambda \cdot (\lambda' + \mathsf{len}(P_g) \cdot \kappa))$. Server's online computation is dominated by $\Theta(mn + \mathsf{size}(P_g))$ public-key operations. For most of the interesting classes $\mathcal{F}$, the proposed SPFE protocol is much more efficient than any of the solutions proposed in [CIK$^+$01]. In particular, it provides sublinear communication in n whenever the *length* of the BDD is smaller than n.

Client-privacy is a straightforward corollary of the client-privacy of the PrivateBDD and the $(n, 1)$-CPIR protocol. The server-privacy on the other hand is a more difficult matter. First, by applying a conditional disclosure of secrets (CDS) protocol of [LL07] to client's input to the PrivateBDD, one can almost without no overhead guarantee that the client obtains a non-random input only if he has encrypted Boolean inputs.

If the client proves in zero-knowledge the consistency of those Boolean inputs with the output of the input selection protocol, then the second BDD-based generalized SPFE protocol is also server-private and thus semisimulatable. However, adding such a zero-knowledge proof will result in considerable overhead. Without zero-knowledge proofs, the only attack the client can do is to replace the inputs f_{x_j} to the PrivateBDD protocol with some values $f_{x_j} \oplus z_j$, with $z = (z_1, \dots, z_m)$ chosen by herself. Thus the second generalized SPFE protocol (when strengthened by the use of the CDS protocol of [LL07]) is a semisimulatably private protocol for the next functionality: client has private inputs $(x_1, \dots, x_m, y, z_1, \dots, z_m)$, server has private inputs $(f_0, \dots, f_{n-1}, g)$. The client obtains the value $g(f_{x_1} \oplus z_1, \dots, f_{x_m} \oplus z_m, y)$. We emphasize that this functionality is different from the SPFE functionality, however, several SPFE protocols from [CIK$^+$01] are secure in exactly the same sense. (The authors if [CIK$^+$01] said that in this case, the SPFE protocol is weakly secure.) We note however that this extended functionality may have it's own applications. Moreover, as also argued in [CIK$^+$01], the damage to server's privacy is still limited.

Thus we have proved that

Theorem 3. *Let the Decisional Composite Residuosity Assumption be true, and let the used $(n, 1)$-CPIR protocol be client-private. Then the second BDD-based generalized SPFE protocol is client-private and (weakly) server-secure.*

4 Private Similarity Test

As specified in the introduction, in a *private similarity test* protocol, the client wants to establish whether $w_h(y, f_x) < t$ (or equivalently, whether $||y - f_x||_2^2 < t$), where all values are λ'-bit long and w_h denotes the Hamming distance and $|| \cdot ||_2$ denotes the ℓ_2 norm. As usually, it is required that no more information is revealed.

Clearly, private similarity test can be seen as a special case of the SPFE protocol (for $m = 1$), and thus we first have to construct an efficient BDD that computes the predicate $[w_h(y, f_x) \overset{?}{<} t]$. In our case, the latter can be reduced to computing the threshold function $T_{\lambda,t}$, where $T_{\lambda,t}(v_1, \dots, v_\lambda) = 1$ iff at least t bits v_i are equal to 1. This is since client's message includes homomorphic encryptions of y's bits and the server knows fingerprint templates. Alternatively, fingerprint templates can be bitwise encrypted by the same key as the client's message — this is possible since they are used to branch in the BDD, and the branching variables may be encrypted. (See [IW06, CH08] for some related work on biometric authentication.)

It is straightforward to construct an (ordered) BDD for $T_{\lambda,t}$ that has $\Theta(\lambda^2)$ nodes and length λ. Currently, the smallest size BDD for threshold function was presented by Sinha and Thathachar in [ST97]. Their BDD has size $O(\lambda \cdot \log^3 \lambda/(\log\log\lambda \cdot \log\log\log\lambda))$, and length a$O(\lambda \cdot \log^2 \lambda/(\log\log\lambda \cdot \log\log\log\lambda))$. On the other hand, it is also known from [BPRS90] that $\mathsf{BDD}(f) = \Omega(\lambda \cdot \log\lambda/\log\log\lambda)$. Thus, the solution of [ST97] is close to the lower bound though not yet equal to it. We briefly sketch the idea of the construction from [ST97]. They choose co-prime integers $p_1, \dots, p_k$, for some k, such that $\prod p_j \geq n$ and $\sum p_j$ is minimal. They then use a Chinese Remainder Theorem-like construction to test the value of the input $v = (v_1, \dots, v_\lambda)$ modulo p_j for all p_j. This step can be implemented by a BDD of size $\Theta(\lambda \cdot \log^2 \lambda/(\log\log\lambda))$ and length $\Theta(\lambda \cdot \log\lambda/\log\log\lambda)$, and it reduces the threshold function for λ-bit inputs to a smaller interval which they then solve recursively. This recursion goes for $O(\log\lambda/\log\log\log\lambda)$ steps.

We now construct a private similarity test protocol by following the BDD-based generalized SPFE protocol of Sect. 3.1. Compared to the general protocol of Sect. 3.1, it is restricted to $m = 1$ and uses the specific BDD-s constructed in [ST97]:

Private Similarity Test

Client's inputs: $x \in \{0, \dots, n-1\}, y \in \{0,1\}^\lambda$.
Server's inputs: $f = (f_0, \dots, f_{n-1})$ with $f_j \in \{0,1\}^\lambda$.
Client's output: $[w_h(f_x, y) <^? t]$.

1. The client and the server run in parallel n different PrivateBDD protocols, based on [ST97], to compute the values $[w_h(f_i, y) <^? t]$ for every possible input i. The server stores the database π, where $\pi_i = [w_h(f_i, y) <^? t]$.
2. In parallel to PrivateBDD protocols, the client and the server execute an n-CPIR protocol to the database π, from which the client retrieves the correct element.

Combining the BDD of Sinha and Thathachar P with the Gentry-Ramzan CPIR protocol results in a private similarity test protocol with communication $\Theta(\log n + \lambda \cdot \mathsf{len}(P)\kappa) = \Theta(\log n + \lambda^2 \cdot \log^2 \lambda/(\log\log\lambda \cdot \log\log\log\lambda))\kappa$ and server's computation $\Theta(n \cdot \mathsf{size}(P)) = \Theta(n \cdot \lambda \cdot \log^3 \lambda/(\log\log\lambda \cdot \log\log\log\lambda))\kappa$ public-key operations, which is efficient for a large range of n and λ. Note that this is close to optimal communication-wise, because non-private similarity test has communication $\log n + \lambda + 1$.

Theorem 4. *Let the Decisional Composite Residuosity Assumption be true, and let the used $(n, 1)$-CPIR protocol be client-private. Then the new private similarity test is private.*

Proof. Simple corollary of the security of protocol in Sect. 3.1. □

Acknowledgments. The authors were supported by Estonian Science Foundation, grant #8058, the European Regional Development Fund through the Estonian Center of Excellence in Computer Science (EXCS), and the 6th Framework Programme project AEOLUS (FP6-IST-15964).

References

[BC09] Bringer, J., Chabanne, H.: Another Look at Extended Private Information Retrieval Protocols. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 305–322. Springer, Heidelberg (2009)

[BPRS90] Babai, L., Pudlák, P., Rödl, V., Szemerédi, E.: Lower Bounds to the Complexity of Symmetric Boolean Functions. Theoretical Computer Science 74(3), 313–323 (1990)

[CH08] Chmielewski, Ł., Hoepman, J.-H.: Fuzzy Private Matching. In: The Third International Conference on Availability, Reliability and Security, ARES 2008, Barcelona, Spain, March 4-7, pp. 327–334. IEEE Computer Society Press, Los Alamitos (2008)

[CIK$^+$01] Canetti, R., Ishai, Y., Kumar, R., Reiter, M.K., Rubinfeld, R., Wright, R.N.: Selective Private Function Evaluation with Applications to Private Statistics. In: PODC 2001, Newport, Rhode Island, USA, August 26-29, pp. 293–304. ACM Press, New York (2001)

[DJ01] 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)

[GR05] Gentry, C., Ramzan, Z.: Single-Database Private Information Retrieval with Constant Communication Rate. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 803–815. Springer, Heidelberg (2005)

[IP07] Ishai, Y., Paskin, A.: Evaluating Branching Programs on Encrypted Data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 575–594. Springer, Heidelberg (2007)

[IW06] Indyk, P., Woodruff, D.P.: Polylogarithmic Private Approximations and Efficient Matching. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 245–264. Springer, Heidelberg (2006)

[Lip05] Lipmaa, H.: An Oblivious Transfer Protocol with Log-Squared Communication. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 314–328. Springer, Heidelberg (2005)

[Lip09] Lipmaa, H.: How to Disassemble CPIR: First CPIR with Database-Dependent Computation. In: ICISC 2009, Seoul, Korea, December 2-4. LNCS. Springer, Heidelberg (2009)

[LL07] Laur, S., Lipmaa, H.: A New Protocol for Conditional Disclosure of Secrets And Its Applications. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 207–225. Springer, Heidelberg (2007)

[ST97] Sinha, R.K., Thathachar, J.S.: Efficient Oblivious Branching Programs for Threshold And Mod Functions. Journal of Computer and System Sciences 55(3), 373–384 (1997)

Optionally Identifiable Private Handshakes

Yanjiang Yang[1], Jian Weng[2,3], Jianying Zhou[1], and Ying Qiu[1]

[1] Institute for Infocomm Research, Singapore
{yyang,jyzhou,qiuying}@i2r.a-star.edu.sg
[2] Dept. of Computer Science, Jinan University, China
[3] School of Information Systems, Singapore Management University
cryptjweng@gmail.com

Abstract. It is now a trend that Internet users are increasingly concerned about individual privacy, and as a result numerous privacy-preserving authentication techniques have been proposed. In this paper, we propose the concept of *private handshakes with optional identifiability*, which allows the two users in a handshake deciding *real time* to either proceed their interaction as secret handshake or as private handshake. Such optionally identifiable private handshakes are a more flexible privacy-preserving authentication primitive than secret handshakes and private handshakes. We formulate a formal definition for optionally identifiable private handshakes, as well as a set of security definitions, and propose a concrete scheme. We implement a proof-of-concept prototype of the proposed scheme, on top of the widely used TLS protocol.

1 Introduction

Nowadays, as Internet has grown to be an indispensable part of our society, more and more services/transactions are becoming electronic, e.g., e-commerce and e-banking. Users, on the one hand, fervently embrace the use of Internet and enjoy the great convenience it brings, while on the other hand, become increasingly conservative in disclosing individual information when using Internet. As a result, *privacy-preserving techniques* that can make users accomplish the desired functionalities and at the same time maintain their individual privacy are expected to play a key part in a wider adoption of Internet applications.

Secret handshake protocols are a privacy-preserving authentication primitive that enables a pair of users from the same group, each holding a group credential, to authenticate each other, while guarantee that 1) non-members learn nothing on the handshake between the two users including whether they recognize each other and whether they belong to the same group; 2) a non-member cannot pretend to be a member, and in turn perform handshakes with members. Secret handshakes turn out to be quite useful, especially at the time when users are more and more concerned about individual privacy, reluctant to reveal their activities and preferences over Internet. Therefore, since first formulated by Balfanz *et al.* [3], secret handshakes have attracted enormous attention.

In their original form, secret handshakes are *linkable* in the sense that different handshakes by the same user can be linked. This is because a user needs to repeatedly use the same pseudonym/ID in different handshake sessions. However, *unlinkability* is often a pursued goal for privacy-preserving protocols, *unlinkable secret handshakes* were thus proposed, e.g., [12]. We should point out that it trivially achieves unlinkability using any secret handshake protocol, as long as a user can have in possession an indefinite number of pseudonyms/IDs (and credentials). Note that to use a pseudonym/ID for secret handshakes, there must be a credential bound to that pseudonym/ID. Hence this trivial approach is not practical, since it requires a user to apply for an ample number of credentials from the group administrator. In contrast, unlinkable secret handshakes attain unlinkability by each user using a single reusable credential. We here should also clarify the differences between unlinkable secret handshake protocols and other commonly used privacy-preserving authentication primitives such as group signature (e.g., [2]) and anonymous credential (e.g., [9]): unlinkable secret handshakes are affiliation-hiding in the sense that non-members cannot learn to which group the users in handshakes belong; but group signature and anonymous credential do not hide the users' affiliation at all.

Among others, *traceability* is a property offered by unlinkable secret handshakes. Traceability allows a designated group administrator to find out the users who have engaged in a secret handshake session (e.g., in case of certain exceptional events), based on the protocol transcript. Although traceability is often a desired feature in unlinkable privacy-preserving systems, possession of it actually weakens privacy protection of the underlying system: anyhow a certain party (group administrator in the setting of unlinkable secret handshakes), however trustful it is, can violate unlinkability. Recently, Hoepman [13] proposed the notion of *private handshakes*, which is essentially unlinkable secret handshakes without traceability. Offering no traceability whatsoever, private handshakes can be useful in certain circumstances, e.g, a user in some applications may not want anyone to identify him in any means.

Our Contribution: In this work, we propose the concept of *private handshakes with optional identifiability*, which allows the two users in a handshake to negotiate *real time* whether to make their handshake identifiable. To make it clearer, optionally identifiable private handshakes can be viewed as a primitive interpolating between private handshakes and secret handshakes. As shown in Figure 1, if the two users agree to achieve no identifiability, then the protocol proceeds as a private handshake; on the contrary, if they decide to achieve identifiability, then the protocol proceeds as a secret handshake. Optionally identifiable private handshakes clearly offer the users the flexibility to choose the level of privacy protection in their handshake, i.e., total privacy or identifiability.

The differences between optionally identifiable private handshakes and unlinkable secret handshakes are substantial, and are explained as follows. Unlinkable secret handshakes provide unlinkability to the two users in a handshake, but the designated group administrator can definitely revoke unlinkability. In comparison, for an optionally identifiable private handshake protocol, if it proceeds as a private handshake, then unlinkability is provided and cannot be revoked by any

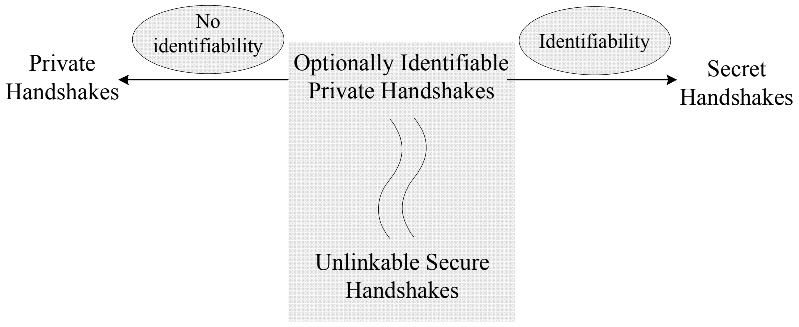

No identifiability
Identifiability
Private Handshakes
Optionally Identifiable Private Handshakes
Secret Handshakes
Unlinkable Secure Handshakes

Fig. 1. Concept of optionally identifiable private handshakes

party; on the other hand, if it proceeds as a secret handshake, then the users in the handshake can directly violate unlinkability.

Finally, the properties of optionally identifiable private handshakes can be summarized informally as follows. (1) *Membership authentication*: users from the same group are guaranteed to authenticate each other. (2) *Non-member exclusiveness*: a user Alice who does not belong to a group pretending to be a member is not able to successfully perform handshakes with a genuine group member Bob; she cannot even learn anything on her counterpart, including whether he belongs to that group or not. (3) *Optional identifiability*: the two users in a handshake can decide real time whether to engage in identifiable or non-identifiable handshake: if the former (i.e., secret handshake), the two learn their respective counterparts' pseudonyms/identities; otherwise, the handshake is entirely unlinkable to them, without any traceability whatsoever (i.e., private handshake). (4) *Unlinkability*: a non-member eavesdropper cannot tell apart the protocol sessions involving the same user from those involving different users. Better still, if the two group members engage in a non-identifiable handshake (i.e., private handshake), they cannot differentiate their respective counterparts either.

Organization: The rest of the paper is organized as follows. In Section 2, we review the related work. We then formulate a model for optionally identifiable private handshakes in Section 3, followed by a concrete scheme in Section 4. Security definitions for optionally identifiable private handshakes are formulated in Section 5, together with the security proofs for the proposed scheme. In Section 6, we report the implementation results, and Section 7 concludes the paper.

2 Related Work

The notion of secret handshakes traces back to private match making [5], where users with the same "target" can locate and authenticate each other secretly. However, in private match making, it is likely that any user can identify members if he correctly guesses the "target".

Balfanz *et al.* [3] first formulated the notion of secret handshakes and revived the interest in this privacy-preserving authentication primitive. Their protocols are based on bilinear parings, and secure under the bilinear Diffie-Hellman assumption [4] and the random oracle model [7]. Subsequently, Castelluccia *et al.* [8] proposed secret handshake protocols, with security under computational Diffie-Hellman assumption. RSA-based secret handshake protocols were due to Jarecki *et al.* [11] and Vergnaud [17].

The above secret handshake protocols are inherently linkable, unless using distinct credentials each time. Achieving unlinkability directly using these protocols requires a user to possess an indefinite number of credentials, which is unlikely to be affordable in practice. A more satisfactory solution is that a user is able to reuse his credential while attaining unlinkability. Xu and Yung [18] achieved the use of reusable credentials in secret handshakes. However, their scheme only achieves k-anonymity, which allows the attacker to learn that a user in a handshake is from one of the k publicly known users. Tsudik and Xu [16] proposed a protocol achieving (full) unlinkability, but all members from the same group are required to share a group secret. One of the main drawbacks of sharing secret is that the propagation of revocation information must be strictly synchronized; otherwise, some members will fail to authenticate. Jarechi and Liu's unlinkable secret handshake protocol [12] does not rely on group members sharing secret, and it also tolerates to some extent unsynchronized propagation of revocation information. Ateniese *et al.* [1] extended unlinkable secret handshakes in several ways, e.g., allowing members from different groups to perform handshakes, and supporting fuzzy attribute-based handshakes.

Unlinkable secret handshakes allow the designated group administrator to violate unlinkability, such that the administrator can find out the members who have encaged in the handshakes. Traceability is often a useful feature in privacy-preserving systems. Nevertheless, in some applications users may not be happy to be traced by any one. This has motivated Hoepman [13] to propose the concept of private handshakes, which is essentially unlinkable secret handshakes, but does not have traceability to any party.

The concept of optionally identifiable private handshakes we propose should be viewed as a primitive interpolating between private handshakes and secret handshakes, allowing the two users in a handshake to decide *real time* either to proceed their handshake as identifiable or to proceed as non-identifiable. It is thus a more flexible handshake primitive than private handshakes and secret handshakes. Optionally identifiable private handshakes are also superior to unlinkable secret handshakes, in the sense that they do not implicate a trusted third party (i.e., the group administrator) for unlinkability revocation.

3 Model

An optionally identifiable private handshake system consists of a set $\mathbb{G}$ of groups, a set $\mathbb{U}$ of users, and a set $\mathbb{A}$ of group administrators who create groups and enrol users in groups. A user may or may not be affiliated to a group (for simplicity, we assume that a user belongs to at most one group). If a user belongs to a group, then he is a *member* of that group; otherwise, he is *non-member* of that group. An optionally identifiable private handshake system consists of the following algorithms.

- $\mathsf{CreateGroup}(1^\kappa) \to \{0,1\}^*$: On input a security parameter 1^κ, the algorithm, executed by a group administrator $A \in \mathbb{A}$, outputs a group secret $s_G \in \{0,1\}^*$ for a group $G \in \mathbb{G}$.
- $\mathsf{EnrolUser}(G, u) \to \{0,1\}^*$: On input a group $G \in \mathbb{G}$ and a user $u \in \mathbb{U}$, the algorithm, executed by the group administrator A_G of G, outputs a secret credential $x_u \in \{0,1\}^*$ bound to the user's identity/pseudonym u. Note that x_u is generated using the group secret s_G.
- $\mathsf{Handshake}(u_1, u_2, b) \to \{0,1\}^* \cup \{\perp\}$: This is an interactive process between two users $u_1, u_2 \in \mathbb{U}$, governed by $b \in \{0,1\}$. If $b = 0$, the interactions between u_1 and u_2 constitute a private handshake protocol; otherwise, the interactions are a secret handshake protocol. In either case, the algorithm outputs a shared key $sk \in \{0,1\}^*$ between u_1 and u_2 if they belong to the same group, or $\perp$ otherwise.
- $\mathsf{RevokeUser}(G, u) \to \{0,1\}^*$: On input a group $G \in \mathbb{G}$ and a user $u \in \mathbb{U}$, the algorithm, executed by the group administrator A_G of G, revokes the membership of u, and inserts u into the RevokedUserList of G and outputs the RevokedUserList.

4 Our Construction

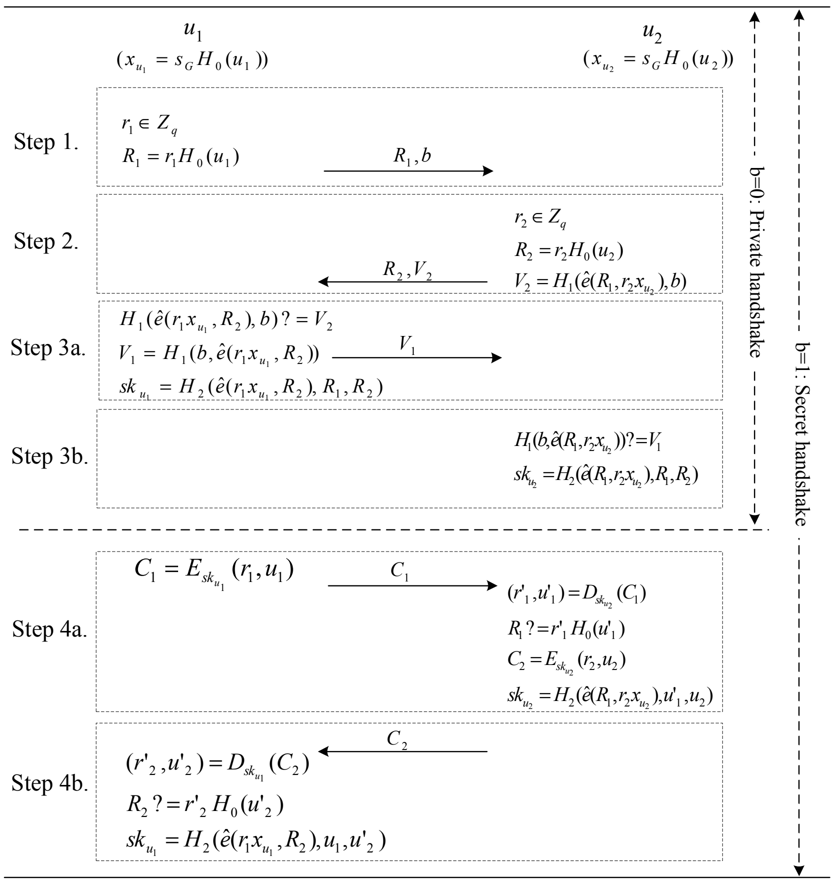
4.1 Preliminaries

Our construction is based on bilinear parings. Let $\mathcal{G}_1, \mathcal{G}_2$ be two cyclic groups of a large prime order q, then $\hat{e} : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$ is a bilinear pairing if for any $a, b \in Z_q, P, Q \in \mathcal{G}_1$, we have $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$. A bilinear pairing should be non-degenerate, i.e., if P, Q are generators in $\mathcal{G}_1$, then $\hat{e}(P, Q)$ is not the identity in $\mathcal{G}_2$. In bilinear pairings, the *Bilinear Diffie-Hellman* (BDH) problem is assumed to be hard, which states that it is hard to compute $\hat{e}(P, P)^{abc}$ from aP, bP, cP for random $a, b, c \in Z_q$ and $P \in \mathcal{G}_1$. Formally, for all probabilistic polynomial time (PPT) $\mathcal{A}$, we define $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{BDH}} = \Pr[\mathcal{A}(aP, bP, cP) = \hat{e}(P, P)^{abc}]$ to be the advantage $\mathcal{A}$ solves the BDH problem, which is negligible in the security parameter.

4.2 Details of Protocol

The system parameters include $\hat{e} : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$ defined as above, cryptographic hash functions $H_0 : \{0,1\}^* \to \mathcal{G}_1, H_1 : \mathcal{G}_2 \times \{0,1\} \to \mathcal{K}, H_2 : \mathcal{G}_2 \times \mathcal{G}_1^2 \to \mathcal{K}$, and a semantically secure symmetric key encryption $E : \{0,1\}^* \times \mathcal{K} \to \{0,1\}^*$ (the decryption algorithm is $D : \{0,1\}^* \times \mathcal{K} \to \{0,1\}^*$), where $\mathcal{K}$ denotes the appropriate key space.

– CreateGroup(1^κ): Given the security parameter 1^κ, a group administrator A creates a group G by selecting a random secret $s_G \in Z_q$, specific to G. Note that the size of q polynomially relates to κ. Then A is the group administrator of G, denoted as A_G.
– EnrolUser(G, u): To enrol user u into group G whose group secret is s_G, the corresponding group administrator A_G computes and issues u a credential $x_u = s_G H_0(u)$, where u denotes the identity/pseudonym of the user.
– Handshake(u_1, u_2, b): The interactions between u_1, u_2, whose credentials are $x_{u_1} = s_G H_0(u_1)$ and $x_{u_2} = s_G H_0(u_2)$, respectively, are the following, aiming to authenticate each other and establish a common key for their subsequent communication. Figure 2 summarizes the details of the procedure.



u_1
$(x_{u_1} = s_G H_0(u_1))$
u_2
$(x_{u_2} = s_G H_0(u_2))$
Step 1.
$r_1 \in Z_q$
$R_1 = r_1 H_0(u_1)$
R_1, b
Step 2.
$r_2 \in Z_q$
$R_2 = r_2 H_0(u_2)$
$V_2 = H_1(\hat{e}(R_1, r_2 x_{u_2}), b)$
R_2, V_2
Step 3a.
$H_1(\hat{e}(r_1 x_{u_1}, R_2), b) ? = V_2$
$V_1 = H_1(b, \hat{e}(r_1 x_{u_1}, R_2))$
V_1
$sk_{u_1} = H_2(\hat{e}(r_1 x_{u_1}, R_2), R_1, R_2)$
Step 3b.
$H_1(b, \hat{e}(R_1, r_2 x_{u_2}))? = V_1$
$sk_{u_2} = H_2(\hat{e}(R_1, r_2 x_{u_2}), R_1, R_2)$
Step 4a.
$C_1 = E_{sk_{u_1}}(r_1, u_1)$
C_1
$(r'_1, u'_1) = D_{sk_{u_2}}(C_1)$
$R_1 ? = r'_1 H_0(u'_1)$
$C_2 = E_{sk_{u_2}}(r_2, u_2)$
$sk_{u_2} = H_2(\hat{e}(R_1, r_2 x_{u_2}), u'_1, u_2)$
Step 4b.
C_2
$(r'_2, u'_2) = D_{sk_{u_1}}(C_2)$
$R_2 ? = r'_2 H_0(u'_2)$
$sk_{u_1} = H_2(\hat{e}(r_1 x_{u_1}, R_2), u_1, u'_2)$
b=0: Private handshake
b=1: Secret handshake

Fig. 2. Details of Handshake(u_1, u_2, b)

Step 1. Let u_1 be the initiator. u_1 chooses a random number $r_1 \in Z_q$, and computes and sends $R_1 = r_1 H_0(u_1)$ to u_2, together with a bit b. Note

that if $b = 0$, then it means u_1 wants to engage in a private handshake with u_2 (non-identifiability); otherwise it means u_1 wants a secret handshake (identifiability):
$u_1 \longrightarrow u_2 : R_1, b$

Step 2. Upon receiving the message, if u_2 does not agree with u_1 on the type of handshake indicated by b, u_2 aborts. Otherwise, u_2 chooses a random number $r_2 \in Z_q$, computes $R_2 = r_2 H_0(u_2)$ and $V_2 = H_1(\hat{e}(R_1, r_2 x_{u_2}), b) = H_1(\hat{e}(H_0(u_1), H_0(u_2))^{s_G r_1 r_2}, b)$, and responds to u_1 with R_2, V_2:
$u_2 \longrightarrow u_1 : R_2, V_2$

Step 3a. u_1 first checks whether $H_1(\hat{e}(r_1 x_{u_1}, R_2), b) = V_2$. If the equation does not hold, u_1 aborts. Otherwise, u_1 computes and sends $V_1 = H_1(b, \hat{e}(r_1 x_{u_1}, R_2)) = H_1(b, \hat{e}(H_0(u_1), H_0(u_2))^{s_G r_1 r_2})$ to u_2, and at the same time computes $sk_{u_1} = H_2(\hat{e}(r_1 x_{u_1}, R_2), R_1, R_2)$:
$u_1 \longrightarrow u_2 : V_1$

Step 3b. Upon receipt of V_1, u_2 checks whether $H_1(b, \hat{e}(R_1, r_2 x_{u_1})) = V_1$. If the equation does not hold, u_2 aborts. Otherwise, u_2 computes $sk_{u_2} = H_2(\hat{e}(R_1, r_2 x_{u_2}), R_1, R_2)$.
It is clear that, at this point u_1 and u_2 have authenticated each other if they belong to the same group, and the protocol has accomplished private handshake. Thus if $b = 0$, then the protocol stops, and sk_{u_1}, sk_{u_2} are the shared key between u_1 and u_2. It is easily verified that $sk_{u_1} = sk_{u_2} = H_2(\hat{e}(H_0(u_1), H_0(u_2))^{s_G r_1 r_2}, R_1, R_2)$.

However, if $b = 1$, u_1 and u_2 continue to send to each other their identity and the random numbers r_1 and r_2, encrypted by sk_{u_1}, sk_{u_2}, respectively, for achieving identifiability. In particular:

Step 4a. $u_1 \longrightarrow u_2 : C_1 = E_{sk_{u_1}}(r_1, u_1)$
Upon receiving u_1's message, u_2 decrypts it to get r_1', u_1', and then checks whether $R_1 = r_1' H_0(u_1')$. If it holds, u_2 accepts and computes a new key $sk_{u_1} = H_2(\hat{e}(R_1, r_2 x_{u_2}), u_1', u_2)$. At the same time, u_2 sends to u_1:
Step 4b. $u_2 \longrightarrow u_1 : C_2 = E_{sk_{u_2}}(r_2, u_2)$
Likewise, u_1 decrypts and gets r_2', u_2', then checks whether $R_2 = r_2' H_0(u_2')$. If it holds, u_1 accepts and computes a new key $sk_{u_1} = H_2(\hat{e}(r_1 x_{u_1}, R_2), u_1, u_2')$. This completes the whole handshake procedure.

– $\mathsf{RevokeUser}(G, u)$: To enable user revocation, we assume that in $\mathsf{CreateGroup}$, the group administrator A_G of G has additionally chosen a t-degree polynomial $f(x) \in F_q[x]$, where t is the maximum possible number of users in the group, and each enroled user u has been issued $f(u)$. As such, to revoke a user $\bar{u}$ from G (suppose there were already m revoked users $\bar{u}_{i_1}, \bar{u}_{i_2}, \cdots, \bar{u}_{i_m}$ before $\bar{u}$), A_G chooses $s_G' \in Z_q$, computes and broadcasts (or publishes in a public bulletin board) $g(x) = (x - \bar{u})(x - \bar{u}_{i_1}) \cdots (x - \bar{u}_{i_m})(x-1)^{t-m-1}.s_G' + f(x) \in F_q[x]$; at the same time, A_G includes $\bar{u}$ in the RevokedUserList and publishes the updated RevokedUserList. It is clear that each non-revoked user u can recover $s_G' = (g(u) - f(u))/(\prod_{u' \in \text{RevokedUserList}}(x - u'))(x-1)^{t-m-1}$, while revoked users cannot. The reason why a revoked user $\bar{u}$ cannot compute s'_G is that $g(\bar{u}) = f(\bar{u})$. With s'_G in possession, each non-revoked user u then updates his credential as $x_u = s'_G x_u$.

4.3 Discussions

In the above handshake protocol, when $b = 1$ the two users u_1, u_2 need to exchange their identities/pseudonyms. There may be a "fairness" issue in the exchange. For example, after getting C_1 from u_1 in Step 4a., u_2 stops without sending C_2 as required, in which case u_2 gains advantage over u_2 in terms of identifiability. It seems to us that there is no good technical solution to this problem without involving a trusted third party.

The bright side is that in secret handshake (recall that when $b = 1$ the above protocol is secret handshake), an underlying assumption is that a user does not mind disclosing pseudonym/identity to his counterpart in a handshake. Therefore, it should be understood that there is no plausible motive for a user to deliberately terminate the protocol prematurely. Furthermore, be noted that the handshake procedure simply helps the two users establish a shared key for the subsequent communication; as such, if a user deviates from the protocol to gain advantages, he will be rejected by his victim counterpart for further communication.

Another issue in our scheme is the requirement of synchronized propagation of the revocation information in RevokeUser. As a matter fact, user revocation is a unsolved problem in almost all existing unlinkable secret handshake schemes, except the one by Jarechi and Liu [12], which to some extent admits unsynchronized propagation of revocation information. We leave the better solution to user revocation in our scheme to future work.

5 Security Definitions and Proofs

We first formulate security definitions for optionally identifiable private handshakes, and then prove that our proposed scheme satisfies the security definitions. The strategies for formulation and proofs are largely based on those in [3].

5.1 Security Definitions

We first review the definition for negligible function.

Definition 1. (Negligible function). *A function $\epsilon(\kappa)$ is negligible in κ if for all polynomials p(.), $\epsilon(\kappa) \leq 1/p(\kappa)$ for sufficiently large κ, where κ is the security parameter.*

Impersonation Resistance. The first security definition for optionally identifiable private handshake is *impersonation resistance*, which relates to the authentication property of the handshake protocol. Informally, impersonation resistance stipulates that only members from the same group can perform a handshake with each other, and non-members cannot impersonate legitimate members of the group. Impersonation resistance is defined through an *Impersonation Game*, where an adversary $\mathcal{A}$ tries to learn how to impersonate members of a certain group G^*: $\mathcal{A}$ interacts with players of the system, corrupts some users, communicates with legitimate members of G^*, and eventually picks a target user $u^* \in G^*$ and attempts to convince u^* that it is a member of G^*. Intuitively, if $\mathcal{A}$ does not corrupt any user $u \in G^*$, it is unable to convince u^* of its membership. In particular, the impersonation game for a Probabilistic Polynomial Time (PPT) $\mathcal{A}$ is as follows.

Step 1. $\mathcal{A}$ interacts with users of its choice, and corrupts some users $U' \subset \mathbb{U}$. To corrupt a user, $\mathcal{A}$ obtains the credential of that user.

Step 2. $\mathcal{A}$ chooses a target user u^* satisfying $u^* \notin U'$ and $u^* \in G^*$.

Step 3. Finally, $\mathcal{A}$ tries to convince u^* that $\mathcal{A} \in G^*$, that is, $\mathcal{A}$'s objective is to pass the verification checks imposed by u^* by engaging in $\mathsf{Handshake}(\mathcal{A}, u^*, b)$.

We say $\mathcal{A}$ *wins the impersonation game* if it indeed passes u^*'s verification checks. We define $\mathcal{A}$'s impersonation advantage $\text{Adv}_{\mathcal{A}}^{\text{Impersn}}$ to be the probability that $\mathcal{A}$ wins the impersonation game on the condition that $\mathcal{A}$ has not corrupted any user in G^*, i.e., $\text{Adv}_{\mathcal{A}}^{\text{Impersn}} = \Pr[\mathcal{A} \text{ wins impersonation game} \mid U' \cap G^* = \varnothing]$.

Definition 1. (Impersonation resistance). *An optionally identifiable private handshake scheme achieves impersonation resistance, if* $\text{Adv}_{\mathcal{A}}^{\text{Impersn}}$ *is negligible for all PPT* $\mathcal{A}$.

Membership Detection Resistance. The second security definition is *membership detection resistance*, which informally requires that a non-member of a group cannot identify members of that group. Membership detection resistance is defined through a *membership detection Game*, where an adversary $\mathcal{A}$ tries to learn how to detect members of a certain group G^*: $\mathcal{A}$ interacts with players of the system, corrupts some users, picks a target user u^*, and attempts to detect whether $u^* \in G^*$. Intuitively, it should not be possible for $\mathcal{A}$ to decide whether or not $u^* \in G^*$ unless $\mathcal{A}$ has corrupted some $u \in G^*$. The details of the membership detection game for a PPT $\mathcal{A}$ is the following.

Step 1. $\mathcal{A}$ interacts with users of its choice, and corrupts some users $U' \subset \mathbb{U}$ and gets the credentials of the corrupted users.

Step 2. $\mathcal{A}$ chooses a target user $u^* \notin U'$.

Step 3. A random bit δ is flipped. If $\delta = 0$, $\mathcal{A}$ interacts with u^*. If $\delta = 1$, $\mathcal{A}$ interacts with a random simulation of u^*. A *random simulation* of a player in a protocol is defined to replace all outgoing messages of that player with uniformly-random bit strings of the same length.

Step 4. Finally, $\mathcal{A}$ outputs a bit δ', for a guess on δ.

We say $\mathcal{A}$ *wins the membership detection game* if $\delta = \delta'$. We define $\mathcal{A}$'s membership detection advantage as $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{MemDetec}} = |\Pr[\mathrm{A}$ wins membership-detection game $| U' \cap G^* = \varnothing] - 1/2|$.

Definition 2. (Membership detection resistance). *An optionally identifiable private handshake scheme achieves membership detection resistance, if* $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{MemDetec}}$ *is negligible for all PPT $\mathcal{A}$.*

Unlinkability of Private Handshake. Recall that when $b = 0$, the optionally identifiable private handshake protocol actually performs private handshake. We next define the *unlinkability* property of private handshake through a *Unlinkability Game*, where an adversary $\mathcal{A}$ attempts to recognize the individual user of G^*, who interacts with $\mathcal{A}$: $\mathcal{A}$ corrupts some users of G^*, picks a target user $u^* \in G^*$, and tries to recognize whether the interacting counterpart is u^*. Specifically, the unlinkability game for a PPT $\mathcal{A}$ is as follows.

Step 1. $\mathcal{A}$ corrupts some users $U' \subset G^*$ and gets the credentials of the corrupted users.

Step 2. $\mathcal{A}$ chooses a target user $u^* \in G^*$.

Step 3. A random bit δ is flipped. If $\delta = 0$, $\mathcal{A}$ is set to engage in $\mathsf{Handshake}(\mathcal{A}, u^*, b = 0)$ with u^*. If $\delta = 1$, $\mathcal{A}$ is set to perform $\mathsf{Handshake}(\mathcal{A}, u, b = 0)$ with a random $u \in G^*, u \neq u^*$.

Step 4. Finally, $\mathcal{A}$ outputs a bit δ', for a guess on δ.

We say $\mathcal{A}$ *wins the unlinkability game* if $\delta = \delta'$. We define $\mathcal{A}$'s linkability advantage as $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Unlnk}} = |\Pr[\mathrm{A}$ wins the unlinkability game$] - 1/2|$.

Definition 3. (Unlinkability of private handshakes). *An optionally identifiable private handshake scheme performs private handshake when $b = 0$, if* $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Unlnk}}$ *is negligible for all PPT $\mathcal{A}$.*

Unlinkability to Eavesdroppers. Performing either private handshake or secret handshake, an optionally identifiable private handshake scheme should remain unlinkable to passive eavesdroppers who simply watches interactions between users. It should be noted that unlinkability of private handshake defined above already implies unlinkability to eavesdroppers in private handshake, since a passive eavesdropper is by no means more powerful than the adversary considered in the above unlinkability game. Hence, below our focus is only on unlikability to eavesdroppers in secret handshake.

Consider an adversary $\mathcal{A}$ corrupting some users U' of its choice, observes an interaction between $u_1^*, u_2^* \notin U'$. *Unlinkability to eavesdroppers* stipulates that $\mathcal{A}$ is not able to learn anything beyond what it already knows, including whether or not u_1^* and u_2^* belong to the same group. We define $\mathcal{A}$'s linkability advantage as follows. Let $Tr_{u_1^*,u_2^*}$ be the actual protocol transcript of $\mathsf{Handshake}(u_1^*, u_2^*, b = 1)$, and Tr_{R_1,R_2} be the resulting transcript of R_1 and R_2, where R_1 and R_2 are random simulations of u_1^*, u_2^*, respectively (recall the definition of random simulation in the membership detection game). Then $\mathcal{A}$'s linkability advantage is $\mathrm{Adv}_{\mathcal{A}}^{\text{Eav-Lnk}} = |\Pr[\mathcal{A}(Tr_{u_1^*,u_2^*}) = 1] - \Pr[\mathcal{A}(Tr_{R_1,R_2}) = 1]|$.

Definition 4. (Unlinkability to eavesdroppers). *An optionally identifiable private handshake scheme achieves unlinkability to eavesdroppers, if* $\mathrm{Adv}_{\mathcal{A}}^{\text{Eav-Lnk}}$ *is negligible for all PPT* $\mathcal{A}$. *That is,* $\mathcal{A}$ *is unable to distinguish between the actual interactions between* u_1^*, u_2^* *and the interactions resulting from the random simulation of* u_1^*, u_2^*.

5.2 Security Proofs

We now prove that our proposed scheme satisfies the above security definitions. In our analysis, we model H_0, H_1, H_2 as random oracles [7]. Let $\mathcal{A}$ be a PPT adversary, we denote $Q_{H_0}, Q_{H_1}, Q_{H_2}$ be the number of distinct queries asked by $\mathcal{A}$ to H_0, H_1, H_2, respectively. We use e to denote the base of the natural logarithm, i.e., $e \approx 2.78$.

Theorem 1. (Impersonation resistance) *Suppose* $\mathcal{A}$ *is a PPT adversary in breaking impersonation resistance of our scheme. Then there is a PPT algorithm* $\mathcal{B}$ *such that* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Impersn}} \leq e.Q_{H_0}.Q_{H_1}.\mathsf{Adv}_{\mathcal{B}}^{\mathsf{BDH}} + \epsilon$, *where recall that* $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{BDH}}$ *is the advantage with which* $\mathcal{B}$ *solves the BDH problem, and* ϵ *is a negligible function.*

The proof of the theorem is largely based on that in [3], and quite lengthy. Due to the space constraint, we omit the details, which can be found in the full paper [19].

Theorem 2. (Membership detection resistance) *Suppose* $\mathcal{A}$ *is a PPT adversary in breaking membership detection resistance of our scheme. Then there is a PPT algorithm* $\mathcal{B}$ *such that* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{MemDetec}} \leq e.Q_{H_0}.Q_{H_1}.\mathsf{Adv}_{\mathcal{B}}^{\mathsf{BDH}} + \epsilon$, *where* $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{BDH}}$ *is the advantage that* $\mathcal{B}$ *solves the BDH problem, and* ϵ *is a negligible function.*

Proof sketch. It suffices to prove $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{MemDtec}} \leq Q_{H_1}.\mathsf{AdvSolvBDH}_{\mathcal{B}} + \epsilon$, based on the above "SolvBDH" game. The proof is similar to that for Theorem 1, with the following minor changes: we claim that $\mathcal{A}$ must have queried m^* to H_1 in order to distinguish u^*'s actual involvement in the handshake from the random simulation of u^*, rather than claiming that $\mathcal{A}$ queried m^* for constructing V_1 in Theorem 1's proof. The rest of the analysis is largely the same. □

Theorem 3. (Unlinkability of private handshakes) *Let* $\mathcal{A}$ *be a PPT adversary in the Unlinkability Game with respect to our scheme. Then* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Unlnk}} = 0$.

Proof sketch. The theorem actually says that our scheme achieves unlinkability (when $b = 1$) unconditionally. To prove this, we need to show that R_2, V_2 are truly random to $\mathcal{A}$. Since V_2 does not reveal more information than R_2, as it is derived from R_2, it suffices to show that $\mathcal{A}$ cannot distinguish between R_2 and a random element in G_1. Recall that G_1 is a cyclic group with prime order q, so every element in the group is a generator. Therefore, the base of R_2 with respect to any r_2 is indeed a uniformly random generator. □

Theorem 4. (Unlinkability to eavesdroppers) *If cryptographic hash functions H_0, H_1, H_2 are pseudorandom functions, and the symmetric key encryption scheme E is semantically secure, then for any PPT $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{Eav\text{-}Lnk} \leq \epsilon$.*

Proof sketch. It is suffices to construct a PPT simulator $\mathcal{A}^*$ such that, for any u_1 and u_2, for all PPT adversaries $\mathcal{A}$, $\mathcal{A}^*$ can simulate $\mathcal{A}(Tr_{u_1,u_2})$ with non-negligible probability, where Tr_{u_1,u_2} is the protocol transcript of u_1 and u_2 in $\mathsf{Handshake}$. In particular, we show that $\mathcal{A}^*(1^\kappa)$ can generate a view $Tr^*_{u_1,u_2}$, which is computationally indistinguishable from Tr_{u_1,u_2}. Let us only consider when $b = 0$: recall that

$$Tr_{u_1,u_2} = \left\{ \begin{array}{c} R_1 = r_1 H_0(u_1) \\ b \\ R_2 = r_2 H_0(u_2) \\ V_2 = H_1(\hat{e}(H_0(u_1), H_0(u_2))^{s_G r_1 r_2}, b) \\ V_1 = H_1(b, \hat{e}(H_0(u_1), H_0(u_2))^{s_G r_1 r_2}) \\ C_1 = E_{sk}(u_1, r_1) \\ C_2 = E_{sk}(u_2, r_2)] \end{array} \right\}.$$

$\mathcal{A}^*$ generates $Tr^*_{u_1,u_2} = [R_1^*, b, R_2^*, V_2^*, V_1^*, C_1^*, C_2^*]$ as follows. Picks random elements R_1^*, R_2^* from G_1; picks a random s_G^* from Z_q and sets $V_2^* = H_1(\hat{e}(R_1^*, R_2^*)^{s_G^*}, b)$, $V_1^* = H_1(b, \hat{e}(R_1^*, R_2^*)^{s_G^*})$; picks random C_1^* and C_2^* from $\{0,1\}^*$. It is easy to check that R_1, R_2 are unconditionally indistinguishable from R_1^*, R_2^*; V_1, V_2 are computationally indistinguishable from V_1^*, V_2^* if H_1 is a pseudorandom function; C_1, C_2 are computationally indistinguishable from C_1^*, C_2^* as long as the symmetric encryption E is semantically secure. This completes the proof. □

6 Implementation Results

We implemented a proof-of-concept prototype of our proposed optionally identifiable private handshake protocol, upon the widely used TLS handshake protocol [15]. Our implementation only requires small changes to the TLS handshake messages. Figure 3 shows how we incorporate our protocol into the messages of TLS (outside of brackets are the standard messages of TLS, and inside of brackets are messages of our protocol). More specifically, in TLS the client (u_1 in our protocol) initiates the protocol by sending a *ClientHello* message to the server (u_2 in our protocol). This message contains a random nonce, which is R_1 in our protocol. We also embed b into the *ClientHello* message. In response, the server returns a *ServerHello* message. The random nonce contained in the message corresponds to R_2 in our protocol. The server then sends a *ServerHelloDone* message, indicating that it completes handshake negotiation. The client and the server then send to each other *ClientKeyExchange* and *ServerKeyExchange*, respectively. These messages contain an *indication* on what algorithm is to be used. We modified these messages to indicate that the optionally identifiable private handshake scheme (OiPHS) is used. At this point of execution, the server and

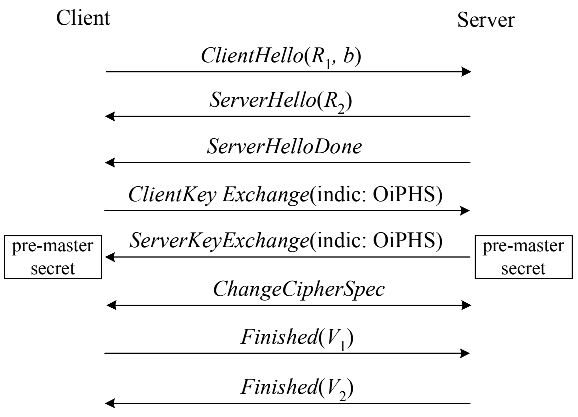


Fig. 3. Implementation of our protocol upon TLS

the client have enough information to compute $\hat{e}(H_0(u_1), H_0(u_2))^{s_G r_1 r_2}$. We set $\hat{e}(H_0(u_1), H_0(u_2))^{s_G r_1 r_2}$ to be the TLS *pre-master secret*, which will be used to derive all subsequent session keys for encryption or authentication.

The server and the client continue to exchange the *ChangeCipherSpec* message, indicating that they should begin to use the keys and algorithms they just negotiated from then on. Finally, they exchange the *Finished* message, containing V_1 and V_2, respectively, to allow each other to confirm that their counterpart has correctly computed the pre-master secret. This completes the TLS interactions.

Note that the above interactions implement private handshakes when $b = 0$. In the case of $b = 1$, we need 2 extra interactions to exchange C_1, C_2 in our protocol. To achieve this, our current implementation simply invokes exchanges of two *Application* messages (the content type is 23).

6.1 Experimental Results

Our implementation was written in C/C++, and used the MIRACL library [14]. The curve we chose is $y^2 = x^3 + x + 1$, and the bilinear map $\hat{e}$ is the Tate pairing. [6] further discussed the properties and performance improvements for this curve. In our protocol, H_0 maps random strings to points in the curve. We implemented H_0 by simply applying a pseudorandom number generator upon the strings to be mapped, and then generating pseudorandom points in the curve.

The security parameters associated with the curve are two primes p, q, where the typical size of p is 1024 bits, and q is 160 bits. Table 1 shows the experiment

Table 1. Experimental results

Size of q	Size of p	Timing
160 bits	1024 bits	2.6 sec
200 bits	2048 bits	13.1 sec

results with respect to varying sizes of p and q, by running our implementation on PCs with 2GHz Pentium CPU and 512M RAM. These results can be further improved, by taking the performance optimization mechanisms in [6]. Our current implementation did not consider any optimization measures, and we leave performance optimization to future work. The experimental results indicate that the performance of our proposed protocol should be satisfactory for practical applications.

7 Conclusion

Nowadays, with the prevalence of Internet applications users are becoming increasingly concerned about individual privacy. In this paper, we proposed a new privacy-preserving authentication primitive, *optionally identifiable private handshakes*. Interpolating between private handshakes and secret handshakes, optionally identifiable private handshakes allow the two users in a handshake to negotiate *real time* the level of privacy protection upon their interaction. As such, optionally identifiable private handshakes represent a more flexible privacy-preserving authentication technique than private handshakes and secret handshakes. We formulated a set of security requirements for optionally identifiable private handshakes. We then propose a concrete scheme, which is provably secure with respect to the formulated security definitions under the Bilinear Diffie-Hellman (BDH) assumption. We further implemented a proof-of-concept prototype of our proposed protocol upon the widely used TLS protocol, and the experimental results showed that our protocol has satisfactory performance.

References

1. Ateniese, G., Blanton, M., Kirsch, J.: Secret Handshakes with Dynamic and Fuzzy Matching. In: Proc. Network and Distributed System Security Symposium, NDSS 2007 (2007)
2. Ateniese, G., Medeiros, B.: Efficient Group Signatures without Trapdoors. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 246–268. Springer, Heidelberg (2003)
3. Balfanz, D., Durfee, G., Shankar, N., Smetters, D., Staddon, J., Wong, H.: Secret Handshakes from Pairing-Based Key Agreements. In: Proc. IEEE Security & Privacy, pp. 180–196 (2003)
4. Boneh, D., Franklin, M.: Identity-based Encryption from the Weil Paring. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
5. Baldwin, R., Gramlich, W.: Cryptographic Protocol for Trustable Matching Making. In: Proc. IEEE Security & Privacy, pp. 92–100 (1985)
6. Barreto, M., Kim, H., Lynn, B., Scott, M.: Efficient Algorithms for Pairing-based Cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, p. 354. Springer, Heidelberg (2002)
7. Bellar, M., Rogaway, P.: Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In: Proc. ACM Computers and Communications Security, CCS 2003, pp. 62–73 (2003)

8. Castelluccia, C., Jarecki, S., Tsudik, G.: Seccret Handshakes from Oblivious Encryption. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 293–307. Springer, Heidelberg (2004)
9. Camenisch, J., Lysyanskaya, A.: An Efficient Sysem for Non-Transferable Anonymous Credentials with Optional Anonymity Revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
10. Coron, J.S.: On the Exact Security of Full Domain Hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (2000)
11. Jarechi, S., Kim, J., Tsudik, G.: Authenticated Group Key Agreement Protocols with the Privacy Property of Affilation-hidding. In: Proc. CT-RSA Conference (2007)
12. Jarecki, S., Liu, X.: Unlinkable Secret Handshakes and Key-Private Group Key Management Schemes. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 270–287. Springer, Heidelberg (2007)
13. Hoepman, J.H.: Private Handshakes. In: Proc. 4th Eur. Symp. on Security and Privacy in Ad hoc and Sensor Networks, pp. 31–42 (2007)
14. Scott, M.: Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL), http://indigo.ie/~mscott/
15. The Transport Layer Security (TLS) Protocol Version 1.2. IETF Network Working Group, http://tools.ietf.org/html/rfc5246
16. Tsudik, G., Xu, S.: Flexible Framework for Secret Handshakes. Cryptology ePrint Archive, Report 2005/034
17. Vergnaud, D.: RSA-based Secret Handshakes. In: Proc. International Workshop on Coding and Cryptogaphy (2005)
18. Xu, S., Yung, M.: K-Anonymous Secret Handshakes with Reusable Credentials. In: Proc. ACM Computers and Communications Security, CCS 2004, pp. 158–167 (2004)
19. Yang, Y.J., Weng, J., Zhou, J.Y., Qiu, Y.: Optionally Identifiable Private Handshakes. In: Proc. ICICS 2009 (2009), http://icsd.i2r.a-star.edu.sg/sta/yanjiang/papers/icics09.pdf

Communication Efficient Statistical Asynchronous Multiparty Computation with Optimal Resilience

Arpita Patra*, Ashish Choudhury**, and C. Pandu Rangan***

Dept of Computer Science and Engineering
IIT Madras, Chennai India 600036
arpitapatra10@gmail.com, partho_31@yahoo.co.in, prangan55@gmail.com

Abstract. We propose an efficient statistically secure *asynchronous multiparty computation* (AMPC) protocol with *optimal fault tolerance*; i.e., with $n = 3t + 1$, where n is the total number of parties and t is the number of parties that can be under the influence of a *Byzantine (active)* adversary $\mathcal{A}_t$ having *unbounded computing power*. Our protocol privately communicates $\mathcal{O}(n^5\kappa)$ bits *per multiplication gate* and involves a negligible error probability of $2^{-\Omega(\kappa)}$, where κ is the error parameter. As far as our knowledge is concerned, the only known statistically secure AMPC protocol with $n = 3t + 1$ is due to [7], which privately communicates $\Omega(n^{11}\kappa^4)$ bits and A-casts $\Omega(n^{11}\kappa^2 \log(n))$ bits per multiplication gate. Here A-cast is an asynchronous broadcast primitive, which allows a party to send some information to all other parties identically. Thus our AMPC protocol shows significant improvement in communication complexity over the AMPC protocol of [7].

1 Introduction

A Multiparty Computation (MPC) [20,11,6,19] protocol is carried out among a set of n parties, say $\mathcal{P} = \{P_1, \ldots, P_n\}$, where every two parties are directly connected by a secure channel and t out of the n parties can be under the influence of a *computationally unbounded Byzantine (active) adversary*, denoted by $\mathcal{A}_t$. The adversary $\mathcal{A}_t$, completely dictates the parties under its control and can force them to deviate from a protocol, in any arbitrary manner. MPC allows the parties in $\mathcal{P}$ to securely compute an agreed function f, even in the presence of $\mathcal{A}_t$. More specifically, assume that the agreed function f can be expressed as $f : \mathbb{F}^n \rightarrow \mathbb{F}^n$ and party P_i has input $x_i \in \mathbb{F}$, where $\mathbb{F}$ is a finite field. At the end of the computation of f, each honest P_i gets $y_i \in \mathbb{F}$, where $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$, irrespective of the behavior of $\mathcal{A}_t$ (**correctness**). Moreover, $\mathcal{A}_t$ should not get any information about the input and output of the honest parties, other than what can be inferred from the input and output

* Financial Support from Microsoft Research India Acknowledged.

** Financial Support from Infosys Technology India Acknowledged.

*** Work Supported by Project No. CSE/05-06/076/DITX/CPAN on Protocols for Secure Communication and Computation Sponsored by Department of Information Technology, Government of India.

of the corrupted parties (**secrecy**). In any general MPC protocol, the function f is specified by an arithmetic circuit over $\mathbb{F}$, consisting of input, linear (e.g. addition), multiplication, random and output gates. We denote the number of gates of these types in the circuit by c_I, c_A, c_M, c_R and c_O respectively. *Among all the different type of gates, evaluation of a multiplication gate requires the most communication complexity. So the communication complexity of any general MPC protocol is usually given in terms of the communication complexity per multiplication gate* (see [4]).

The MPC problem has been studied extensively over synchronous networks. However, MPC in asynchronous network has got comparatively less attention, due to its inherent hardness. As asynchronous networks model real life networks like Internet more appropriately than synchronous networks, fundamental problems like MPC is worthy of deep investigation over asynchronous networks.

Asynchronous Networks: In an asynchronous network, the communication channels have arbitrary, yet finite delay (i.e the messages are guaranteed to reach eventually). To model this, $\mathcal{A}_t$ is given the power to schedule delivery of *all* messages in the network. However, $\mathcal{A}_t$ can *only schedule* the messages communicated between honest parties, without having any access to them. Here the inherent difficulty in designing a protocol comes from the fact that when a party does not receive an expected message then he cannot decide whether the sender is corrupted (and did not send the message at all) or the message is just delayed. So a party can not wait to consider the values sent by all parties, as waiting for them could turn out to be endless. Hence the values of up to t (potentially honest) parties may have to be ignored. Due to this the protocols in asynchronous network are generally involved in nature and require new set of primitives.

Asynchronous Multiparty Computation (AMPC): Any asynchronous MPC (AMPC) protocol should satisfy **termination** condition, in addition to **correctness** and **secrecy** condition (specified earlier). According to the termination condition, every honest party should eventually terminate the protocol. There are mainly two types of AMPC protocols: (i) A *perfectly secure* AMPC protocol satisfies all the properties of AMPC *without any error*; (ii) On the other hand, a *statistically secure* (statistical in short) AMPC protocol involves a *negligible error* probability of $2^{-\Omega(\kappa)}$ in **correctness** and/or **termination**, for an error parameter κ, where $\kappa = poly(n)$. From [5], *perfectly secure* AMPC is possible iff $n \geq 4t+1$. On the other hand, *statistically secure* AMPC is possible iff $n \geq 3t+1$ [7]. In this paper, we concentrate on *statistically secure* AMPC with *optimal resilience*; i.e., with $n = 3t+1$. As far our knowledge is concerned, the only known statistically secure AMPC protocol with $n = 3t+1$ is due to [7], which privately communicate $\Omega(c_M n^{11} \kappa^4)$ bits and A-cast $\Omega(c_M n^{11} \kappa^2 \log(n))$ bits.

Our Contribution: We design a statistically secure AMPC protocol with $n = 3t+1$ which privately communicates $\mathcal{O}(n^5\kappa)$ bits per multiplication gate. Thus our AMPC protocol significantly improves the communication complexity of only known optimally resilient statistically secure AMPC protocol of [7]. As a tool for our AMPC protocol, we present a new *Asynchronous Complete Secret*

Sharing (ACSS) scheme. For designing our ACSS, we present a new statistical *asynchronous verifiable secret sharing* (AVSS) protocol with $n = 3t + 1$. The novelty of our ACSS protocol is its specific design approach and the way we use the AVSS in it.

Remark 1. Recently in [16], the authors have designed an ACSS scheme which has the same communication complexity as the ACSS scheme of this paper. Moreover, using the ACSS scheme of [16], we can design a statistically secure AMPC protocol with $n = 3t + 1$ which privately communicates $\mathcal{O}(n^5\kappa)$ bits per multiplication gate. However, the ACSS scheme of this paper is completely different from the ACSS scheme of [16] and based on completely different techniques. We believe that the techniques used in our ACSS scheme can be further optimised, leading to efficiency gain in AMPC protocol.

1.1 Definitions

For the rest of the paper, we assume a finite field $\mathbb{F} = GF(2^\kappa)$, over which all the computation and communication are performed. Here κ is the error parameter. Thus each field element can be represented by $\mathcal{O}(\kappa)$ bits. Moreover, without loss of generality we assume that $\kappa = \text{poly}(n)$. We now present the definition of the primitives that are used for the design of our AMPC protocol.

Statistical Asynchronous Weak Secret Sharing (AWSS) [10,9]: Let (Sh, Rec) be a pair of protocols in which a *dealer* $D \in \mathcal{P}$ shares a secret s from $\mathbb{F} \cup \{NULL\}$ using Sh. We say that (Sh, Rec) is a t-resilient *statistically secure* AWSS scheme if the following hold:

- **Termination**: With probability at least $1 - 2^{-\Omega(\kappa)}$, the following holds: (1) If D is *honest* then each *honest* party will eventually terminate protocol Sh; (2) If some honest party has terminated protocol Sh, then irrespective of the behavior of D, each honest party will eventually terminate Sh; (3) If all the honest parties have terminated Sh and if all the honest parties invoke protocol Rec, then each honest party will eventually terminate Rec.
- **Correctness**: With probability at least $1 - 2^{-\Omega(\kappa)}$, the following holds: (1) If D is *honest* then each honest party upon terminating Rec outputs s; (2) If D is *corrupted* and some honest party has terminated Sh, then there exists a fixed $\overline{s} \in \mathbb{F} \cup \{NULL\}$, such that each honest party upon terminating Rec, will output either $\overline{s}$ or $NULL$.
- **Secrecy**: If D is honest and no honest party has begun Rec, then $\mathcal{A}_t$ has no information about s.

Statistical Asynchronous Verifiable Secret Sharing (AVSS) [10,9]: The definition of AVSS is same as AWSS except that **Correctness 2** property is *strengthened* as follows: If D is *corrupted* and some honest party has terminated Sh, then there exists a fixed $\overline{s} \in \mathbb{F} \cup \{NULL\}$, such that each honest party upon terminating Rec, will output *only* $\overline{s}$.

Statistical Asynchronous Complete Secret Sharing (ACSS): It is same as AVSS. In addition, ACSS should satisfy the following *completeness* property with probability at least $(1 - 2^{-\Omega(\kappa)})$: *If some honest party has terminated*

Sh, then each honest party will eventually hold proper share of secret. More importantly ACSS enforces D to share a secret s *only from* $\mathbb{F}$ (as opposed to $\mathbb{F} \cup \{NULL\}$ in AVSS).

Remark 2 (Difference Between AVSS and ACSS). An AVSS ensures the unique reconstruction of a shared secret, without ensuring *completeness* property i.e., each honest party may not hold proper share of the shared secret at the end of Sh. However, ACSS ensures the *completeness* property, apart from satisfying all the properties of AVSS. Moreover while AVSS can only ensure $s \in \mathbb{F} \cup \{NULL\}$, ACSS ensures that $s \in \mathbb{F}$. Thus ACSS has more stronger property than AVSS.

All the above definitions can be extended for secret S containing m elements from $\mathbb{F}$ with $m > 1$.

A-cast: It is an asynchronous broadcast primitive, which allows a special party in $\mathcal{P}$ (called the sender) to distribute a message identically among the parties in $\mathcal{P}$. If the sender is honest, then every honest party eventually terminates A-cast with the sender's message. For a corrupted sender, if some honest party terminates with some message, then every other honest party will eventually terminate with **same** message. A-cast is elegantly implemented in [8] with $n = 3t + 1$, which incurs a private communication of $\mathcal{O}(n^2 b)$ bits for a b-bit message.

Agreement on Common Subset (ACS)[3,7]: It is an asynchronous primitive presented in [5,7]. It outputs a common set, containing at least $n-t$ parties, who correctly shared their values. Moreover, each honest party will eventually get a share, corresponding to each value, shared by the parties in the common set. ACS requires private communication of $\mathcal{O}(\text{poly}(n, \kappa))$ bits.

2 Approach Used in AMPC of [7] and Current Article

AMPC of [7]: The AMPC protocol of [7] consists of input phase and computation phase. In input phase every party shares (or commits) his input x_i. All the parties then decide on a common set of $n - t$ parties (using ACS) who have done proper sharing of their input. Now for sharing/committing inputs, a natural choice is to use AVSS protocol which can be treated as a form of *commitment*, where the commitment is held in a distributed fashion among the parties. Before [7], the only known AVSS scheme with $n = 3t + 1$ was due to [10]. But it is shown in [7] that the use of the AVSS protocol of [10] for committing inputs (secrets), does not allow to compute the circuit robustly in a straight-forward way. This is because *for robust computation of the circuit, it is to be ensured that at the end of AVSS sharing phase, every honest party should have access to proper share of the secret.* Unfortunately the AVSS of [10] does not guarantee the above property, which we may refer as *ultimate* property. This very reason motivated Ben-Or et. al [7] to introduce a new asynchronous primitive called *Ultimate Secret Sharing* (USS) which not only ensures that every honest party has access to the proper share of secret, but also offers all the properties of AVSS. Thus [7] presents an USS scheme with $n = 3t + 1$ using the AVSS protocol of [10] as a building block. A secret s that is shared using USS is called *ultimately* *shared*. Now in the input phase of AMPC in [7], parties *ultimately share* their inputs. Then in computation phase, for every gate (except output gate), *ultimate sharing* of the output is computed from *ultimate sharing* of the inputs, following approach of [6,19].

AMPC of Current Article: Our AMPC protocol is presented in preprocessing model of [1] and proceeds in three phases: preparation phase, input phase and computation phase. We call a triple (a, b, c) as a random multiplication triple if a, b are random and $c = ab$. In the preparation phase, sharing of $c_M + c_R$ random multiplication triples are generated. Each multiplication and random gate of the circuit is associated with a multiplication triple. In the input phase the parties share (commit) their inputs and agree on a common subset of $n-t$ parties (using ACS) who correctly shared their inputs. In the computation phase, the actual circuit will be computed gate by gate, based on the inputs of the parties in common set. Due to the linearity of the used secret-sharing, the linear gates can be computed locally. Each multiplication gate will be evaluated using the circuit randomization technique of [1] with the help of the associated multiplication triple (generated in preparation phase).

For committing/sharing secrets, we use a new asynchronous primitive called ACSS. There is a slight *definitional difference* between the USS of [7] and our ACSS, though both of them offer all the properties of AVSS. While USS of [7] ensures that every honest party has *access* to proper share of secret (*but may not hold the share directly*), our ACSS ensures that *every honest party holds proper share* of secret. This property of ACSS is called *completeness* property as mentioned in the definition of ACSS. The advantages of ACSS over USS are: (a) it makes the computation of the gates very simple, (b) reconstruction phase of ACSS is very simple, efficient and can be achieved using *on-line error correction* of [9]. Apart from these advantages, our ACSS is strikingly better than USS of [7] in terms of communication complexity. While Sh protocol of our ACSS privately communicates $\mathcal{O}((mn^4 + n^7\kappa)\kappa))$ bits and A-casts $\mathcal{O}(n^4 \log(n))$ bits to share m *secrets concurrently*, the Sh of USS in [7] privately communicates $\Omega(n^{10}\kappa^4)$ bits and A-casts $\Omega(n^{10}\kappa^2 \log(n))$ bits to share *only one secret*.

3 Asynchronous Complete Secret Sharing (ACSS)

Here we present an ACSS scheme that allows a dealer $D \in \mathcal{P}$ to share $m \geq 1$ secrets concurrently. For that, we first recall existing protocols from [16] for two asynchronous primitives, namely Information Checking Protocol (ICP) and AWSS. The AWSS uses ICP as a building block. We then design a new AVSS scheme from the AWSS scheme of [16], using the approach of [17]. Notice that [17] also presents an ICP and AWSS scheme. However, the A-cast communication of ICP and AWSS of [17] is too high in comparison to ICP and AWSS of [16].

We then *extend* the ICP in a special way to design an *extended AWSS* which in turn is used to design an *extended AVSS* protocol. Finally ACSS is designed using the extended AVSS as a building block. Here our main contribution is the specific design approach of the ACSS protocol using the extended AVSS and the specific ideas of extending AVSS, AWSS and ICP. The extension of ICP, AWSS and AVSS are done in order to maintain a low communication complexity of our ACSS protocol (it will be clear while we present the ACSS protocol). For the sake of simplicity, we will first present ICP, AWSS and AVSS and then describe their extended form.

3.1 Information Checking Protocol and IC Signature

The Information Checking Protocol (ICP) is a tool for authenticating messages in the presence of $\mathcal{A}_t$. The notion of ICP was first introduced by Rabin et.al [19,18]. In [16], the authors have designed an ICP in asynchronous settings, called as A-ICP. We now briefly outline the A-ICP of [16], which is further used in AWSS protocol. The A-ICP is executed among three entities: the dealer $D \in \mathcal{P}$, an intermediary $INT \in \mathcal{P}$ and entire set $\mathcal{P}$ (including D, INT) acting as verifiers. The dealer D hands a secret S containing $\ell \geq 1$ elements from $\mathbb{F}$ to INT. At a later stage, INT is required to hand over S to the verifiers in $\mathcal{P}$ and convince them that S is indeed the secret which INT received from D. A-ICP is structured into sequence of following three phases:

1. Generation Phase: is initiated D. Here D hands over the secret S, along with *authentication information* to *intermediary* INT and some *verification information* to individual *verifiers* in $\mathcal{P}$.

2. Verification Phase: is carried out by INT and the set of verifiers $\mathcal{P}$. Here INT decides whether to continue or abort the protocol depending upon the prediction whether in **Revelation Phase**, the secret S held by INT will be (eventually) accepted by honest verifier(s) in $\mathcal{P}$. INT achieves this by setting a boolean variable $\mathsf{Ver} = 0/1$, where $\mathsf{Ver} = 0$ (resp. 1) implies abortion (resp. continuation) of the protocol. If $\mathsf{Ver} = 1$, then the *authentication information*, along with S, held by INT is called D's IC *signature* on S, denoted as $ICSig(D, INT, \mathcal{P}, S)$.

3. Revelation Phase: is carried out by INT and the verifiers in $\mathcal{P}$. **Revelation Phase** can be presented in two flavors: (i) *Public Revelation* of $ICSig(D, INT, \mathcal{P}, S)$, where INT *publicly* reveals $ICSig(D, INT, \mathcal{P}, S)$ to all the verifiers in $\mathcal{P}$. If S is properly revealed, then every honest P_i sets $\mathsf{Reveal}_i = S$, otherwise he sets $\mathsf{Reveal}_i = NULL$; (ii) P_α*-private-revelation* of $ICSig(D, INT, \mathcal{P}, S)$: Here INT *privately* reveals $ICSig(D, INT, \mathcal{P}, S)$ to *only* a specific verifier, say $P_\alpha \in \mathcal{P}$. If the revelation is successful, then P_α sets $\mathsf{Reveal}_\alpha = S$, otherwise he sets $\mathsf{Reveal}_\alpha = NULL$.

Protocol A-ICP satisfies the following properties (assuming *Public Revelation* in **Revelation Phase**): (1) If D and INT are honest, then S will be accepted in **Revelation phase** by each honest verifier. (2) If INT is honest and Ver =1, then S held by INT will be accepted in **Revelation phase** by each honest verifier, except with probability $2^{-\Omega(\kappa)}$. (3) If D is honest, then during **Revelation phase**, with probability at least $1 - 2^{-\Omega(\kappa)}$, every $S' \neq S$ produced by a corrupted INT will be not be accepted by an honest verifier. (4) If D and INT are honest and INT has not started **Revelation phase**, then S is information theoretically secure.

For A-ICP with P_α*-private-revelation* in **Revelation Phase**, the above properties are modified by replacing every "every honest verifier" with "honest P_α". In [16], A-ICP is presented with only P_α-private-revelation of $ICSig(D, INT, \mathcal{P}, S)$. Here we present the code for the public revelation of $ICSig(D, INT, \mathcal{P}, S)$ and recall protocol A-ICP. For proof of the properties of A-ICP, see [16]. For ease of reference, we provide the communication complexity of A-ICP here.

Protocol A-ICP$(D, INT, \mathcal{P}, S)$

Generation Phase: Gen$(D, INT, \mathcal{P}, S)$

1. D selects a random $\ell + t\kappa$ degree polynomial $f(x)$ whose lower order ℓ coefficients are the secrets in $S = (s^1, \ldots, s^\ell)$. D also picks $n\kappa$ random non-zero elements from $\mathbb{F}$, denoted by $\alpha_1^i, \ldots, \alpha_\kappa^i$, for $1 \le i \le n$.
2. D sends $f(x)$ to INT and the verification tags $z_1^i = (\alpha_1^i, a_1^i), \ldots, z_\kappa^i = (\alpha_\kappa^i, a_\kappa^i)$ to P_i, where $a_j^i = f(\alpha_j^i)$.

Verification Phase: Ver$(D, INT, \mathcal{P}, S)$

1. Every verifier P_i randomly partitions the index set $\{1, \ldots, \kappa\}$ into two sets I^i and $\overline{I^i}$ of equal size and sends I^i and z_j^i for all $j \in I^i$ to INT.
2. For every verifier P_i from which INT has received values, INT checks whether for *every* $j \in I^i$, $f(\alpha_j^i) \overset{?}{=} a_j^i$.
3. (a) If for at least $2t + 1$ verifiers, the above condition is satisfied, then INT sets $\mathsf{Ver} = 1$. If $\mathsf{Ver} = 1$, then INT assumes $f(x)$ to be $ICSig(D, INT, \mathcal{P}, S)$.
   (b) If for at least $t + 1$ verifiers, the above condition is not satisfied, then INT sets $\mathsf{Ver} = 0$.

Revelation Phase:

– Reveal-Private$(D, INT, \mathcal{P}, S, P_\alpha)$: P_α*-private-revelation* of $ICSig(D, INT, \mathcal{P}, S)$
  1. To party P_α, INT sends $f(x)$.
  2. To party P_α, every verifier P_i sends the index set $\overline{I^i}$ and all z_j^i such that $j \in \overline{I^i}$.
  3. Upon receiving the values from a verifier P_i, party P_α checks whether for *any* $j \in \overline{I^i}$, $f(\alpha_j^i) \overset{?}{=} a_j^i$.
     (a) If for at least $t + 1$ verifiers the above condition is satisfied, then P_α sets $\mathsf{Reveal}_\alpha = S$, where S is lower order ℓ coefficients of $f(x)$.
     (b) If for at least $2t + 1$ verifiers the above condition is not satisfied, then P_α sets $\mathsf{Reveal}_\alpha = NULL$.

– Reveal-Public$(D, INT, \mathcal{P}, S)$: *Public Revelation* of $ICSig(D, INT, \mathcal{P}, S)$
  1. INT A-casts $f(x)$.
  2. Every verifier P_i checks whether for *any* $j \in \overline{I^i}$, $f(\alpha_j^i) \overset{?}{=} a_j^i$. If yes, then he A-casts `Yes`, otherwise he A-casts `No`.
  3. If `Yes` is received from the A-cast of at least $t + 1$ verifiers, then every verifier P_i sets $\mathsf{Reveal}_i = S$, where S is lower order ℓ coefficients of $f(x)$.
  4. If `No` is received from the A-cast of at least $2t + 1$ verifiers, then every verifier P_i sets $\mathsf{Reveal}_i = NULL$.

Lemma 1. *Protocol Gen, Ver and Reveal-Private privately communicate $\mathcal{O}((\ell + n\kappa)\kappa)$ bits each. Protocol Reveal-Public A-casts $\mathcal{O}((\ell + n\kappa)\kappa)$ bits. Simulating* *the* A-cast *by protocol of* [8], Reveal-Public *requires private communication of* $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ *bits.*

Notation 1. *In the AWSS protocol, we will use following notations: Recall that D and INT can be any party from $\mathcal{P}$. We say that: (i) "P_i sends $ICSig(P_i, P_j, \mathcal{P}, S)$ to P_j" to mean that P_i as D, executes* Gen$(P_i, P_j, \mathcal{P}, S)$*; (ii) "$P_i$ receives $ICSig(P_j, P_i, \mathcal{P}, S)$ from P_j" to mean that P_i as INT has completed* Ver$(P_j, P_i, \mathcal{P}, S)$ *with* Ver $= 1$ *with the help of the verifiers in $\mathcal{P}$; (iii) "P_i reveals $ICSig(P_j, P_i, \mathcal{P}, S)$ to P_α" to means P_i as INT executes* Reveal-Private$(P_j, P_i, \mathcal{P}, S, P_\alpha)$ *along with the participation of the verifiers in $\mathcal{P}$ (similarly "P_i reveals $ICSig(P_j, P_i, \mathcal{P}, S)$" is to be interpreted with respect to* Reveal-Public$(P_j, P_i, \mathcal{P}, S)$*); (iv) "$P_\alpha$ completes revelation of $ICSig(P_j, P_i, \mathcal{P}, S)$ with* Reveal$_\alpha = \overline{S}$ *" to mean that P_α has completed* Reveal-Private$(P_j, P_i, \mathcal{P}, S, P_\alpha)$/ Reveal-Public$(P_j, P_i, \mathcal{P}, S)$ *with* Reveal$_\alpha = \overline{S}$.

3.2 Extended ICP

In our extended AWSS protocol, we come across the following situation: D holds $\mathcal{B} = n^2$ blocks, denoted by $S^1, \ldots, S^{\mathcal{B}}$, each containing $\eta = \frac{m}{n^2}$ secrets. In addition D has another random secret block, say S^0 containing η elements. Let $S^b = (s^{1b}, \ldots, s^{\eta b})$ for $b = 0, \ldots, \mathcal{B}$. Now D wants to send $ICSig(D, INT, \mathcal{P}, S^b)$ for every $b \in \{0, \ldots, \mathcal{B}\}$ to INT such that INT can later compute and reveal $ICSig(D, INT, \mathcal{P}, S^*)$ where $S^* = (s^{1*}, \ldots, s^{\eta *})$ with $s^{k*} = \sum_{b=0}^{\mathcal{B}} r^b s^{kb}$ ($k = 1, \ldots, \eta$) for some *agreed upon random value* r, generated by all the parties in $\mathcal{P}$. Note that the random value r will be generated *after* INT ensures the receipt of $ICSig(D, INT, \mathcal{P}, S^b)$ for every $b \in \{0, \ldots, \mathcal{B}\}$. We will use the following notation: $S^* = \sum_{b=0}^{\mathcal{B}} r^b S^b$. To achieve the above task, we first extend the protocols for **Generation Phase** and **Verification Phase** of A-ICP protocol. Subsequently, we present the code for generating $ICSig(D, INT, \mathcal{P}, S^*)$ and revealing the same. The protocols are given on next page.

Theorem 1. *If D is honest then during* Reveal-IC-Sig*, S^* will not reveal any information about $S^1, \ldots, S^{\mathcal{B}}$.*

Theorem 2. *Protocol* Extd-Gen *and* Extd-Ver *privately communicate $\mathcal{O}((m + n^3\kappa)\kappa)$ bits each. Protocol* Compute-IC-Sig *does not require any communication. Protocol* Reveal-IC-Sig *requires $\mathcal{O}((m + n^3\kappa)\kappa)$ bits of private communication (follows from the communication complexity of* Reveal-Public*).*

Notation 2. *In our extended AWSS, we use following notations. We say that: (i) "P_i sends $ICSig(P_i, P_j, \mathcal{P}, (S^0, \ldots, S^{\mathcal{B}}))$ to P_j" to mean that P_i as a dealer executes* Extd-Gen*$(P_i, P_j, \mathcal{P}, (S^0, S^1, \ldots, S^{\mathcal{B}}))$; (ii) "$P_i$ receives $ICSig(P_j, P_i, \mathcal{P}, (S^0, \ldots, S^{\mathcal{B}}))$ from P_j" to mean that P_i as INT has successfully completed* Ext-Ver *$(P_j, P_i, \mathcal{P}, (S^0, S^1, \ldots, S^{\mathcal{B}}))$ with the help of the verifiers in $\mathcal{P}$; (c) "P_i participates in computing $ICSig(P_a, P_b, \mathcal{P}, S^*)$ from $ICSig(P_a, P_b, \mathcal{P}, (S^0, \ldots, S^{\mathcal{B}}))$" to mean that P_i executes* Compute-IC-Sig*$(P_a, P_b, \mathcal{P}, r, (S^0, S^1, \ldots, S^{\mathcal{B}}))$ as a verifier and also as INT if $P_b = P_i$.*

Remark 3. Note that during Extd-Ver, once (honest) INT receives $ICSig(D, INT, \mathcal{P}, (S^0, \ldots, S^{\mathcal{B}}))$, later he can privately reveal $ICSig(D, INT, \mathcal{P}, S^b)$ for every b to any P_α, using Reveal-Private.

Protocol Extd-A-ICP$(D, INT, \mathcal{P}, (S^0, S^1, \ldots, S^{\mathcal{B}}))$

Generation Phase: Extd-Gen$(D, INT, \mathcal{P}, (S^0, S^1, \ldots, S^{\mathcal{B}}))$

1. D sends $ICSig(D, INT, \mathcal{P}, S^b)$ to INT for every $b \in \{0, \ldots, \mathcal{B}\}$.
2. Let during the execution of Gen$(D, INT, \mathcal{P}, S^b)$, D sends $f^b(x)$ of degree $\eta + t\kappa$ to INT (recall that each S^b is of size η) and the verification tags $z_1^{ib} = (\alpha_1^i, a_1^{ib}), \ldots, z_\kappa^{ib} = (\alpha_\kappa^i, a_\kappa^{ib})$ to P_i, where $a_k^{ib} = f^b(\alpha_k^i)$ for $k = 1, \ldots, \kappa$. **Notice that corresponding to P_i, the same set of α values are selected for each execution of Gen$(D, INT, \mathcal{P}, S^b)$.**

Verification Phase: Extd-Ver$(D, INT, \mathcal{P}, (S^0, S^1, \ldots, S^{\mathcal{B}}))$

1. While executing Ver$(D, INT, \mathcal{P}, S^b)$ for every $b \in \{0, \ldots, \mathcal{B}\}$, INT waits for a *common set* of at least $2t + 1$ verifiers, say COM, for which the condition specified in step 2 of **Verification Phase** (see Protocol Ver) is satisfied for every Ver$(D, INT, \mathcal{P}, S^b)$. After that INT sets Ver $= 1$ in Ver$(D, INT, \mathcal{P}, S^b)$ for every $b \in \{0, \ldots, \mathcal{B}\}$. To be concise, we say that INT now has received $ICSig(D, INT, \mathcal{P}, (S^0, \ldots, S^{\mathcal{B}}))$ from D such that later INT can compute $ICSig(D, INT, \mathcal{P}, S^*)$ and can reveal the same using following protocols.

Protocol Compute-IC-Sig$(D, INT, \mathcal{P}, r, (S^0, S^1, \ldots, S^{\mathcal{B}}))$

1. INT computes $f^*(x) = \sum_{b=0}^{\mathcal{B}} r^b f^b(x)$ and assumes $f^*(x)$ to be $ICSig(D, INT, \mathcal{P}, S^*)$ where $S^* = \sum_{b=0}^{\mathcal{B}} r^b S^b$. Clearly, the η lower order coefficients of $f^*(x)$ are elements from S^*.
2. Every verifier P_i (note that INT and D are also included in the set of verifiers $\mathcal{P}$) computes corresponding verification tags $z_1^{i*} = (\alpha_1^i, a_1^{i*}), \ldots, z_\kappa^{i*} = (\alpha_\kappa^i, a_\kappa^{i*})$, where $a_k^{i*} = \sum_{b=0}^{\mathcal{B}} r^b a_k^{ib}$ for $k = 1, \ldots, \kappa$.

Protocol Reveal-IC-Sig$(D, INT, \mathcal{P}, S^*)$

1. Now INT and the verifiers in $\mathcal{P}$ can execute Reveal-Public$(D, INT, \mathcal{P}, S^*)$ for *public revelation* of $ICSig(D, INT, \mathcal{P}, S^*)$ with the information that they compute in protocol Compute-IC-Sig.

3.3 Statistical Asynchronous Weak Secret Sharing

We now recall AWSS scheme of [16] with $n = 3t + 1$, consisting of protocols AWSS-Share and AWSS-Rec-Private. Protocol AWSS-Share allows D to commit a secret $S = (s^1, \ldots, s^\ell)$ containing ℓ elements from $\mathbb{F}$. Moreover, if D is corrupted then he may commit $NULL$, instead of elements from $\mathbb{F}$ (the meaning of it will be clear in the sequel). In fact, protocol AWSS-Share is similar to the sharing protocol of AWSS scheme of [17]. Protocol AWSS-Rec-Private enables private reconstruction of S or $NULL$ to any $P_\alpha \in \mathcal{P}$. The authors of [16] called the private reconstruction as *P_α-weak-private-reconstruction*. For our ACSS, we require public reconstruction of the secret as well. So we also present a protocol AWSS-Rec-Public (which was not presented in [16]).

AWSS-Share($D, \mathcal{P}, S$)

DISTRIBUTION: CODE FOR D – Only D executes this code.

1. For $l = 1, \ldots, \ell$, select a random, symmetric bivariate polynomial $F^l(x, y)$ of degree-t in x and y such that $F^l(0, 0) = s^l$. Let $f_i^l(x) = F^l(x, i)$, for $l = 1, \ldots, \ell$.
2. For $i = 1, \ldots, n$, send $ICSig(D, P_i, \mathcal{P}, (f_i^1(j), \ldots, f_i^\ell(j))$ for each $j = 1, \ldots, n$ to P_i.

VERIFICATION: CODE FOR P_i – Every party including D executes this code.

1. Wait to receive $ICSig(D, P_i, \mathcal{P}, (f_i^1(j), \ldots, f_i^\ell(j))$ for all $j = 1, \ldots, n$ from D.
2. Check if $(f_i^l(1), \ldots, f_i^l(n))$ defines degree-t polynomial for every $l = 1, \ldots, \ell$. If yes then send $ICSig(P_i, P_j, \mathcal{P}, (f_i^1(j), \ldots, f_i^\ell(j)))$ to P_j for all $j = 1, \ldots, n$.
3. If $ICSig(P_j, P_i, \mathcal{P}, (f_j^1(i), \ldots, f_j^\ell(i)))$ is received from P_j and if $f_j^l(i) = f_i^l(j)$ for $l = 1, \ldots, \ell$, then A-cast $\texttt{OK}(P_i, P_j)$.

WCORE CONSTRUCTION : CODE FOR D – Only D executes this code.

1. For each P_j, build a set $OKP_j = \{P_i | D$ receives $\texttt{OK}(P_i, P_j)$ from the A-cast of $P_i\}$. When $|OKP_j| = 2t + 1$, then add P_j in $WCORE$ (which is initially empty). In this case, we say that P_j is *IC-committed* to $(f_j^1(0), \ldots, f_j^\ell(0))$ by sending $ICSig(P_j, P_i, \mathcal{P}, (f_j^1(i), \ldots, f_j^\ell(i)))$ to every P_i in OKP_j.
2. Wait until $|WCORE| = 2t + 1$. Then A-cast $WCORE$ and OKP_j for all $P_j \in WCORE$.

WCORE VERIFICATION & AGREEMENT ON WCORE : CODE FOR P_i

1. Wait to receive $WCORE$ and OKP_j for all $P_j \in WCORE$ from D's A-cast, such that $|WCORE| = 2t + 1$ and $|OKP_j| = 2t + 1$ for each $P_j \in WCORE$.
2. Wait to receive $\texttt{OK}(P_k, P_j)$ for all $P_k \in OKP_j$ and $P_j \in WCORE$. After receiving, accept the $WCORE$ and OKP_j's and terminate **AWSS-Share**.

Remark 4 (D's AWSS-commitment). We say that D is AWSS-committed to $S = (s^1, \ldots, s^\ell) \in \mathbb{F}^\ell$ if for every $l = 1, \ldots, \ell$ there is a unique degree-t polynomial $f^l(x)$ such that $f^l(0) = s^l$ and every *honest* P_i in $WCORE$ receives $f^l(i)$ from D and *IC-commits* $f^l(i)$ among the parties in OKP_i. Otherwise, we say that D is AWSS-committed to $NULL$. An honest D always AWSS-commits $S \in \mathbb{F}^\ell$ as in this case $f^l(x) = f_0^l(x) = F^l(x, 0)$, where $F^l(x, y)$ is the symmetric degree-(t, t) bivariate polynomial chosen by D. But AWSS-Share can *not* ensure that corrupted D also AWSS-commits $S \in \mathbb{F}^\ell$.

Notation 3. *In subsequent sections, we will invoke AWSS-Share as AWSS-Share $(D, \mathcal{P}, (f^1(x), \ldots, f^\ell(x)))$ where D is asked to choose bivariate polynomials $F^1(x, y), \ldots, F^\ell(x, y)$, each of degree-$t$ in x and y, such that $F^l(x, 0) = f^l(x)$ holds for $l = 1, \ldots, \ell$. Similarly, AWSS-Rec-Private will be invoked as AWSS-Rec-Private($D, \mathcal{P}, (f^1(x), \ldots, f^\ell(x)), P_\alpha$) to enable P_α-weak-private-reconstruction of $(f^1(x), \ldots, f^\ell(x))$. Furthermore AWSS-Rec-Public will be invoked as AWSS-Rec-Public($D, \mathcal{P}, (f^1(x), \ldots, f^\ell(x))$) to enable weak-public-reconstruction of $(f^1(x), \ldots, f^\ell(x))$ and hence $s^l = f^l(0)$ for $l = 1, \ldots, \ell$.*

Theorem 3. *AWSS-Share privately communicates $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ and A-casts $\mathcal{O}(n^2 \log n)$ bits. AWSS-Rec-Private and AWSS-Rec-Public privately communicates $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ and $\mathcal{O}((\ell n^4 + n^5\kappa)\kappa)$ bits respectively.*

```
AWSS-Rec-Private($D, \mathcal{P}, S, P_\alpha$): $P_\alpha$-weak-private-reconstruction of $S$:

SIGNATURE REVELATION: CODE FOR $P_i$
    1. If $P_i$ belongs to $OKP_j$ for some $P_j \in WCORE$, then reveal
       $ICSig(D, P_i, \mathcal{P}, (f_i^1(j), \ldots, f_i^\ell(j)))$, $ICSig(P_j, P_i, \mathcal{P}, (f_j^1(i), \ldots, f_j^\ell(i)))$ to $P_\alpha$.
LOCAL COMPUTATION: CODE FOR $P_\alpha$
    1. For every $P_j \in WCORE$, reconstruct $P_j$'s IC-Commitment, say
       $(\overline{f_j^1}(0), \ldots, \overline{f_j^\ell}(0))$ as follows:
       (a) Construct a set $ValidP_j = \emptyset$.
       (b) Add $P_k \in OKP_j$ to $ValidP_j$ if the following conditions hold:
           i. Revelation of $ICSig(D, P_k, \mathcal{P}, (f_k^1(j), \ldots, f_k^\ell(j)))$ and
              $ICSig(P_j, P_k, \mathcal{P}, (f_j^1(k), \ldots, f_j^\ell(k)))$ are completed with Reveal$_\alpha$ $= (\overline{f_k^1}(j), \ldots, \overline{f_k^\ell}(j))$ and Reveal$_\alpha$ $= (\overline{f_j^1}(k), \ldots, \overline{f_j^\ell}(k))$;
              and
           ii. $\overline{f_k^l}(j) = \overline{f_j^l}(k)$ for all $l = 1, \ldots, \ell$.
       (c) Wait until $|ValidP_j| = t+1$. For $l = 1, \ldots, \ell$, construct polynomial $\overline{f_j^l(x)}$
           passing through the points $(k, \overline{f_j^l(k)})$ where $P_k \in ValidP_j$ and associate
           $\overline{f_j^l(0)}$ with $P_j \in WCORE$.
    2. Wait until $\overline{f_j^l(0)}$ for $l = 1, \ldots, \ell$ is reconstructed for every $P_j$ in $WCORE$.
    3. Check whether the points $(j, \overline{f_j^l(0)})$ for $P_j \in WCORE$ lie on a unique degree-$t$
       polynomial $\overline{f_0^l(x)}$. If yes, then set $\overline{s^l} = \overline{f_0^l(0)}$. Else set $\overline{s^l} = NULL$. If some $s^l$
       is $NULL$, then set $\overline{S} = NULL$. Otherwise, set $\overline{S} = (\overline{s^1}, \ldots, \overline{s^\ell})$ and terminate
       AWSS-Rec-Private.

AWSS-Rec-Public($D, \mathcal{P}, S$): Weak-public-reconstruction of $S$: Same as the steps of
AWSS-Rec-Private except with the following modifications: In the code specified in
[SIGNATURE REVELATION:], public revelation of $ICSig$s are performed. Then every
party $P_i$ acting as $P_\alpha$ executes the code specified under [LOCAL COMPUTATION:CODE
FOR $P_\alpha$] to reconstruct either $\overline{S} \in \mathbb{F}^\ell$ or $NULL$.
```

3.4 Extended AWSS Protocol

In our extended AVSS protocol, we come across the following situation: D has $\mathcal{B} = n^2$ blocks, denoted by $S^1, \ldots, S^{\mathcal{B}}$, each containing $\eta = m/n^2$ secrets. In addition D has another random secret block, say S^0, containing η elements. Let $S^b = (s^{1b}, \ldots, s^{\eta b})$ for $b \in \{0, \ldots, \mathcal{B}\}$. Now D wants to AWSS-commit S^b using degree-t polynomials say, $\mathcal{F}^b = (f^{1b}(x), \ldots, f^{\eta b}(x))$, where$f^{kb}(0) = s^{kb}$ for $k = 1, \ldots, \eta$, such that later the parties can generate (by local computation) AWSS-commitment of $S^* = \sum_{b=0}^{\mathcal{B}} r^b.S^b = (s^{1*}, \ldots, s^{\eta *})$ using polynomials $\mathcal{F}^* = (f^{1*}(x), \ldots, f^{\eta *}(x))$, where $f^{k*}(x) = \sum_{b=0}^{\mathcal{B}} r^b f^{kb}(x)$ and $f^{k*}(0) = s^{k*}$, for $k = 1, \ldots, \eta$. Here r is some agreed upon random value generated by all the parties in $\mathcal{P}$. Later the parties can also reconstruct S^* by invoking AWSS-Rec-Public $(D, \mathcal{P}, (f^{1*}(x), \ldots, f^{\eta *}(x)))$. To achieve the above task, we first extend AWSS-Share to design Extd-AWSS-Share (given below) that uses Extd-A-ICP as building block. We then give protocols for generating AWSS-commitment of S^* (protocol Compute-AWSS-Commit) and revealing the same (protocol Rec-AWSS-Commit).

Note that in Compute-AWSS-Commit, the random value r will be generated after every honest party ensures that D is AWSS-committed to every S^b.

Protocol Extd-AWSS-Share($D, \mathcal{P}, (\mathcal{F}^0, \ldots, \mathcal{F}^{\mathcal{B}})$)

DISTRIBUTION: CODE FOR D – Only D executes this code.

1. For every $\mathcal{F}^b$, select η random, symmetric bivariate polynomials $F^{1b}(x,y), \ldots, F^{\eta b}(x,y)$ of degree-t in x and y, such that $F^{kb}(x,0) = f^{kb}(x)$ for $k = 1, \ldots, \eta$. Let for $i = 1, \ldots, n$, $f_i^{kb}(x) = F^{kb}(x,i)$.
2. Let $\Delta_i^b(j) = (f_i^{1b}(j), \ldots, f_i^{\eta b}(j))$ for $b = 0, \ldots, \mathcal{B}$ and $i, j = 1, \ldots, n$. Send $ICSig(D, P_i, \mathcal{P}, (\Delta_i^0(j), \ldots, \Delta_i^{\mathcal{B}}(j)))$ for each $j = 1, \ldots, n$ to P_i.

VERIFICATION: CODE FOR P_i – Every party including D executes this code.

1. Wait to receive $ICSig(D, P_i, \mathcal{P}, (\Delta_i^0(j), \ldots, \Delta_i^{\mathcal{B}}(j)))$ for $j = 1, \ldots, n$ from D.
2. Check if $(f_i^{kb}(1), \ldots, f_i^{kb}(n))$ defines degree-t polynomial for every $k = 1, \ldots, \eta$ and $b = 0, \ldots, \mathcal{B}$. If yes then send $ICSig(P_i, P_j, \mathcal{P}, (\Delta_i^0(j), \ldots, \Delta_i^{\mathcal{B}}(j)))$ to P_j for all $j = 1, \ldots, n$.
3. If $ICSig(P_j, P_i, \mathcal{P}, (\Delta_j^0(i), \ldots, \Delta_j^{\mathcal{B}}(i)))$ is received from P_j and if $f_i^{kb}(j) = f_j^{kb}(i)$, then A-cast `OK`$(P_i, P_j)$.

WCORE CONSTRUCTION and [WCORE VERIFICATION & AGREEMENT ON WCORE] are same as in AWSS-Share.

Protocol Compute-AWSS-Commit($D, \mathcal{P}, r, \mathcal{F}^*$)

CODE FOR P_i

1. Participate in computing $ICSig(D, P_j, \mathcal{P}, \Delta_j^*(k))$ for all $k = 1, \ldots, n$ for every $P_j \in WCORE$ where $\Delta_j^*(k) = \sum_{b=0}^{\mathcal{B}} r^b \Delta_j^b(k)$.
2. Participate in computing $ICSig(P_j, P_k, \mathcal{P}, \Delta_j^*(k))$ where $\Delta_j^*(k) = \sum_{b=0}^{\mathcal{B}} r^b \Delta_j^b(k)$ for every $P_k \in OKP_j$ and every $P_j \in WCORE$.

Protocol Rec-AWSS-Commit($D, \mathcal{P}, \mathcal{F}^*$)

1. Parties execute AWSS-Rec-Public($D, \mathcal{P}, \mathcal{F}^* = (f^{1*}(x), \ldots, f^{\eta *}(x))$ to enable weak-public-reconstruction of $\mathcal{F}^* = (f^{1*}(x), \ldots, f^{\eta *}(x))$ and hence $S^* = (f^{1*}(0), \ldots, f^{\eta *}(0))$.

Remark 5. Notice that once $WCORE$ is constructed and agreed upon in Extd-AWSS-Share, then the parties can later enable P_α-weak-private-reconstruction of any $\mathcal{F}^b$, where $b \in \{1, \ldots, \mathcal{B}\}$ and $P_\alpha \in \mathcal{P}$.

Theorem 4. *(i) If D is honest then revealing S^* during Rec-AWSS-Commit will not reveal any information about $S^1, \ldots, S^{\mathcal{B}}$; (ii) If D is corrupted and S^* revealed during Rec-AWSS-Commit is non-NULL (i.e., $S^* \in \mathbb{F}^\eta$), then with very high probability D has AWSS-Committed non-NULL $S^0, \ldots, S^{\mathcal{B}}$.*

Theorem 5. *Protocol Extd-AWSS-Share privately communicates $\mathcal{O}((mn^2+n^5\kappa)\kappa)$ bits and A-casts $\mathcal{O}(n^2 \log n)$ bits. Compute-AWSS-Commit does not require any communication. Rec-AWSS-Commit privately communicates $\mathcal{O}((mn^2 + n^5\kappa)\kappa)$ bits.*

3.5 Statistical Asynchronous Verifiable Secret Sharing

We now present an AVSS scheme with $n = 3t + 1$, consisting of protocols AVSS-Share, AVSS-Rec-Private and AVSS-Rec-Public. These protocols are similar to [17]. Protocol AVSS-Share allows D to commit a secret $S = (s^1, \ldots, s^\ell)$ containing ℓ elements from $\mathbb{F}$. Moreover, if D is corrupted, then he may commit $NULL$ instead of elements from $\mathbb{F}$ (the meaning of it will be clear in the sequel). Protocol AVSS-Share uses protocol AWSS-Share as a black-box. Protocol AVSS-Rec-Private and AVSS-Rec-Public enables private reconstruction of S (to any $P_\alpha \in \mathcal{P}$) and public reconstruction of S respectively. We call the private and public reconstruction as P_α*-private-reconstruction* and *public-reconstruction* respectively.

AVSS-Share($D, \mathcal{P}, S$)

DISTRIBUTION: CODE FOR D

1. For $l = 1, \ldots, \ell$, select a degree-t random symmetric bivariate polynomial $F^l(x, y)$ such that $F^l(0, 0) = s^l$ and send $f_i^l(x) = F^l(x, i)$ to P_i.

AWSS COMMITMENT AFTER VERIFICATION BY INDIVIDUAL PARTY: CODE FOR P_i

1. Wait to obtain $f_i^1(x), \ldots, f_i^\ell(x)$ from D.
2. If all the polynomials are of degree-t then as a dealer, execute AWSS-Share($P_i, \mathcal{P}, (f_i^1(x), \ldots, f_i^\ell(x))$) to *AWSS-commit* $(f_i^1(0), \ldots, f_i^\ell(0))$ using $(f_i^1(x), \ldots, f_i^\ell(x))$. The AWSS-Share initiated by P_i is called as AWSS-Share^{P_i}.
3. Wait to receive $(f_j^1(i), \ldots, f_j^\ell(i))$ from P_j in execution AWSS-Share^{P_j}. Check $f_i^l(j) \overset{?}{=} f_j^l(i)$ for all $l = 1, \ldots, \ell$. If the test passes then participate in AWSS-Share^{P_j} and act according to the remaining steps of AWSS-Share^{P_j}.

VCORE CONSTRUCTION: CODE FOR D

1. If AWSS-Share^{P_j} is terminated, then denote corresponding $WCORE$ and OKP_k sets by $WCORE^{P_j}$ and $OKP_k^{P_j}$ for every $P_k \in WCORE^{P_j}$. Add P_j in a set $VCORE$ (initially empty).
2. Keep updating $VCORE$, $WCORE^{P_j}$ and corresponding $OKP_k^{P_j}$'s for every $P_j \in VCORE$ upon receiving new A-casts of the form OK(.,.) (during AWSS-Share^{P_j}s), until for at least $2t+1$ $P_j \in VCORE$, the condition $|VCORE \cap WCORE^{P_j}| \geq 2t+1$ is satisfied. Remove (from $VCORE$) all $P_j \in VCORE$ for whom the above condition is not satisfied.
3. A-cast $VCORE$, $WCORE^{P_j}$ for $P_j \in VCORE$ and $OKP_k^{P_j}$ for every $P_k \in WCORE^{P_j}$.

VCORE VERIFICATION & AGREEMENT ON VCORE : CODE FOR P_i

1. Wait to receive $VCORE$, $WCORE^{P_j}$ for $P_j \in VCORE$ and $OKP_k^{P_j}$ for every $P_k \in WCORE^{P_j}$ from D's A-cast.
2. Wait to terminate AWSS-Share^{P_j} corresponding to every P_j in $VCORE$.
3. Wait to receive OK(P_m, P_k) for every $P_k \in WCORE^{P_j}$ and every $P_m \in OKP_k^{P_j}$, corresponding to every $P_j \in VCORE$.
4. Accept $VCORE$, $WCORE^{P_j}$ for $P_j \in VCORE$ and $OKP_k^{P_j}$ for every $P_k \in WCORE^{P_j}$ and terminate AVSS-Share.

AVSS-Rec-Private($D, \mathcal{P}, S, P_\alpha$**)**

P_α-WEAK-PRIVATE-RECONSTRUCTION OF $(f_j^1(x), \ldots, f_j^\ell(x))$ FOR EVERY $P_j \in VCORE$:

CODE FOR P_i

1. Participate in AWSS-Rec-Private($P_j, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x)), P_\alpha$) to enable P_α-weak-private-reconstruction of $(f_j^1(x), \ldots, f_j^\ell(x))$ for every $P_j \in VCORE$

LOCAL COMPUTATION: CODE FOR P_α

1. Add $P_j \in VCORE$ to $FINAL$ if P_α-weak-private-reconstruction of $(f_j^1(x), \ldots, f_j^\ell(x))$ is successful with degree-t polynomials $(\overline{f_j^1}(x), \ldots, \overline{f_j^\ell}(x))$.
2. For every pair $(P_\gamma, P_\delta) \in FINAL$ check $\overline{f_\gamma^l(\delta)} \overset{?}{=} \overline{f_\delta^l(\gamma)}$. If the test passes for every pair of parties in $FINAL$ then recover $\overline{F^l(x,y)}$ using $\overline{f_j^l(x)}$'s corresponding to each $P_j \in FINAL$ and reconstruct $\overline{s^l} = \overline{F^l(0,0)}$. Else reconstruct $\overline{s^l} = NULL$. Finally output $\overline{S} = (\overline{s^1}, \ldots, \overline{s^\ell})$ when every $\overline{s^l}$ is nonNULL otherwise output $\overline{S} = NULL$ and terminate **AVSS-Rec-Private**.

AVSS-Rec-Public($D, \mathcal{P}, S$**):** Public-reconstruction of S: Same as AVSS-Rec-Private except the following: instead of P_α-weak-private-reconstruction, weak-public-reconstruction of $(f_j^1(x), \ldots, f_j^\ell(x))$ for every $P_j \in VCORE$ is performed. Then every P_i, acting as P_α executes the code in [LOCAL COMPUTATION:CODE FOR P_α] to reconstruct $\overline{S}$.

Remark 6 (D's AVSS-commitment). We say that D has AVSS-committed $S = (s^1, \ldots, s^\ell) \in \mathbb{F}^\ell$ in AVSS-Share if for every $l = 1, \ldots, \ell$ there is a unique degree-t symmetric bivariate polynomial $F^l(x,y)$ such that $F^l(0,0) = s^l$ and every *honest* P_i in $VCORE$ receives $f_i^l(x) = F^l(x,i)$ from D and *AWSS-commits* $f_i^l(0)$ using $f_i^l(x)$ among the parties in $WCORE^{P_i}$. Otherwise, we say that D has committed $NULL$. Notice that the above condition implies that for $l = 1, \ldots, \ell$ there exist a unique degree-t univariate polynomial $f^l(x)(= f_0^l(x) = F^l(x,0))$ such that $f^l(0) = s^l$ and every honest $P_i \in VCORE$ receives $f^l(i)(= f_0^l(i) = f_i^l(0))$ from D. **The value $f^l(i)$ is referred as i^{th} share of** s^l. An honest D always commits s^l from $\mathbb{F}$ as he always chooses a proper symmetric bivariate polynomial $F^l(x,y)$ and properly distributes $f_i^l(x) = F^l(x,i)$ to party P_i. But AVSS-Share can *not* ensure that corrupted D also commits $s^l \in \mathbb{F}$ for all l. When a corrupted D commits $NULL$, the $f_i^l(x)$ polynomials of the honest parties in $VCORE$ do not define a degree-t symmetric bivariate polynomial for at least one l implying that there will be an honest pair (P_γ, P_δ) in $VCORE$ such that $f_\gamma^l(\delta) \neq f_\delta^l(\gamma)$.

Notation 4. *In subsequent sections, we will invoke AVSS-Share as AVSS-Share(D, $\mathcal{P}, (f^1(x), \ldots. f^\ell(x))$) to mean that D AVSS-commits to $f^l(0)$ using $f^l(x)$ for $l = 1, \ldots, \ell$ in AVSS-Share. Essentially here D is asked to choose a bivariate polynomial $F^l(x,y)$ such that $F^l(x,0) = f^l(x)$. By doing this every honest $P_i \in VCORE$ should ideally receive $f^l(i)$. Similarly, AVSS-Rec-Private will be invoked as AVSS-Rec-Private($D, \mathcal{P}, (f^1(x), \ldots, f^\ell(x)), P_\alpha$) to enable P_α-private-reconstruction of $(f^1(x), \ldots, f^\ell(x))$ and hence $(f^1(0), \ldots, f^\ell(0))$. AVSS-Rec-Public will also be invoked in a similar way.*

Theorem 6. *AVSS-Share communicates $\mathcal{O}((\ell n^3+n^4\kappa)\kappa)$ and A-casts $\mathcal{O}(n^3 \log n)$ bits. AVSS-Rec-Private and AVSS-Rec-Public communicates $\mathcal{O}((\ell n^3+n^4\kappa)\kappa)$ and $\mathcal{O}((\ell n^5+n^6\kappa)\kappa)$ bits respectively.*

3.6 Extended AVSS Protocol

Protocol Extd-AVSS-Share($D, \mathcal{P}, (\mathcal{F}^0, \ldots, \mathcal{F}^{\mathcal{B}})$)

DISTRIBUTION: CODE FOR D – Only D executes this code.

1. For every $\mathcal{F}^b$, select η random, symmetric bivariate polynomials $F^{1b}(x,y), \ldots, F^{\eta b}(x,y)$ of degree-t in x and y, such that $F^{kb}(x,0) = f^{kb}(x)$ for $k = 1, \ldots, \eta$ and $b = 0, \ldots, \mathcal{B}$. Let for $i = 1, \ldots, n$, $f_i^{kb}(x) = F^{kb}(x,i)$.
2. For every $b = 0, \ldots, \mathcal{B}$, send $(f_i^{1b}(x), \ldots, f_i^{\eta b}(x))$ to P_i.

AWSS-COMMITMENT AFTER VERIFICATION BY INDIVIDUAL PARTY: CODE FOR P_i

1. Wait to obtain $(f_i^{1b}(x), \ldots, f_i^{\eta b}(x))$ for all b from D.
2. If all the polynomials are of degree-t then as a dealer, execute Extd-AWSS-Share($P_i, \mathcal{P}, (\Delta_i^0, \ldots, \Delta_i^{\mathcal{B}})$) where $\Delta_i^b = (f_i^{1b}(x), \ldots, f_i^{\eta b}(x))$ for $b = 0, \ldots, \mathcal{B}$. The Extd-AWSS-Share initiated by P_i is called as Extd-AWSS-Share^{P_i}.
3. Wait to receive $\Delta_j^b(i) = (f_j^{1b}(i), \ldots, f_j^{\eta b}(i))$ for all b from P_j in execution AWSS-Share^{P_j}. Check $f_i^{kb}(j) \overset{?}{=} f_j^{kb}(i)$ for all $k = 1, \ldots, \eta$. Do the same for all $b = 0, \ldots, \mathcal{B}$. If the test passes then participate in Extd-AWSS-Share^{P_j} and act according to the remaining steps of Extd-AWSS-Share^{P_j}.

VCORE CONSTRUCTION and [VCORE VERIFICATION & AGREEMENT ON VCORE] are same as in AVSS-Share.

Protocol Compute-AVSS-Commit($D, \mathcal{P}, r, (\mathcal{F}^0, \ldots, \mathcal{F}^{\mathcal{B}})$)

CODE FOR P_i

1. Participate in computing AWSS-commitment of Δ_i^* for every $P_i \in VCORE$ where $\Delta_i^* = \sum_{b=0}^{\mathcal{B}} r^b \Delta_i^b$ by executing Compute-AWSS-Commit. This will generate AVSS-commitment of $\mathcal{F}^*$.

Protocol Rec-AVSS-Commit($D, \mathcal{P}, \mathcal{F}^*$)

1. Parties execute AVSS-Rec-Public($D, \mathcal{P}, \mathcal{F}^* = (f^{1*}(x), \ldots, f^{\eta *}(x))$) to enable public-reconstruction of $\mathcal{F}^* = (f^{1*}(x), \ldots, f^{\eta *}(x))$ and hence $S^* = (f^{1*}(0), \ldots, f^{\eta *}(0))$.

As described in Extended AWSS protocol, we now extend our AVSS protocol for the same situation, where D has $\mathcal{B} = n^2$ blocks, denoted by $S^0, \ldots, S^{\mathcal{B}}$, each containing $\eta = \frac{m}{n^2}$ secrets with $S^b = (s^{1b}, \ldots, s^{\eta b})$. Now D wants to AVSS-commit S^b using degree-t polynomials say, $\mathcal{F}^b = (f^{1b}(x), \ldots, f^{\eta b}(x))$ (where $f^{kb}(0) = s^{kb}$) for every $b \in \{0, \ldots, \mathcal{B}\}$ such that later the parties can generate (by local computation) AVSS-commitment of $S^* = \sum_{b=0}^{\mathcal{B}} r^b.S^b = (s^{1*}, \ldots, s^{\eta *})$ using polynomials $\mathcal{F}^* = (f^{1*}(x), \ldots, f^{\eta *}(x))$ where $f^{k*}(x) = \sum_{b=0}^{\mathcal{B}} r^b f^{kb}(x)$ and $f^{k*}(0) = s^{k*}$. Here r is an agreed upon random value which will be generated *after* every honest party ensures that D is AVSS-committed to every S^b. Later the parties can reconstruct S^* by invoking AVSS-Rec-Public($D, \mathcal{P}, (f^{1*}(x), \ldots, f^{\eta *}(x))$). To achieve the above task, we first extend protocol AVSS-Share to design Extd-AVSS-Share that uses Extd-AWSS-Share as building block. We then give protocols for generating AVSS-commitment of S^* (protocol Compute-AVSS-Commit) and revealing the same (protocol Rec-AVSS-Commit).

Remark 7. Notice that once $VCORE$ is constructed and agreed upon in Extd-AVSS-Share, then the parties can later enable P_α-private-reconstruction of any $\mathcal{F}^b$, where $b \in \{1, \ldots, \mathcal{B}\}$ and $P_\alpha \in \mathcal{P}$.

Theorem 7. *(i) If D is honest then revealing S^* during Rec-AVSS-Commit will not reveal any information about $S^1, \ldots, S^{\mathcal{B}}$; (ii) If D is corrupted and S^* revealed during Rec-AVSS-Commit is non-NULL (i.e., $S^* \in \mathbb{F}^\eta$), then with very high probability D has AVSS-Committed non-NULL $S^0, \ldots, S^{\mathcal{B}}$.*

Theorem 8. *Protocol Extd-AVSS-Share privately communicates $\mathcal{O}((mn^3 + n^6\kappa)\kappa)$ and A-casts $\mathcal{O}(n^3 \log n)$ bits. Compute-AVSS-Commit does not require any communication. Rec-AVSS-Commit privately communicates $\mathcal{O}((mn^3 + n^6\kappa)\kappa)$ bits.*

3.7 Statistical Asynchronous Complete Secret Sharing

As specified in Remark 6, in AVSS-Share a corrupted D may commit $NULL$ and more importantly AVSS-Share ensures that *only* the honest parties in $VCORE$ receive their respective shares of the committed secret. As it may happen that potentially t honest parties are *not* present in $VCORE$, AVSS-Share lacks *completeness* property. So we now outline how our AVSS scheme can be further used to design an ACSS protocol. Specifically, we present an ACSS scheme called ACSS, consisting of sub-protocols ACSS-Share, ACSS-Rec-Private and ACSS-Rec-Public. ACSS-Share allows D to share secret S containing m field elements from $\mathbb{F}$. Given the sharing of S, satisfying *completeness* property, ACSS-Rec-Private allows a specific party say P_α to privately reconstruct S. On the other hand, ACSS-Rec-Public allows every party in $\mathcal{P}$ to reconstruct S. Both ACSS-Rec-Private and ACSS-Rec-Public use Online Error Correcting (OEC) algorithm [9]. In ACSS-Share, we come across a situation where the parties jointly need to generate a random number. It can be achieved as follows:

Random Number Generation: Each $P_i \in \mathcal{P}$ shares a random non-zero $r_i \in \mathbb{F}$ using AVSS-Share (here $\ell = 1$). The parties then run ACS to agree on a common set, say $\mathcal{C}$ of at least $2t + 1$ parties who did proper sharing of their r_i's. Once $\mathcal{C}$ is agreed upon, AVSS-Rec-Public is executed for every $P_i \in \mathcal{C}$ in order to reconstruct back P_i's committed secret. Now every party in $\mathcal{P}$ locally add the committed secret of every $P_i \in \mathcal{C}$ such that the committed secret is non-NULL. Now it is easy to see that the sum value is random. We call this protocol as RNG, which privately communicates $\mathcal{O}(n^7\kappa^2)$ bits. $\square$

In ACSS-Share, if $CCORE$ is agreed upon, then the parties generate a random number r and checks whether $S^* = (s^{1*}, \ldots, s^{\eta*}) = \sum_{b=0}^{\mathcal{B}} r^b S^b$ is from $\mathbb{F}^\eta$ by doing public-reconstruction of the same. As S^0 was chosen to be random, S^* does not leak any information about S^b's $(b \neq 0)$ when D is honest. Moreover, by Theorem 7, if S^* is reconstructed as nonNULL, then with very high probability each of the individual S^b's is also nonNULL.

ACSS-Share($D, \mathcal{P}, S$)

i. Distribution by D: Code for D – Only D executes this code

1. Divide S into $\mathcal{B} = n^2$ blocks, denoted by $S^1, \ldots, S^{\mathcal{B}}$, each containing $\eta = \frac{m}{n^2}$ secrets. Let $S^b = (s^{1b}, \ldots, s^{\eta b})$. Also select η random secrets $S^0 = (s^{10}, \ldots, s^{\eta 0})$.
2. For each S^b, $b \in \{0, \ldots, \mathcal{B}\}$, select η random bivariate polynomials $F^{1b}(x,y), \ldots, F^{\eta b}(x,y)$ of degree-t in x and y such that $F^{kb}(0,0) = s^{kb}$ for $k = 1, \ldots, \eta$.
3. For each block S^b, send polynomials $g_i^{kb}(y) = F^{kb}(i,y)$ for all $k = 1, \ldots, \eta$ to P_i.
4. Let $\Delta_i^b = (f_i^{1b}(x), \ldots, f_i^{\eta b}(x))$ for $b = 0, \ldots, \mathcal{B}$, where $f_i^{kb}(x) = F^{kb}(x,i)$. For $i = 1, \ldots, n$, initiate Extd-AVSS-Share($D, \mathcal{P}, (\Delta_i^0, \ldots, \Delta_i^{\mathcal{B}})$) for sharing $(\Delta_i^0, \ldots, \Delta_i^{\mathcal{B}})$ simultaneously. The i^{th} instance of Extd-AVSS-Share is denoted by Extd-AVSS-Share^i.

ii. Code for P_i – Every party in $\mathcal{P}$, including D, executes this code

1. For each S^b, wait to receive polynomials $g_i^{kb}(y)$ for all $k = 1, \ldots, \eta$ from D.
2. Participate in Extd-AVSS-Share($D, \mathcal{P}, (\Delta_j^0, \ldots, \Delta_j^{\mathcal{B}})$) for all $j = 1, \ldots, n$.
3. For each S^b, if $(f_j^{1b}(i), \ldots, f_j^{\eta b}(i))$ is received from Extd-AVSS-Share($D, \mathcal{P}$, $(\Delta_j^0, \ldots, \Delta_j^{\mathcal{B}})$) then check whether $g_i^{kb}(j) = f_j^{kb}(i)$ holds for all $k = 1, \ldots, \eta$. If the test passes for all $b = 0, \ldots, \mathcal{B}$ and $j = 1, \ldots, n$, then A-cast `Matched`.

iii. CCORE Construction: Code for D – Only D executes this code.

1. Construct $VCORE$ for every Extd-AVSS-Share^i. Corresponding to Extd-AVSS-Share^i, the sets are denoted by $VCORE^i$, $WCORE_j^i$ for every $P_j \in VCORE^i$ and $OKP_{kj}^{\;i}$ for every $P_k \in WCORE_j^i$.
2. Keep updating these sets until $CCORE = \cap_{i=1}^{n} VCORE^i$ of size at least $2t+1$ is obtained and `Matched` is received from A-cast of every $P_j \in CCORE$.
3. Make the parties agree on $CCORE$ and corresponding sets by A-casting them and letting all other parties to verify (follows almost the same steps as in [VCORE verification & Agreement] in Protocol AVSS-Share).

iv. Checking D's commitment: Code for P_i

1. Participate in RGB to generate and agree on a common random number, say r.
2. Participate in computing AVSS-commitment of Δ_j^* for every $j = 1, \ldots, n$ where $\Delta_j^* = \sum_{b=0}^{\mathcal{B}} r^b \Delta_j^b$ by executing Compute-AVSS-Commit($D, \mathcal{P}, r, (\Delta_j^0, \ldots, \Delta_j^{\mathcal{B}})$).
3. Participate in Rec-AVSS-Commit($D, \mathcal{P}, \Delta_j^*$) to publicly reconstruct Δ_j^* for $j = 1, \ldots, n$. Let $\Delta_j^* = (f_j^{1*}(x), \ldots, f_j^{\eta *}(x))$.
4. Conclude that D has committed secrets from $\mathbb{F}$ and proceed to the next step, if for $k = 1, \ldots, \eta$, there is a bivariate polynomial of degree-t in x and y, say $F^{k*}(x,y)$ with $F^{k*}(x,i) = f^{k*}(x)$. Let $s^{k*} = F^{k*}(0,0)$ for all k and $S^* = (s^{1*}, \ldots, s^{\eta *})$.

v. For $j = 1, \ldots, n$, P_j-private-reconstruction of $(f_j^{1b}(0), \ldots, f^{\eta b}(0))$ for all $b \in \{0, \ldots, \mathcal{B}\}$: Code for P_i

1. Participate in AVSS-Rec-Private($D, \mathcal{P}, (f_j^{1b}(x), \ldots, f_j^{\eta b}(x)), P_j$), for $b = 1, \ldots, \mathcal{B}$ and $j = 1, \ldots, n$, to enable P_j-private-reconstruction of $(f_j^{1b}(0), \ldots, f_j^{\eta b}(0))$ using the sets ($CCORE$, $WCORE_i^j$ etc.) agreed before.
2. Output $(f_i^{1b}(0), \ldots, f_i^{\eta b}(0))$ as i^{th} share of $(s^{1b}, \ldots, s^{\eta b})$ after the completion of P_i-private-reconstruction of $(f_i^{1b}(0), \ldots, f_i^{\eta b}(0))$ for all $b = 1, \ldots, \mathcal{B}$.

ACSS-Rec-Private($D, \mathcal{P}, S, P_\alpha$): Private reconstruction of S by P_α:

1. Code for P_i:For every $b = 1, \ldots, \mathcal{B}$ and $k = 1, \ldots, \eta$, send $f_i^{kb}(0) (= f_0^{kb}(i))$, the i^{th} share of secret s^{kb} to P_α.
2. Code for P_α: Apply *On-line error correction* [5, 9] on the received shares to reconstruct s^{kb} for every $b = 1, \ldots, \mathcal{B}$ and $k = 1, \ldots, \eta$

ACSS-Rec-Public($D, \mathcal{P}, S, \mathcal{P}$): Public reconstruction of S: Run ACSS-Rec-Private($D, \mathcal{P}, S, P_\alpha$) for every $P_\alpha \in \mathcal{P}$.

Lemma 2. *In ACSS-Share, if CCORE is agreed upon and $S^* \in \mathbb{F}^\eta$, then for $b = 0, \ldots, \mathcal{B}$ and $k = 1, \ldots, \eta$, D has committed a unique degree-(t, t) bivariate polynomial $\overline{F^{kb}}(x, y)$ (hence unique secret $\overline{s^{kb}} = \overline{F^{kb}}(0, 0)$), such that for every i, polynomial $f_i^{kb}(x)$ satisfies $f_i^{kb}(x) = \overline{F^{kb}}(x, i)$ and for every honest $P_j \in CCORE$, $g_j^{kb}(y) = \overline{F^{kb}}(j, y)$ holds. If D is honest then $\overline{F^{kb}}(x, y) = F^{kb}(x, y)$.*

Now it is clear from the description of ACSS-Share, that the generation of random number and then the public verification of $S^* = (s^{1*}, \ldots, s^{\eta*}) = \sum_{b=0}^{\mathcal{B}} r^b S^b$ to be from $\mathbb{F}^\eta$ ensures that each S^b's are also committed from $\mathbb{F}^\eta$. The main reason that D divides the secrets into n^2 blocks each of size $\eta = \frac{m}{n^2}$ and discloses S^* of size η is that it keeps the communication cost of the public reconstruction (of S^*) to be $\mathcal{O}((mn^4 + n^7\kappa)\kappa)$ bits. Otherwise if the secrets are not divided into blocks, then S^* would have been of size m and its public reconstruction would take $\mathcal{O}((mn^6 + n^5\kappa)\kappa)$ bits. Now once above lemma statement is assured, then by the property of OEC [9], P_i-private-reconstruction of $f_i^{kb}(0)$ will be successful for every $b = 1, \ldots, \mathcal{B}$ and $k = 1, \ldots, \eta$ and hence finally every (honest) P_i will output $f_i^{kb}(0)$ as the i^{th} share of $\overline{s^{kb}}$.

Theorem 9. *ACSS-Share privately communicates $\mathcal{O}((mn^4 + n^7\kappa)\kappa)$ bits and A-casts $\mathcal{O}(n^4 \log n)$ bits. ACSS-Rec-Private and ACSS-Rec-Public privately communicate $\mathcal{O}(mn\kappa)$ and $\mathcal{O}(mn^2\kappa)$ bits respectively.*

Notation 5. *Following previous notations for AWSS and AVSS, we will invoke ACSS-Share as ACSS-Share($D, \mathcal{P}, (h^1(x), \ldots, h^m(x))$) and by doing so, we mean that D executes ACSS-Share with m degree-t polynomials $h^1(x), \ldots, h^m(x)$ such that for $l = 1, \ldots, m$, $h^l(0) = s^l$. As a result of this execution, each party P_i gets the shares $h^1(i), \ldots, h^m(i)$.*

Definition 1 (t-1D-sharing). *We say that a secret s is t-1D-shared, denoted as $[s]_t$, if there exists a degree-t polynomial $h(x)$, with $h(0) = s$, such that each (honest) P_i holds the i^{th} share $h(i) = s_i$ of s.*

Notice that ACSS-Share generates t-1D-sharing of m secrets simultaneously.

4 Our Statistical AMPC Protocol

Once we have an ACSS scheme for generating t-1D-sharing of m secrets simultaneously, we can design an efficient statistical AMPC protocol with $n = 3t + 1$. However, due to space constraints, the readers are referred to Section 9 of [16] for the details of AMPC.

5 Open Problem

It would be interesting to further improve the communication complexity of AMPC with $n = 3t + 1$.

References

1. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 420–432. Springer, Heidelberg (1992)
2. Beerliová-Trubíniová, Z., Hirt, M.: Efficient multi-party computation with dispute control. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 305–328. Springer, Heidelberg (2006)
3. Beerliová-Trubíniová, Z., Hirt, M.: Simple and efficient perfectly-secure asynchronous mpc. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 376–392. Springer, Heidelberg (2007)
4. Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure MPC with linear communication complexity. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 213–230. Springer, Heidelberg (2008)
5. Ben-Or, M., Canetti, R., Goldreich, O.: Asynchronous secure computation. In: STOC, pp. 52–61 (1993)
6. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC, pp. 1–10 (1988)
7. Ben-Or, M., Kelmer, B., Rabin, T.: Asynchronous secure computations with optimal resilience. In: PODC, pp. 183–192 (1994)
8. Bracha, G.: An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol. In: PODC, pp. 154–162 (1984)
9. Canetti, R.: Studies in Secure Multiparty Computation and Applications. PhD thesis, Weizmann Institute, Israel (1995)
10. Canetti, R., Rabin, T.: Fast asynchronous Byzantine Agreement with optimal resilience. In: STOC, pp. 42–51 (1993)
11. Chaum, D., Crpeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: STOC, pp. 11–19 (1988)
12. Cramer, R., Damgård, I.: Multiparty Computation, an Introduction. In: Contemporary Cryptography. Birkhuser, Basel (2005)
13. Cramer, R., Damgård, I., Dziembowski, S., Hirt, M., Rabin, T.: Efficient multiparty computations secure against an adaptive adversary. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 311–326. Springer, Heidelberg (1999)
14. Damgård, I., Nielsen, J.B.: Scalable and unconditionally secure multiparty computation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 572–590. Springer, Heidelberg (2007)
15. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified VSS and fact-track multiparty computations with applications to threshold cryptography. In: PODC, pp. 101–111 (1998)
16. Patra, A., Choudhary, A., Pandu Rangan, C.: Efficient statistical asynchronous verifiable secret sharing and multiparty computation with optimal resilience. In: Cryptology ePrint Archive, Report 2009/492. A preliminary version of this paper got accepted in ICITS 2009 (2009)
17. Patra, A., Choudhary, A., Pandu Rangan, C.: Simple and efficient asynchronous Byzantine Agreement with optimal resilience. In: Cryptology ePrint Archive, Report 2008/424. Also appeared in Proc. of PODC (2009)
18. Rabin, T.: Robust sharing of secrets when the dealer is honest or cheating. J. ACM 41(6), 1089–1109 (1994)
19. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: STOC, pp. 73–85 (1989)
20. Yao, A.C.: Protocols for secure computations. In: FOCS, pp. 160–164 (1982)

Gemstone: A New Stream Cipher Using Coupled Map Lattice

Ruming Yin, Jian Yuan, Qiuhua Yang, Xiuming Shan, and Xiqin Wang

Department of Electronic Engineering,
Tsinghua University, Beijing 100084, China
yrm05@mails.tsinghua.edu.cn, jyuan@tsinghua.edu.cn

Abstract. In this paper, we propose a new stream cipher Gemstone by discretizing coupled map lattices (CML), which is a nonlinear system of coupled chaotic maps. Gemstone uses a 128-bit key and a 64-bit initialization vector (IV). We show that there is no high probability difference propagations or high correlations over the IV setup scheme. Thus the IV setup of Gemstone is very secure. We also verify that the largest linear correlations between consecutive key streams are below the safe bounds. Gemstone is slightly slower than AES-CTR, but its initialization speeds are higher than some finalists of eSTREAM.

Keywords: Stream cipher, coupled map lattice, confusion and diffusion, differential cryptanalysis, linear cryptanalysis.

1 Introduction

A stream cipher usually generates a long pseudo-random keystream and encryptes the plaintext by the bitwise XOR of the keystream and the plaintext. Stream ciphers can be designed to be faster than general block ciphers. Additionally, they have limited error propagation. With these desirable properties, many stream cipher algorithms have been published in recent years [22]. Most of these ciphers try to follow the confusion and diffusion principles from Shannon to strengthen the security. Typically, various highly nonlinear S-boxes are widely used [23]. In this paper, following these design principles, we propose a new stream cipher for software Gemstone. Gemstone uses a 128-bit key and a 64-bit initialization vector (IV).

Like the famous stream cipher Rabbit [15], which is constructed by using chaos, the design of Gemstone is inspired by coupled map lattice (CML). CML a real-valued nonlinear system of coupled chaotic maps [24]. The original CML based ciphers suffer from some defects [19-21]. First, the security can not be substantially improved by cryptographic techniques, since typically cryptographic operations are performed on binary numbers. In addition, the operations on real numbers are usually realized by using floating-point arithmetic. Due to the different implementation of floating-point operations on various processors, the ciphers may be difficult to realize synchronization between the sender and the receiver. Therefore, it is essential to properly discretize the CML to make it operate on binary numbers. Thus the constructed cipher can be advantageous for security improvement as well as the performance.

In this paper, we properly discretize the CML and design a new stream cipher. CML consists of local chaotic map and spatial coupling. From the point of view of cryptosystem, the local nonlinear chaotic map provides confusion and the coupling operation achieves diffusion. When discretizing CML, we preserve this good confusion and diffusion property. The discretized CML is then used to design a stream cipher. In the design, we mainly consider two kinds of attacks. One type is the attack against IV setup. IV setup has been proven to be a crucial component in stream ciphers [11-13]. In a typical scenario, an attacker can compare the keystream sequences associated with several known or chosen IV values, and distinguish the IV setup algorithm from a random function. With this type of attacks in mind, we investigate the difference propagations and the linear correlations over the IV setup and verify that no distinguishability from random is possible. Another type of attack that is considered explores the high linear correlation between key streams. This is also a major concern in the recent designs of stream ciphers [14, 15]. For our cipher, we show that the largest linear correlations between consecutive key streams are below the safe bounds.

We also compare the performance of our stream cipher with some recent stream cipher proposals by using the eSTREAM testing framework. The test results show that Gemstone is slightly slower than AES-CTR. However, the initialization speeds of Gemstone are higher than some finalists of eSTREAM.

Notation. Throughout this paper, $\oplus$ denotes bitwise XOR, $\boxplus$ denotes addition modulo 2^{16}, $\diamond$ denotes concatenation of two bit sequences. $x^{[u_1 \cdots u_2]}$ means bit number u_1 through u_2 of variable x. Following the convention, the least significant bit is denoted by 0.

2 A Discretized CML Model

The CML consisting of skew tent maps can be constructed as

$$\begin{aligned} z_{i,n+1} &= (1-\varepsilon)v(z_{i,n}) + \frac{\varepsilon}{2}[v(z_{i-1,n}) + v(z_{i+1,n})], \\ & i = 0, 1, \cdots, L-1, \end{aligned} \tag{1}$$

where L is the number of the sites. $z_{i,n} \in (0,1)$ represents the real-valued state variable for the site i at time $n\,(n = 0, 1, \cdots)$. $\varepsilon \in (0,1)$ is a coupling constant. The periodic boundary condition is used, i.e., $z_{-1,n} = z_{L-1,n}$, $z_{L,n} = z_{0,n}$. $v(\cdot)$ is the chaotic skew tent map which is defined as follows

$$v(z) = \begin{cases} \frac{z}{p} & 0 \le z \le p \\ \frac{z-p}{1-p} & p < z \le 1 \end{cases} \tag{2}$$

where $p \in (0,1)$ is the parameter.

We discretize CML to make it operates on integer numbers while preserving the basic structure for good confusion and diffusion. Firstly, the skew tent map as the local map has to be discretized. The chaotic skew tent map is nonlinear and can exhibit random-like behavior. From the point of view of cryptosystem, this map creats confusion in CML. In the best case, we would like to discretize the skew tent map to obtain a one-to-one substitution. Thus the improved CML can be bijective, which is desirable for our stream cipher. In this paper, the discretized skew tent map proposed in [1] is adopted. It is a one-to-one mapping on $[1, M]$. Here $[1, M]$ denotes the set of all integers between 1 and M including both. The discretized map V is as follows

$$V(Z) = \begin{cases} \lceil \frac{MZ}{A} \rceil & 1 \leq Z \leq A \\ \lfloor \frac{M(M-Z)}{M-A} \rfloor + 1 & A < Z \leq M \end{cases} . \tag{3}$$

Where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ are the floor and ceiling operators respectively. The parameters Z, A, M are integers. $M = 2^m$. Z, $A \in [1, M]$. The cryptosystem is usually required to operate on the interval $[0, M-1]$. Therefore we further make improvements for the map V. We iterate the discretized map N times and obtain the substitution S on $[0, M-1]$ as

$$S(T) = V^N(T+1) - 1. \tag{4}$$

The iteration is made to obtain the excellent properties of confusion. Finally S is selected as the local map in our improved CML.

On the other hand, the coupling operation in CML has to be improved. The nearest-neighbor coupling provides diffusion. We hope that the modified coupling operation can still preserve this diffusion property and is also deeply investigated in modern cryptosystems. Thus the improved CML can be clearly analyzed by using cryptographic technology. On the basis of these considerations, addition modulo $M = 2^m$ is used to modify the nearest-neighbor coupling. This modified coupling component will be called mixing transformation following the general description of block ciphers.

Additionally, we modify the CML by first applying the mixing transformation and then performing substitution. The aim of this modification is to faciliate the security analysis of our cipher (see Section 5.3). Thus the improved CML can be formulated as follows

$$y_n = Dx_n \ mod \ M, \quad g_n = S(y_n). \tag{5}$$

Where x_n is the vector consisting of L state variables, i.e., $x_n = (x_{L-1,n}, x_{L-2,n}, \cdots, x_{0,n})^t$. Similarly, $y_n = (y_{L-1,n}, y_{L-2,n}, \cdots, y_{0,n})^t$, $g_n = (g_{L-1,n}, g_{L-2,n}, \cdots, g_{0,n})^t$, $S(y_n) = (S(y_{L-1,n}), S(y_{L-2,n}), \cdots, S(y_{0,n}))^t$.

$$D = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 1 \\ 1 & 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 1 & \cdots & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & 0 & \cdots & 1 \end{pmatrix}_{L\times L} .$$

The t suffix denotes transposition of the vector. We will use this discretized CML model to construct the state-update function of our cipher. In our stream cipher, the parameters of the discretized CML are selected as follows. The number of the sites $L = 8$. The parameters of S-box are $A = 2^{14} + 11 = 16409$ and $N = 20$. Addition modulo $M = 2^{16}$ is chosen to modify the nearest-neighbor coupling in CML. In this case the improved CML function, denoted by $\mathbf{g}$, is shown in Fig. 1. In the figure, D represents the mixing transformation and S denote substitutions.

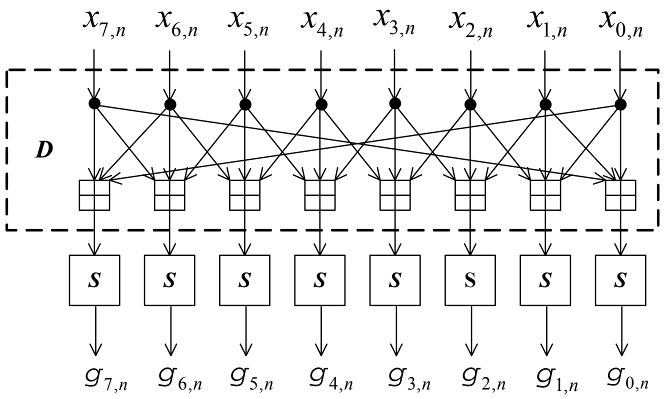


Fig. 1. Function $\mathbf{g}$: the discretized coupled map lattice with parameters $L = 8$, $A = 16409$, $N = 20$ and $M = 2^{16}$

3 Description of the Stream Cipher

In this section, we describe our stream cipher. The proposed cipher is a synchronous stream cipher which uses a 128-bit key K and a 64-bit initialization vector IV. The 256-bit internal state of the cipher are divided into eight 16-bit state variables $x_{i,n}$ and eight 16-bit counter variables $c_{i,n}$. $x_{i,n}$ is the i-th 16-bit state variable at iteration n and $c_{i,n}$ is the corresponding 16-bit counter variable. The cipher works in two phases: first the internal state of the cipher is initialized using the key and initialization vector, then the state is repeatedly updated and used to generate the key stream bits.

3.1 The State-Update Function

The state-update function is shown in Fig. 2. In this function, the discretized CML is first iterated two times as

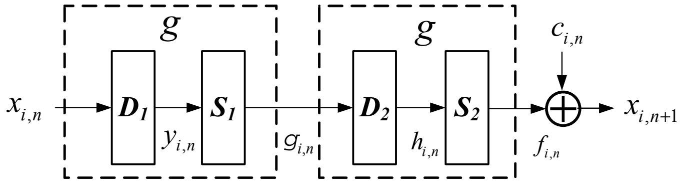


Fig. 2. The state-update function

$$\begin{aligned}
y_{j,n}&=x_{j-1,n} \boxplus x_{j,n} \boxplus x_{j+1,n}\\
g_{j,n}&=S(y_{j,n})\\
h_{j,n}&=g_{j-1,n} \boxplus g_{j,n} \boxplus g_{j+1,n}\\
f_{j,n}&=S(h_{j,n}), \quad j = 0, 1, \cdots, 7
\end{aligned} \tag{6}$$

where $x_{-1,n} = x_{7,n}$, $x_{8,n} = x_{0,n}$, $g_{-1,n} = g_{7,n}$, $g_{8,n} = g_{0,n}$. Then a counter is used to expand the cycle length of the key stream. The counter-assisted algorithm is as follows

$$x_{i,n+1} = f_{i,n} \oplus c_{i,n},\ i = 0, 1, \cdots, 7. \tag{7}$$

The selected counter is shown in Fig. 3. It is a maximum-length LFSR of length 128 over $\mathbb{F}_2$ defined by the following primitive feedback polynomial [2]:

$$q(\gamma) = \gamma^{128} + \gamma^7 + \gamma^2 + \gamma + 1. \tag{8}$$

The 128-bit counter $C_n^{[127\cdots 0]}$ are divided into eight 16-bit counter variables $c_{0,n} = C^{[15\cdots 0]}, c_{1,n} = C^{[31\cdots 16]}, \cdots, c_{7,n} = C^{[127\cdots 112]}$. These eight counter variables are updated at each iteration as follows.

$$\begin{aligned}
c_{i,n+1}&=c_{i+1,n}^{[0]} \diamond c_{i,n}^{[15\cdots 1]} && 0 \le i \le 6\\
c_{i,n+1}&=\varphi \diamond c_{i,n}^{[15\cdots 1]} && i = 7
\end{aligned} \tag{9}$$

where $\varphi = c_{7,n}^{[15]} \oplus c_{7,n}^{[14]} \oplus c_{7,n}^{[9]} \oplus c_{0,n}^{[0]}$.

3.2 Key Setup

Firstly the counter variables are initialized with the carefully selected values as

$$\begin{aligned}
c_{7,0} &= 0x0123 & c_{6,0} &= 0x4567\\
c_{5,0} &= 0x89AB & c_{4,0} &= 0xCDEF\\
c_{3,0} &= 0x3210 & c_{2,0} &= 0x7654\\
c_{1,0} &= 0xBA98 & c_{0,0} &= 0xFEDC.
\end{aligned}$$

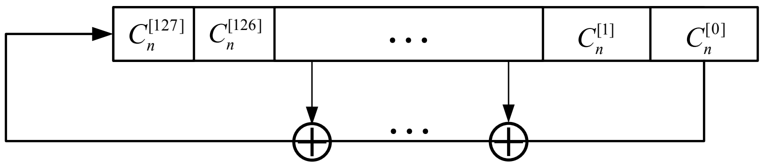


Fig. 3. The selected 128-bit Linear Feedback Shift Register

Then the 128-bit key $K^{[127\cdots0]}$ is divided into 8 subkeys $k_0 = K^{[15\cdots0]}, k_1 = K^{[31\cdots16]}, \cdots, k_7 = K^{[127\cdots112]}$. Each subkey is loaded into a state variable as follows

$$x_{i,0} = k_i,\ i = 0, 1, \cdots, 7. \tag{10}$$

Then, the state-update function is iterated three times to spread the influence of each key bit over all the state variable bits. At last, the counter variables are modified as

$$c_{i,3} = c_{i,3} \oplus x_{i,3},\ i = 0, 1, \cdots, 7. \tag{11}$$

3.3 IV Setup

The 64-bit initialization vector $IV^{[63\cdots0]}$ is divided into 4 16-bit variables as $IV_0 = IV^{[15\cdots0]}, IV_1 = IV^{[31\cdots16]}$, $IV_2 = IV^{[47\cdots32]}, IV_3 = IV^{[63\cdots48]}$. These IVs are used to modify the state variables as follows

$$\begin{aligned} x_{i,3} &= IV_i \oplus x_{i,3}, \quad i = 0, 1, 2, 3, \\ x_{i,3} &= IV_{i-4} \oplus x_{i,3},\ i = 4, 5, 6, 7. \end{aligned} \tag{12}$$

Then, the state-update function is iterated two times to spread the influence of each IV bit over all the state variable bits.

3.4 Key Stream Generation and Encryption/Decryption

After each iteration, 64 bits of key stream $s_n = s_{3,n} \diamond s_{2,n} \diamond s_{1,n} \diamond s_{0,n}$ are generated as follows:

$$\begin{aligned} s_{3,n} &= x_{7,n} \oplus x_{3,n} \qquad & s_{2,n} &= x_{6,n} \oplus x_{2,n} \\ s_{1,n} &= x_{5,n} \oplus x_{1,n} \qquad & s_{0,n} &= x_{4,n} \oplus x_{0,n}. \end{aligned} \tag{13}$$

In the encryption phase, the plaintext is transformed into the ciphertext by the bitwise XOR of the key stream s_n and the plaintext. The decryption is accomplished by applying the enciphering a second time.

4 Design Rationale

4.1 The Substitution

The substitution-box (S-box) in the proposed cipher deserves a few explanations. Firstly, the S-box performs one-to-one transformation on 16 bits. Thus the discretized CML function **g** can be bijective, which is desirable for the cipher.

Secondly, the value of parameter A of the S-box is carefully selected such that the corresponding continuous parameter of the original skew tent map, denoted by p, is close to 1/4. With this parameter, the discretized skew tent map have nice cryptographic properties [1].

Thirdly, we select the iterating times of the discretized skew tent map such that our S-box resists well against differential and linear cryptanalysis. The iterating times is denoted by N. For differential cryptanalysis, the maximum difference propagation probability is an important measure [3, 4]. Let S be an S-box with l-bit binary input vector $I = (I_1, I_2, \cdots, I_l)^t$ and l-bit binary output vector $O = (O_1, O_2, \cdots, O_l)^t$. Consider a pair of input vectors I and I^* with bitwise difference $I \oplus I^* = I^{'}$. Let $O = S(I)$, $O^* = S(I^*)$ and $O^{'} = O \oplus O^*$. The difference $I^{'}$ propagates to the difference $O^{'}$ through S. A difference propagation probability $P^S(I^{'}, O^{'})$ is defined as

$$P^S(I^{'}, O^{'}) = \frac{\#\{I \in \mathcal{I} | S(I) \oplus S(I \oplus I^{'}) = O^{'}\}}{2^l}. \tag{14}$$

Where $\#\{\Omega\}$ denotes the number of elements of the set Ω. $\mathcal{I}$ is the set of all possible input vectors and the number of its elements is 2^l. The maximum difference propagation probability is $P^S_{max} = \max\limits_{I^{'} \neq 0,\, O^{'}} \{P^S(I^{'}, O^{'})\}$.

In linear cryptanalysis, we study the following linear approximations of S-box:

$$\bigoplus_{i=1}^{l} a_i \cdot I_i = \bigoplus_{i=1}^{l} b_i \cdot O_i. \tag{15}$$

Where the symbol $\cdot$ denotes a bitwise AND operation. $a_i \in \{0, 1\}$ and $b_i \in \{0, 1\}$. In order to have a compact notation, we define binary vectors $a = (a_1, a_2, \cdots, a_l)^t$ and $b = (b_1, b_2, \cdots, b_l)^t$. Analogous to the inner product of vectors in linear algebra, we use the follow notation to express the linear equation (15):

$$a^T I = b^T O. \tag{16}$$

According to [5], the probability bias of this linear equation is defined as

$$\epsilon_{a,b} = |\frac{\#\{I \in \mathcal{I} | a^T I = b^T O\}}{2^n} - \frac{1}{2}|. \tag{17}$$

The maximum bias is $\epsilon_{max} = \max\limits_{a,b \neq 0} \{\epsilon_{a,b}\}$. The probability bias $\epsilon_{a,b}$ can be effectively computed by using Walsh-Hadamard Transform (WHT) and convolutions of the WHT spectra [6].

For various values of the iterating times N, we numerically compute P^S_{max} and ϵ_{max} for our S-box. When computing ϵ_{max}, due to the limited computing resource, we just consider the linear equations containing single output bit. The results are shown in Table 1. We find that P^S_{max} and ϵ_{max} decrease to steady values as the iterating times N increases. In our cipher we select $N = 20$. In this case, the maximum probability of difference propagation over our S-box

$P^S_{max} < 2^{-11}$ and the maximum linear probability bias $\epsilon_{max} < 2^{-6}$. Since we can implement our S-box in the form of a look-up table in software, the large iterating times of the discretized maps will not decrease the encryption speed of our cipher. The memory requirement of our S-box is $16 \times 2^{16} = 2^{20}$ bits, which is acceptable for general applications.

Table 1. P^S_{max} and ϵ_{max} for various values of the iterating times N

N	1	5	10	15	20	25	30	35
P^S_{max}	0.2960	0.0438	0.0068	0.0006	0.0003	0.0003	0.0003	0.0003
ϵ_{max}	0.2502	0.1765	0.0598	0.0213	0.0096	0.0096	0.0090	0.0096

4.2 The Mixing Transformation

In this section, we demonstrate that the mixing transformation of our cipher is one-to-one. The mixing transformation is represented by matrix D. For various values of L, the determinant of the matrix D, denoted by $det(D)$, can be computed. The results are shown in Table 2. We can find that $det(D)$ is nonzero and equal to 3 or -3 for some values of L. Based on number theory [7], since the nonzero $det(D)$ and the modulus $M = 2^{16}$ are relatively prime, there exists an inverse of D modulo M, i.e., the mixing transformation is bijective when $det(D)$ is nonzero. In the proposed cipher, we select $L = 8$ such that the mixing transformation is one-to-one. Since the map S is also bijective, the discretized CML function **g** in our cipher performs one-to-one transformation.

Table 2. The determinant of the matrix D for different L

L	4	5	6	7	8	9	10	11	12
$det(D)$	-3	3	0	3	-3	0	-3	3	0

4.3 The Counter System

Period length. In our cipher, we use a maximum-length LFSR of 128 as a counter to guarantee the large period length of the key streams. Since the state of the LFSR has a period length of $2^{128} - 1$, according to [8], the period length of the key streams is larger than $\sqrt{2^{128} - 1} \approx 2^{64}$. This period length is able to satisfy the requirements of general applications. For example, assume that the encryption speed our cipher is 1G bit/s, which is the maximum encryption speed we reach with our computer, then our stream cipher can be used to successively encrypt messages for several hundred years without changing the keys.

Bit-flip probabilities. In addition to a fixed large period length, the sequences generated by the LFSR have nice statistical properties. As a result, the bit-flip probabilities for all bit positions of our counter are close to 0.5 [9], which makes the counter bits very difficult to predict. In fact, in the early design of our cipher, we adopt the 128-bit counter that is incremented by one after each iteration.

This counter has a period length of 2^{128}. However, the bits of this counter is very predictable. For example, the least significant bit flips after each iteration and the most significant bit keeps unchanged for many iterations. We find that this weakness of the counter can be used to facilitate the attacks against our stream cipher [26]. Therefore, in this paper we select the LFSR to enhance the security of the cipher.

The removal of symmetry. Finally, the selected LFSR can eliminate the symmetry of the state-update function and the key setup thus avoid a class of weak keys. We give an example of the weak keys caused by the symmetry of the state-update function. Suppose the counter is omited. Consider a key with the relation $k_i = k_{8-i}\,(i = 1, 2, 3)$, then due to the symmetry, the state variable $x_{i,n}$ would preserve the relations $x_{i,n} = x_{8-i,n}\,(i = 1, 2, 3)$ for all the round n. In this case, according to the key stream generation scheme, the key streams always keep the relations $s_{3,n} = s_{1,n}$ and $s_{2,n} = 0$. Fortunately the states of the LFSR does not preserve this symmetry property and thus can avoid these weak keys. The initial states of the LFSR is just carefully selected for this purpose.

4.4 Key and IV Setup

The one-to-one correspondence between key/IV and stream. For the key and IV setup, we have the following two secure properties. First, different keys lead to different key streams. As the improved CML function $\mathbf{g}$ is one-to-one, different keys result in different counter values after key setup. Further, different counter values must necessarily lead to different key streams. This is easy to proof, so we omit it here. Second, the two key streams with the same key and different IVs will also be different. The same key result in the same counter values and the state variable values after key setup. Since IV bits is XORed to the state variable bits at the start of IV setup, the state variable values will be different for different IVs after IV setup. Function $\mathbf{g}$ is one-to-one, therefore different state variable values and the same counter value must necessarily lead to different key streams.

The diffusion property for key and IV setup. In the key and IV setup, we require that the influence of a single key and IV bit can be spread over all state variable bits. We can easily find that it requires four iterations of function $\mathbf{g}$ to spread the influence of a single key bit over all the eight state variables. That is to say, we need at least two iterations of the state-update function in key setup. Similarly, it requires at least one iteration of the state-update function to spread the influence of a single IV bit on all the state variables. In our cipher, a safety margin is provided by iterating the state-update function three times and two times for key setup and IV setup respectively. The diffusion of each bit of key and IV on the state variables can be examined by statistical tests. In section 6, we use four structural tests to analyze the key and IV setup of our cipher. The results verify the enough diffusion in our key and IV setup.

5 Security Analysis

5.1 The Branch Number of the Mixing Transformation

The mixing transformation in our cipher can be rewritten as follows:

$$y = Dx \; mod \; M. \tag{18}$$

For compactness, the subscript n which represents the round number has been omitted. This mixing transformation is nonlinear where the linearity here refers to a bit-wise XOR operation. It takes a 128-bit binary string X as the input and a 128-bit binary string Y as the output. Where X is obtained by concatenation of the eight state variables x_i as $X = x_7 \diamond x_6 \diamond \cdots \diamond x_0$. Similarly, $Y = y_7 \diamond y_6 \diamond \cdots \diamond y_0$. The 128-bit binary strings X and Y are also regarded as 128-bit vectors as $X = (X_1, X_2, \cdots, X_{128})$ and $Y = (Y_1, Y_2, \cdots, Y_{128})$ respectively in the following description.

The differential branch number. According to [10], the differential branch number of the mixing transformation is defined by

$$\begin{aligned} B_d(D) &= \min_{X,\, X^* \neq X} \{w(X \oplus X^*) + w(Y \oplus Y^*)\} \\ &= \min_{X,\, X' \neq 0} \{w(X^{'}) + w(Y^{'})\}. \end{aligned} \tag{19}$$

Where $X^{'} = X \oplus X^* = x_7^{'} \diamond x_6^{'} \diamond \cdots \diamond x_0^{'}$, $Y^{'} = Y \oplus Y^* = y_7^{'} \diamond y_6^{'} \diamond \cdots \diamond y_0^{'}$. $w(X^{'})$ represents the number of variables $x_i^{'}$ $(i = 0, 1, \cdots, 7)$ that is non-zero. For differential cryptanalysis of block cipher, the cryptanalyst examines the differential trails with a high probability. A differential trail is a sequence of input and output differences to the rounds so that the output difference from one round corresponds to the input difference for the next round. The S-boxes involved in a differential trail which have a non-zero input difference are called active S-boxes. Based on this definition, the number of active S-boxes of any two-round differential trail is lower bounded by the branch number. We have the following result for our mixing transformation.

Proposition 1. The differential branch number of the mixing transformation is 4.

Proof. We analyze the value of $w(X^{'}) + w(Y^{'})$ for different values of $w(X^{'})$. First, consider $w(X^{'}) = 1$, there is only one variable $x_i^{'}$ that is non-zero. Due to the symmetry of the mixing transformation, without loss of generality, we assume that $x_0^{'}$ is non-zero. Thus $x_0 \neq x_0^*$. The output variables of the mixing transformation are $y_0 = x_7 \boxplus x_0 \boxplus x_1$, $y_7 = x_6 \boxplus x_7 \boxplus x_0$, $y_j = x_{j-1} \boxplus x_j \boxplus x_{j+1}$, $j = 1, 2, \cdots, 6$. It is easy to verify $y_0^{'} \neq 0$, $y_7^{'} \neq 0$ and $y_1^{'} \neq 0$. We take the proof of $y_0^{'} \neq 0$ as an example. Assume $y_0^{'} = 0$, i.e., $y_0 = y_0^*$, then we have $x_7 \boxplus x_0 \boxplus x_1 = x_7^* \boxplus x_0^* \boxplus x_1^*$. Observe that $x_7 \boxplus x_1 = x_7^* \boxplus x_1^*$, then $x_0 = x_0^* \; mod \; M$. Since $x_0, x_0^* \in [0, M-1]$, we obtain $x_0 = x_0^*$ which leads to contradiction with the assumption that $x_0 \neq x_0^*$. This shows that $y_0^{'} \neq 0$. Similarly, we can proof $y_7^{'} \neq 0$ and $y_1^{'} \neq 0$. Thus in this case the value of $w(X^{'}) + w(Y^{'})$ is at least 4.

Next, suppose $w(X^{'}) = 2$. Due to the symmetry of the mixing transformation, we need to consider the following four cases: $1)x_0^{'}, x_1^{'} \neq 0, 2)x_0^{'}, x_2^{'} \neq 0, 3)x_0^{'}, x_3^{'} \neq 0, 4)x_0^{'}, x_4^{'} \neq 0$. With the similar analysis as for $w(X^{'}) = 1$, we have the following results for each case: $1)y_2^{'}, y_7^{'} \neq 0$, $2)y_2^{'}, y_7^{'}, y_3^{'}, y_0^{'} \neq 0$, $3)y_1^{'}, y_0^{'}, y_7^{'}, y_2^{'}, y_3^{'}, y_4^{'} \neq 0$, $4)y_1^{'}, y_0^{'}, y_7^{'}, y_3^{'}, y_4^{'}, y_5^{'} \neq 0$. Thus the value of $w(X^{'}) + w(Y^{'})$ is still at least 4.

Finally, suppose $w(X^{'}) \geq 3$. Since the mixing transformation is one-to-one, different input values will always lead to different output values. That is to say, there is at least one variable $y_i \neq y_i^*$ for $X \neq X^*$. Therefore we have $w(X^{'}) + w(Y^{'}) \geq 4$.

The linear branch number. Let $\Phi = (\Phi_1, \Phi_2, \cdots, \Phi_{8m})$, $\Psi = (\Psi_1, \Psi_2, \cdots, \Psi_{8m})$ be $8m$-bit vectors. We can divide the different bit positions of Φ into 8 sets denoted by $\phi_1, \phi_2, \cdots, \phi_8$, where $\phi_i = \{\Phi_{8(i-1)+1}, \cdots, \Phi_{8i}\}$. Similarly, we can define $\psi_i = \{\Psi_{8(i-1)+1}, \cdots, \Psi_{8i}\}$. For the mixing transformation in our stream cipher, the value of m is 16, Φ and Ψ are 128-bit vectors. Following the definition proposed in [10], the linear branch number of the mixing transformation is

$$B_l(D) = \min_{\Phi,\, \Psi,\, \epsilon_{\Phi,\Psi} \neq 0} \{w(\Phi) + w(\Psi)\}. \tag{20}$$

Where $w(\Phi)$ represents the number of sets ϕ_i $(i = 1, 2, \cdots, 8)$ that have at least one non-zero bit. $\epsilon_{\Phi,\Psi}$ is the probability bias of equation $\Phi^T X = \Psi^T Y$, which is the linear relationship between the input vector X and the output vector Y of the mixing transformation. $\epsilon_{\Phi,\Psi}$ can be defined as

$$\epsilon_{\Phi,\Psi} = |\frac{\#\{X \in \mathcal{X} | \{\Phi^T X = \Psi^T Y\}}{2^{8m}} - \frac{1}{2}|. \tag{21}$$

Where $\mathcal{X}$ is the set of all possible input vectors.

Linear cryptanalysis of block ciphers tries to exploit the linear trail with a high probability. A linear trail is specified by a series of linear approximations of the rounds such that the involving output bits in the approximation of one round are the same as the involving input bits in the approximation of the next round. The S-boxes involved in the linear trail are referred as active S-boxes. Then the number of active S-boxes of any two-round linear trail is lower bounded by the linear branch number. Clearly, we cannot compute $B_l(D)$ for the original mixing transformation, since $\epsilon_{\Phi,\Psi}$ is difficult to numerically computed for all the possible 128-bit vectors Φ and Ψ. However, we can investigate $B_l(D)$ of the reduced version of the mixing transformation where each state variable has been given in 2 or 3 bits, i.e., $m = 2$ or 3. The results are as follows.

Proposition 2. The linear branch number of the reduced version of the mixing transformation where $m = 2$ or 3 is 4.

Assume $m = 2$ or 3, then the input and output of the mixing transformation are reduced to 16-bit or 24-bit vectors. We compute the value of $w(\Phi) + w(\Psi)$ for different values of $w(\Psi)$. For compactness, we use the denotion $\psi_i \neq 0$ to express that the set ψ_i has non-zero bits. First, consider $w(\Psi) = 1$, there is only one set of bit positions of Ψ that have non-zero bits. Due to the symmetry of the mixing transformation, without loss of generality, we assume that $\psi_0 \neq 0$. We can obtain the vectors Φ such that $\epsilon_{\Phi,\Psi} \neq 0$ by Walsh-Hadamard Transform (WHT) and convolutions of the WHT spectra. Then we get the minimum value of $w(\Phi) + w(\Psi)$ as 4. Next, suppose $w(\Psi) = 2$. In this case we need to consider the following four cases: 1)$\psi_0, \psi_1 \neq 0$, 2)$\psi_0, \psi_2 \neq 0$, 3)$\psi_0, \psi_3 \neq 0$, 4)$\psi_0, \psi_4 \neq 0$. By computing $\epsilon_{\Phi,\Psi}$, we again obtain the minimum value of $w(\Phi) + w(\Psi)$ as 4. Finally, suppose $w(\Psi) \geq 3$. Since the mixing transformation is one-to-one, $\epsilon_{\Phi,\Psi}$ will be zero for the linear approximations involving only the output bits and no input bits. In other words, if $\epsilon_{\Phi,\Psi} \neq 0$, the linear approximations should involve at least one input bit. That is to say, $w(\Phi) \geq 1$. Thus we have $w(\Phi) + w(\Psi) \geq 4$.

With the above results, we can reasonably presume that the linear branch number of our original mixing transformation is 4.

5.2 Resistance of IV Setup against Differential and Linear Cryptanalysis

In modern stream ciphers, IV setup has been proven to be a crucial component [11-13]. In a typical attack scenario, we assume that an attacker can know or choose the IV values with a fixed unknown key and get the corresponding key stream sequences. The attacker aims to derive information upon the keys or distinguish the IV setup algorithm from a random function. This kind of attack can be preformed by exploring the high probability difference propagations or high correlations over the IV setup scheme. Below we demonstrate the resistance of our IV setup against these two attacks.

Our IV setup algorithm is very similar to a two-round key-alternating block cipher. In each round, the discretized CML function **g** is iterated two times. The IV bits are XORed to the state variable bits before the iteration in IV setup. Thus in the scenario that IV is known or can be chosen by the adversary, IV corresponds to the plaintexts of block cipher. The first 64-bit key streams are extracted from the state variable bits after the IV setup, these key streams correspond to the ciphertexts of block cipher. The counter bits are XORed to the state variable bits and behave as round keys. Following the analysis of key-alternating block cipher [10], we can demonstrate the resistance of our IV setup against differential and linear cryptanalysis.

For differential cryptanalysis, the number of active S-boxes of each round is lower bounded by the differential branch number of the mixing transformation $B_d(D) = 4$. The IV setup consists of two rounds, therefore the number of active S-boxes of IV setup is at least 8. Since the maximum difference propagation probability of the S-box is $P^S_{max} < 2^{-11}$, the maximum difference propagation probability over the IV setup is about $(P^S_{max})^8 < 2^{-88}$. Thus generally it requires about 2^{88} chosen IV pairs to mount the attack. It is impossible for our 64-bit IV.

For linear cryptanalysis, the number of active S-boxes of each round is lower bounded by the linear branch number of mixing transformation $B_l(D) = 4$. Then the number of active S-boxes of IV setup is also at least 8. The maximum linear probability bias of the S-box is $\epsilon_{max} < 2^{-6}$. According to piling-up lemma [5], the maximum probability bias of linear approximations of IV setup is about $2^7(\epsilon_{max})^8 < 2^{-41}$. Then generally it requires about 2^{82} known IVs to mount the attack. It is also impossible for our 64-bit IV.

5.3 Linear Correlations between Consecutive Key Stream Bits

Linear correlation is another major concern in the design of stream ciphers [14, 15]. In this section, we analyze the linear correlations between consecutive key stream bits of our cipher and show that this linear correlations are below the safe bounds.

In the following we ignore the counter system, i.e., we have $x_{i,n+1} = f_{i,n}$ for the state-update function. We try to find the linear correlations between bits in consecutive key stream bits s_n and s_{n+1}. Let Γ and Θ be 64-bit vectors. s_n and s_{n+1} are also regarded as 64-bit vectors. Then the linear equations involving bits in s_n and s_{n+1} are expressed as

$$\Gamma^T s_n = \Theta^T s_{n+1}. \tag{22}$$

We investigate the minimum number of the active S-boxes in these linear approximations. To get the equation of the form (22), the substitution layer S_2 of the state-update function should be first approximated (see Fig.2). As an example, consider that we aim to construct the linear approximations involving the bits in $s_{i,n+1}$. Then according to (13), we have

$$s_{i,n+1} = x_{i+4,n+1} \oplus x_{i,n+1} = f_{i+4,n+1} \oplus f_{i,n+1}. \tag{23}$$

We need to approximate the two S-boxes with the outputs $f_{i+4,n+1}$ and $f_{i,n+1}$ in the substitution layer S_2. The following linear approximations should be obtained:

$$\theta_i^T f_{i+4,n} = \eta^T h_{i+4,n}, \quad \theta_i^T f_{i,n} = \tau^T h_{i,n}. \tag{24}$$

Consequently we have

$$\theta_i^T s_{i,n+1} = (\eta^T h_{i+4,n}) \oplus (\tau^T h_{i,n}). \tag{25}$$

We can finally get the linear equations of form (22) by further finding the linear approximations between $h_{i,n}$ and $s_{i,n}$.

With the above analysis, we show that the S-boxes with the outputs $f_{i+4,n+1}$ and $f_{i,n+1}$ need to be active simultaneously. Thus when applying the definition (20) to compute the minimum number of active S-boxes at the input and the output of the mixing transformation D_2, Ψ needs to satisfy that $\psi_i = \psi_{i+4} = 0$ or $\psi_i, \psi_{i+4} \neq 0$ for $1 \leq i \leq 4$. As in the computation of the linear branch number, we consider the reduced version of the mixing transformation D_2 with $m = 2$ and 3. By numerical computation, we can get $B_l(D_2) = 8$. We reasonably presume $B_l(D_2) = 8$ for the original mixing transformation where $m = 16$. Thus it needs at least 2^{82} consecutive iterations of the state-update function to mount a successful distinguishing attack. Since the cipher will not be iterated more than 2^{64} times with the same key, this distinguishing attack is infeasible.

In fact, if the counter in the state-update function is considered, it will be more difficult to construct effective linear approximations between consecutive key stream bits. Therefore we believe that our stream cipher is secure against this kind of attack.

6 Statistical Tests

The key streams generated by the cipher should have good randomness properties so that the statistics of the plaintexts can be masked. The randomness property is usually analyzed by applying statistical tests. For our cipher, the key stream bits are tested by using the NIST Statistical Test Suite [16] and the Diehard Battery of Tests [17]. We did not find any obvious deviation from randomness in these tests.

Table 3. The results of four statistical tests for the key and IV setup

Test name	Key/State Correlation	IV/State Correlation	Frame Correlation	Key Diffusion	IV Diffusion
Average p-value	0.4377	0.4498	0.4691	0.5001	0.5427

In addition, we use statistical tests to measure the correlations between the key, IV and key streams. Four statistical tests are applied following the methods proposed in [18]. They are Key/State Correlation Test, IV/State Correlation Test, Frame Correlation Test and Diffusion Test. The details of these tests are refer to [18]. The test results for our cipher are shown in Table 3. For each test the average of 10 p-values is computed. We find that the average p-values are all close to 0.5. That is to say, there is no weakness found in these tests.

7 Performance

In this section we provide the performance of Gemstone. We compare the performance of Gemstone with some recent stream ciphers in two 32-bit processors: AMD-athlon 3200+, Intel Pentium 4 3.00GHz, using the eSTREAM testing framework [25]. The number of cycles consumed per byte are tested. The elementary tests are: the encryption rate for long streams by ciphering a long stream in chunks of about 4Kb; the packet encryption rate for three packet lengths (40, 576 and 1500 bytes) including an IV setup; the key setup and the IV setup. Table 4 and Table 5 sum up the results.

As shown in these tables, the cipher Gemstone is slightly slower than AES-CTR when encrypting long streams. However, the initialization speeds of Gemstone are higher than some finalists of eSTREAM, such as HC-128, CryptMT and NLS. Therefore Gemstone may have some advantages in the applications where many very small packets are encrypted.

Table 4. Number of CPU cycles for stream ciphers using an Intel Pentium 4 at 3.00GHz

Algorithm	Stream	40 bytes	576 bytes	1500 bytes	Key setup	IV setup
AES-128	17.84	33.34	18.65	18.14	406.79	202.07
AES-256	27.13	45.36	27.97	27.22	2027.35	183.06
HC-128	4.06	1671.91	119.51	47.43	115.21	67450.55
Py6	3.46	77.84	8.56	5.40	959.63	2012.87
NLS	15.47	125.75	42.76	17.77	1999.70	3061.88
CryptMT	15.40	2429.27	191.31	80.59	106.81	96626.29
DICING	23.20	908.75	71.72	44.41	86.15	24908.67
FUBUKI	144.22	3463.98	374.30	228.57	148.23	130556.80
Frogbit	819.65	1445.70	842.69	827.51	2520.40	30514.76
Gemstone	35.85	60.25	37.54	36.27	977.42	763.12

Table 5. Number of CPU cycles for stream ciphers using an AMD-athlon 3200+ at 2.01GHz

Algorithm	Stream	40 bytes	576 bytes	1500 bytes	Key setup	IV setup
AES-128	17.85	29.19	18.47	18.13	197.91	110.06
AES-256	40.84	59.23	41.35	41.15	1489.95	103.05
HC-128	3.99	2206.18	156.57	62.70	78.61	87891.40
Py6	4.24	60.49	8.18	5.84	917.77	1762.09
NLS	4.70	50.51	7.56	5.44	1068.89	995.97
CryptMT	9.30	680.97	60.14	27.75	76.51	26646.53
DICING	23.20	908.75	71.72	44.41	86.15	24908.67
FUBUKI	144.22	3463.98	374.30	228.57	148.23	130556.80
Frogbit	819.65	1445.70	842.69	827.51	2520.40	30514.76
Gemstone	30.55	48.10	31.83	60.89	684.72	594.88

8 Conclusion

In this paper, we presented a new stream cipher called Gemstone by discretizing CML. For the security of the cipher, we show that there is no high probability difference propagations or high correlations over the IV setup scheme. Additionally, the largest linear correlations between consecutive key streams are below the safe bounds. In terms of performance, Gemstone is slightly slower than AES-CTR, but its initialization speeds are higher than some finalists of eSTREAM.

References

1. Masuda, N., Jakimoski, G., Aihara, K., Kocarev, L.: Chaotic Block Ciphers: From Theory to Practical Algorithms. IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications 53(6), 1341–1352 (2006)
2. Schneier, B.: Applied Cryptography – Protocols, Algorithms, and Source Code in C, 2nd edn. John Wiley & Sons, Chichester (1996)

3. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991)
4. Heys, H.M.: A Tutorial on Linear and Differential analysis, http://www.engr.mun.ca/~howard/PAPERS/ldc_tutorial.ps
5. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
6. Daemen, J., Govaerts, R., Vandewalle, J.: Correlation matrices. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 275–285. Springer, Heidelberg (1995)
7. Rosen, K.H.: Elementary Number Theory and its Applications, 5th edn. Addison-Wesley, Reading (2005)
8. Shamir, A., Tsaban, B.: Guaranteeing the Diversity of Number Generators. Information and Computation 171(2), 350–363 (2001)
9. Menezes, A., Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press LLC, Boca Raton (1997)
10. Daemen, J., Rijmen, V.: The Wide Trail Design Strategy. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 222–238. Springer, Heidelberg (2001)
11. Joux, A., Muller, F.: A Chosen IV Attack Against Turing. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 194–207. Springer, Heidelberg (2004)
12. Berbain, C., Gilbert, H.: On the Security of IV Dependent Stream Ciphers. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 254–273. Springer, Heidelberg (2007)
13. Englund, H., Johansson, T., Turan, M.S.: A Framework for Chosen IV Statistical Analysis of Stream Ciphers. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 268–281. Springer, Heidelberg (2007)
14. Canniere, C.D., Preneel, B.: Trivium. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 244–266. Springer, Heidelberg (2008)
15. Boesgaard, M., Vesterager, M., Pedersen, T., et al.: Rabbit: A New High-Performance Stream Cipher. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 307–329. Springer, Heidelberg (2003)
16. Rukhin, A., Soto, J., Nechvatal, J., et al.: A statistical test suite for random and pseudorandom number generators for cryptographic applications (2001), http://www.nist.gov
17. Marsaglia, G.: DIEHARD Statistical Tests, http://stat.fsu.edu/geo/diehard.html
18. Turan, M.S., Doganaksoy, A., Calik, C.: Statistical Analysis of Synchronous Stream Ciphers. In: eSTREAM, ECRYPT Stream Cipher Project, Report 2006/012 (2006)
19. Wang, S., Kuang, J., Li, J., et al.: Chaos-based secure communications in a large community. Physical Review E 66(6), 065202(1-4) (2002)
20. Lu, H., Wang, S., Li, X., et al.: A new spatiotemporally chaotic cryptosystem and its security and performance analyses. Chaos 14(3), 617–629 (2004)
21. Li, P., Li, Z., Halang, W.A., Chen, G.: A stream cipher based on a spatiotemporal chaotic system. Chaos Solitons Fractals 32(5), 1867–1876 (2007)
22. Robshaw, M.: The eSTREAM Project. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 1–6. Springer, Heidelberg (2008)
23. Johansson, T.: Analysis and Design of Modern Stream Ciphers. In: Paterson, K.G. (ed.) Cryptography and Coding 2003. LNCS, vol. 2898, p. 66. Springer, Heidelberg (2003)

24. Scholl, E., Schuster, H.G.: Handbook of chaos control. Wiley-VCH, New York (2008)
25. Canniere, C.D.: eSTREAM Software Performance. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 119–139. Springer, Heidelberg (2008)
26. Yin, R., Yuan, J., Yang, Q., et al.: Linear cryptanalysis for a chaos-based stream cipher. In: International Conference on Communications, Information and Network Security 2009 (2009)

Appendix

Test Vectors of Gemstone

Let the 128-bit key $K = k_7 \diamond k_6 \diamond \cdots \diamond k_0$ and 64-bit initialization vector $IV = IV_3 \diamond IV_2 \diamond IV_1 \diamond IV_0$. The first 64 bits of keystream are given for different values of key and IV. They are presented byte-wise as follows.

1. The key and IV are set as 0.

   68E8 98E3 222B 49F2

2. The key is set as 0 and IV_i is set as 0 except that $IV_0 = 1$.

   B265 EA2F C7F5 306C

3. The IV is set as 0 and k_i is set as 0 except that $k_0 = 1$.

   3901 D9C8 30BA 7883

Proposition of Two Cipher Structures

Lei Zhang[1,2], Wenling Wu[1], and Liting Zhang[1,2]

[1] State Key Laboratory of Information Security,
Institute of Software, Chinese Academy of Sciences, Beijing 100190, P.R. China
{zhanglei1015,wwl,liting}@is.iscas.ac.cn
[2] State Key Laboratory of Information Security,
Graduate University of Chinese Academy of Sciences, Beijing 100049, P.R. China

Abstract. In this paper, we have proposed two block cipher structures which can be considered as variants of SP-network and Generalized Feistel structure respectively. Our main idea is to improve the diffusion effect when mixing all the sub-blocks together in each round. We also show that compared with the original structures, our structures have several important advantages. Then we evaluate the security of our structures against main attacks by estimating the upper bounds for differential and linear probabilities, and also the maximum number of rounds for impossible differential. In the end, we present two example ciphers which are based on the structures proposed, and we also adopt several novel and state-of-the-art design techniques. Then by explaining the design rationales and evaluating the security of the example ciphers under main attack settings, we can conclude that both of our ciphers can achieve enough immunity against known attacks and also have high performances.

Keywords: Block Cipher, cipher structure, differential probability, linear probability, provable security, impossible differential characteristic.

1 Introduction

The overall structure of a block cipher is one of the most important properties, and it will play important roles in both security aspects and implementation performances of the cipher. At present, the most often used structures include Feistel structure[1], SP-network[2] and Generalized Feistel structure[3] etc. As is adopted by the most well known block cipher AES, the SP-network is very famous. The main advantages of SP structure include that it is very simple and clear, and it can achieve full diffusion very quickly. However, usually its decryption and encryption process can not be similar, and this will increase additional costs in software and hardware implementations. Especially in a resource restricted environment this will be a disadvantage which can not be ignored. On the other hand, Feistel structure and Generalized Feistel structure both have similar advantages such as decryption-encryption similarity and the inverse of round function is not necessary in decryption. This can make the design of round function more simple and flexible. However, since only part of a block goes through the round function in each round, it needs more rounds to achieve enough diffusion. Furthermore, there always exists long rounds of impossible differential in Feistel structure (5-round ID)[4] and 4-branch Generalized Feistel structure (9-round ID)[5] which makes them vulnerable to impossible differential attack.

Furthermore, new design and cryptanalysis techniques are evolved day by day. For example, the newly proposed block cipher CLEFIA[5] had been attacked up to 12 out of the 18 rounds[6], which mainly uses a 9-round impossible differential existed for the Generalized Feistel structure. Moreover, only recently the first attack on full AES-256 was announced by Biryukov-Khovratovich-Nikolic[7]. They exploited slow diffusion and other differential weaknesses in the key schedule of AES-256 and presented the first related-key attacks on the full 14-round AES-256, but it works only for a weak key class. Later, Biryukov-Khovratovich[8] improved these results and presented the first related-key attack on full AES-256 that works for all the key space and also developed the first related-key attack on full AES-192 by using boomerang attack. All these cryptanalytic results show that now it is good timing to enhance the structure design and increase security strength of both round function and key scheduling of the block cipher.

In this paper, we propose two block cipher structures which are variants of SP-network and Generalized Feistel structures respectively. Our main idea is to use permutation with good branch number to improve the diffusion effect when mixing all the sub-blocks together in each round. Our first structure is a variant of SP-network, which employs the combination of SDS transformation as round function to achieve best diffusion in each sub-block, and then employs a permutation with good branch number to mix all the sub-blocks together. Note here we can employ a simple 4×4 binary matrix as permutation to obtain enough number of active columns. Therefore, this structure can be more efficient in the sense that it can use less MDS transforms while keeping the number of active S-boxes unchanged. Furthermore, since the round function contains two layers of S-boxes, if we choose appropriate S-boxes and MDS matrix, we can make this structure be involution which means its decryption and encryption can be similar. On the other hand, our second structure can be considered as a variant of Generalized Feistel structure. It uses a 4×4 binary matrix transformation instead of the ordinary switch transformation in each round. By using this kind of complicated transformation, we can not only improve the diffusion effect but also reduce the maximum number of rounds for impossible differential existed.

Then we evaluate the security of these two structures against several main attacks respectively. By estimating the upper bounds for differential and linear hull probabilities, and the maximum number of rounds for impossible differential, we can prove their security against these attacks. In the end, we present two example ciphers based on the structures proposed, and we also adopt several novel and state-of-the-art design techniques. Then we evaluate the security of these two example ciphers against several most often used attacks and their implementation performances briefly.

This paper is organized as follows. Sect. 2 gives some useful definitions and notations. Then Sect. 3 describes our first proposed structure, and also explains its design rationales and security evaluations briefly. Sect. 4 describes our second proposed structure and its design rationales and security evaluations respectively. Sect. 5 and Sect. 6 present the two example ciphers designed and evaluations of their security against main attacks and implementation performances etc. Finally, Sect. 7 concludes the paper.

2 Preliminary

In this section, we give some definitions and notations which are useful for later analysis. If the input difference (linear mask) of a sbox in differential characteristic (linear approximation) is nonzero, then it is called a differential (linear) active sbox. For the concept of branch number with respect to a bundle partition, we can follow the definitions of differential and linear branch number defined in [2].

Definition 1. *The differential branch number of a linear transformation ϕ is given by*

$$B_d(\phi) = \min_{\alpha \neq 0}\{w_b(\alpha) + w_b(\phi(\alpha))\},$$

where $w_b(\cdot)$ represents the number of bundles with nonzero difference.

Definition 2. *The linear branch number of a linear transformation $\lambda(x) = M \cdot x$ is given by*

$$B_l(\lambda) = \min_{\beta \neq 0}\{w_b(\beta) + w_b(M^T \cdot \beta\},$$

where $w_b(\cdot)$ represents the number of bundles with nonzero linear mask.

Obviously, an upper bound for the differential and linear branch number of a linear transformation is $m + 1$, where m denotes the total number of bundles in the state. Notice that if a linear transformation has the maximal possible differential or linear branch number $m+1$, then both branch numbers are equal. Furthermore, if a linear transformation satisfies the condition of being symmetric, namely $M^T = M$, then both branch numbers are equal too.

For the 4×4 binary matrix which will be used to combine all the sub-blocks together in our structures, its possible differential branch number is at most 4. Hence we try to choose symmetric matrix, so that its differential and linear branch number are equal. Furthermore, to make the structures be involution the binary matrix should be involution, too. After searching all the 4×4 binary matrices, there are 10 matrices satisfy all the above conditions: symmetric, involution, and maximal possible branch number. We simply choose the first one as permutation P used in the first structure, and combine the switch transformation with it to obtain permutation Q which will be used in the second structure.

$$P = \begin{pmatrix} 0\,1\,1\,1 \\ 1\,0\,1\,1 \\ 1\,1\,0\,1 \\ 1\,1\,1\,0 \end{pmatrix} \qquad Q = \begin{pmatrix} 1\,0\,1\,1 \\ 0\,1\,1\,1 \\ 1\,1\,1\,0 \\ 1\,1\,0\,1 \end{pmatrix}$$

3 Description of Structure 1

In this section, we describe our first proposed structure, which can be called Variant-SP structure. Fig. 1 illustrates one round of this structure in detail. In each round, the input block is split into 4 sub-blocks of equal size, and the input state can be denoted as $X = (X_0, X_1, X_2, X_3)$. Then round function F is applied to each sub-block in parallel, and we denote the intermediate state as $Z = (Z_0, Z_1, Z_2, Z_3)$. Finally, the 4×4 binary matrix P described in Sect. 2 is used to mix all the sub-blocks together and the output state is denoted as $Y = (Y_0, Y_1, Y_2, Y_3)$. Therefore, one round of this structure can be expressed as:

$$\begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = P \cdot \begin{pmatrix} F(X_0) \\ F(X_1) \\ F(X_2) \\ F(X_3) \end{pmatrix} = \begin{pmatrix} 0\,1\,1\,1 \\ 1\,0\,1\,1 \\ 1\,1\,0\,1 \\ 1\,1\,1\,0 \end{pmatrix} \cdot \begin{pmatrix} Z_0 \\ Z_1 \\ Z_2 \\ Z_3 \end{pmatrix}$$

The round function F is defined as $F = S^{-1} \circ D \circ S$, which contains two layers of Sbox transformation and one layer of MDS matrix multiplication transformation. Here we use sbox S_0 in the first layer and its inverse sbox S_0^{-1} in the second layer. Furthermore, an involution MDS matrix is used in the middle to make the decryption process of our structure be similar with encryption. Round subkeys $K_0 = (K_{0,0}, K_{0,1}, K_{0,2}, K_{0,3})$ and $K_1 = (K_{1,0}, K_{1,1}, K_{1,2}, K_{1,3})$, will be bitwise XORed to the internal state before each layer of S-box transformation respectively. Therefore, round function F can be expressed as follows.

$$F\,(X_i): \quad Z_i = S^{-1}(\ MDS\ (\ S(X_i \oplus K_{0,i})\ ) \oplus K_{1,i}), \qquad (0 \le i \le 3).$$

3.1 Design Rationale

In the design of this structure, our main idea is to achieve high level of security while using as few operations as possible. Therefore, we first apply the combination of SDS transformations as round function to achieve best diffusion within each sub-block. Then we employ a simple permutation P to mix all the sub-blocks together whose branch number can guarantee the least number of active sub-blocks for every two rounds. Compared with the original SP structure, our structure can achieve the same number of active sboxes while using less MDS transforms. This can make our structure be more efficient in resource restricted implementation environments, especially where Sbox and MDS transformations can not be combined into a big table. Moreover, even in the big table implementation, to combine the output value of big tables together still needs more XOR operations than the permutation P which needs only 7 XOR operations.

This structure can be used to construct involution ciphers. If we omit transformation P in the last round and employ post-whitening key addition in the end, we can prove that the decryption and encryption frameworks can be similar. Considering r-round encryption, the plaintext and ciphertext are denoted as P and C respectively. Then the decryption process can be expressed as follows.

$$P = \underbrace{S^{-1} \circ MDS^{-1} \circ S \circ P^{-1}}_{r-1} \circ S^{-1} \circ MDS^{-1} \circ S(C \oplus WK)$$

Note in our structure the binary matrix P and MDS matrix are both involution, namely $MDS^{-1} = MDS$ and $P^{-1} = P$. Therefore, the decryption process is exactly the same with encryption process, except that additional transforms are needed to compute decryption subkey from encryption subkey.

Furthermore, we can prove that this structure can reach pseudorandomness after 3 rounds and super pseudorandomness after 6 rounds respectively, based on the assumption that their underlying round function F is pseudorandom.

3.2 Security Evaluation

In this section, we evaluate the security of this structure against differential cryptanalysis[9] and linear cryptanalysis[10] by estimating the upper bounds for differential and linear hull probabilities respectively.

Differential Probability. According to the definitions of active sbox and differential branch number in Sect. 2, we can get the following proposition about the upper bound for differential probability of this structure.

Proposition 1. *Suppose the differential branch numbers of MDS matrix and permutation P used in this structure are B_D and B_P respectively. Then for any two rounds, there are at least $B_D \times B_P$ differential active Sboxes. If the best differential probability of Sbox used in this structure is p, then the upper bound for differential probability of any two rounds is $p^{\,B_D \cdot B_P}$.*

Proof. According to the assumption, the permutation P guarantees that there are at least B_P active sub-blocks for any two rounds. Moreover, for each active sub-block, the combination of SDS transformations make sure that there are at least B_D active Sboxes. Therefore, for any two rounds of this structure, there are at least $B_P \times B_D$ differential active Sboxes. Furthermore, if the best differential probability of the Sbox used is p, then the best differential probability of any two rounds of this structure is less than $p^{\,B_D \cdot B_P}$. □

Hence, after we have chosen appropriate Sbox and MDS matrix, based on this proposition we can easily conclude how many rounds of this structure is provably secure against differential cryptanalysis since there exists no differential characteristic whose probability is high enough to mount the key recovery attack.

Linear Hull Probability. Similar to the analysis of differential probability, we can get the following proposition about the upper bound for linear hull probability of this structure.

Proposition 2. *Suppose the linear branch numbers of MDS matrix and permutation P used in our structure are L_D and L_P respectively. Then for any two rounds of this structure, there are at least $L_D \times L_P$ linear active Sboxes. If the best linear bias of Sbox used is q, then the bias for best linear approximation of any two rounds is upper bounded by $2^{(L_D \cdot L_P - 1)} q^{\,L_D \cdot L_P}$.*

Proof of this proposition is similar with Propostion 1, and we will omit the detail here for simplicity. Note that for the MDS matrix used in round function, it achieves maximal possible branch number and so that both its linear and differential branch numbers are equal, namely $L_D = B_D$. Moreover, for the binary matrix P used, since it is symmetric, then its differential and linear branch numbers are equal too, namely $L_P = B_P$.

After we have chosen appropriate Sbox and MDS matrix, based on Proposition 2 we can easily conclude how many rounds of this structure is provably secure against linear cryptanalysis since there exists no linear approximation whose bias is high enough to mount the key recovery attack.

4 Description of Structure 2

In this section, we describe our second proposed structure, which is a variant of Generalized Feistel structure. Our main idea is to use a complex permutation instead of the simple switch transformation usually used in Feistel-Type structures. By choosing a permutation with good branch number, we can not only improve diffusion effect but also enhance the immunity against impossible differential attack. The following Fig. 2 illustrates one round of this structure.

Suppose all the internal states are split into 4 sub-blocks of equal size, and then the input and output block of each round can be denoted as $X = (X_0, X_1, X_2, X_3)$ and $Y = (Y_0, Y_1, Y_2, Y_3)$ respectively. In each round, we apply the same round function F to the left part for every pair of sub-blocks. The intermediate state after this transformation is expressed as follows.

$$(Z_0,\ Z_1,\ Z_2,\ Z_3) = (X_0,\ F(X_0) \oplus X_1,\ X_2,\ F(X_2) \oplus X_3)$$

Then we use the binary matrix Q described in Sect. 2 to mix all the sub-blocks together. Therefore, one round of this structure can be expressed as follows.

$$\begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = Q \cdot \begin{pmatrix} Z_0 \\ Z_1 \\ Z_2 \\ Z_3 \end{pmatrix} = \begin{pmatrix} 1\,0\,1\,1 \\ 0\,1\,1\,1 \\ 1\,1\,1\,0 \\ 1\,1\,0\,1 \end{pmatrix} \cdot \begin{pmatrix} Z_0 \\ Z_1 \\ Z_2 \\ Z_3 \end{pmatrix}$$

For round function F, we also use the combination of $F = S \circ D \circ S$ to enhance the security level. Moreover, considering that this structure is involution itself since Q is involution, we can use the same sbox S_0 in both layers of Sbox transformation and a low hamming weight MDS matrix in the middle layer. The subkeys of each round are denoted as $K = (K_0, K_1, K_2, K_3)$, which will be bitwise XORed to the internal state before each layer of Sbox transformation. Therefore, round function F of this structure can be expressed as follows.

$$F\,(X_i) = S(\ MDS\ (\ S(X_i \oplus K_i)\ ) \oplus K_{i+1}), \qquad (i = 0, 2).$$

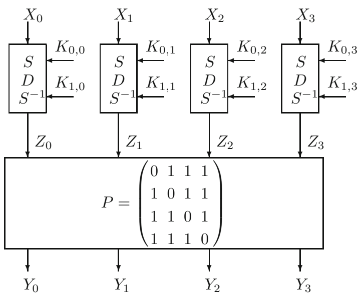


Fig. 1. One round of structure 1

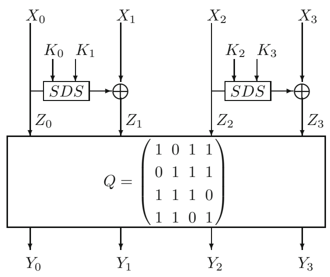


Fig. 2. One round of structure 2

4.1 Design Rationale

In the design of this structure, our main idea is to use a transformation with good branch number to replace the switch transformation usually used in Feistel-Type structures. The main advantage of this design is that we can achieve better immunity against impossible differential attack. In next section we will show that in this structure the best impossible differential is only 6 rounds, while traditional generalized Feistel structures usually has a 9-round impossible differential.

Furthermore, we carefully choose binary matrix Q which satisfies the conditions that it is symmetric, involution and with possible maximum branch number. Hence if transformation Q is omitted in the last round, then this structure is obviously involution. Moreover, since Q achieves the maximum branch number possible for 4×4 binary matrix, its good diffusion effect can make the number of active sub-blocks as high as possible. We also apply combination of SDS as round function to achieve best diffusion within sub-block, and the choice of MDS matrix with low hamming weight can reduce cost in hardware implementation.

Furthermore, we can prove that this structure can reach pseudorandomness after 3 rounds and super pseudorandomness after 6 rounds respectively, based on the assumption that their underlying F function is pseudorandom.

4.2 Security Evaluation

In this section, we first evaluate the security of this structure against differential and linear cryptanalysis by estimating the upper bounds for differential and linear hull probabilities. Then we evaluate its immunity against impossible differential attack by searching the best impossible differential existed.

Differential Probability. We can evaluate the best differential probability by estimating the least number of active Sboxes of this structure. Therefore, we can search for the least number of differential active sub-blocks first, and then the least number of active Sboxes can be computed easily since round function can guarantee the number of active Sboxes in each active sub-block.

First of all, we represent the internal state of this structure by a 4-bit value, in which the i-th bit represent the i-th sub-block. If a sub-block has nonzero difference then the corresponding bit is set as 1, and otherwise it is set as 0. Considering that only two sub-blocks will pass through the round function, hence only if these two sub-blocks have nonzero differences they will be called active sub-blocks. Then we can search for the least number of active sub-blocks by computer simulation, and the conclusions are as follows.

Proposition 3. *In this structure the least number of differential active sub-blocks for one round is 0, for two rounds is 1, for three rounds is 2, for four rounds is 2, for five rounds is 3 and for six rounds is 4.*

Proof. The number of least active sub-blocks for one round and two rounds are trivial. In the following, we first prove the case of three rounds.

First of all, we show there can be only 2 active sub-blocks for three rounds. The following 3-round differential characteristic just has 2 active sub-blocks:

$$(\alpha,\ 0,\ 0,\ \alpha) \xrightarrow{p} (0,\ 0,\ 0,\ \alpha) \rightarrow (\alpha,\ \alpha,\ 0,\ \alpha) \xrightarrow{p} (0,\ \alpha,\ \alpha,\ 0),$$

where p means the probability of the input and output difference of round function F are both equal to α.

In the following, we show that there can not be only 1 active sub-block for three rounds. If the active sub-block is in the first or third round, the situations are exactly the same. Hence we can assume the active sub-block in first round for simplicity. Then input difference of the second round must be the following form $(0, \alpha, 0, \beta)$. Hence input difference of the third round should be $(\beta, \alpha\oplus\beta, \alpha, \alpha\oplus\beta)$, which contradicts the assumption that there is no active sub-block in third round, since $\beta = \alpha = 0$ is impossible.

If the active sub-block is in the second round, then input differences of the first and third round should both be the following form $(0, \alpha, 0, \beta)$. Assume input difference of the first round is $(0, \alpha_1, 0, \beta_1)$, then input difference of the second round is $(\beta_1, \alpha_1 \oplus \beta_1, \alpha_1, \alpha_1 \oplus \beta_1)$. Considering that there should be only one active sub-block in the second round, we can assume $\beta_1 \neq 0$ and $\alpha_1 = 0$. On the other hand, suppose input difference of the third round is $(0, \alpha_2, 0, \beta_2)$, then the output difference after round function F of second round should be $(\beta_2, \alpha_2 \oplus \beta_2, \alpha_2, \alpha_2 \oplus \beta_2)$. According to round function F, it must satisfy that $\beta_2 = \beta_1 \neq 0$, $\alpha_2 = \alpha_1 = 0$ and $\beta_1 \oplus F(\beta_1) = \beta_1$. Obviously the third expression is impossible since $\beta_1 \neq 0$. Hence, there can not be only 1 active sub-block.

Therefore, there are at least 2 active sub-blocks for three rounds of this structure. Furthermore, the least number of active sub-blocks for four rounds etc. can be proved in a similar way. □

Suppose the differential branch numbers of MDS matrix is B_D. Then for each active sub-block, the combination of SDS transformations make sure that there are at least B_D active Sboxes. If the best differential probability of Sbox is p, then the best differential probabilities for three rounds, four rounds, five rounds and six rounds are upper bounded by p^{2B_D}, p^{2B_D}, p^{3B_D} and p^{4B_D} respectively.

Linear Hull Probability. Similar to the analysis of differential probability, we can get the following proposition about the linear active sub-blocks of this structure. Considering that the binary matrix Q is symmetric and involution, hence the least number of linear active sub-blocks obtained by searching algorithm will be the same as differential active sub-blocks.

Proposition 4. *For this structure, the least number of linear active sub-blocks for one round is 0, for two rounds is 1, for three rounds is 2, for four rounds is 2, for five rounds is 3 and for six rounds is 4.*

Proof of this proposition is similar with that of Proposition 4, and we will omit the detail here. Suppose the linear branch numbers of MDS matrix is L_D, and then for each active sub-block, the combination of SDS transformations make sure that there are at least L_D active Sboxes. If the best linear bias of Sbox is q, then the biases of best linear approximation for three rounds, four rounds, five rounds and six rounds are upper bounded by $2^{2L_D-1}q^{2L_D}$, $2^{2L_D-1}q^{2L_D}$, $2^{3L_D-1}q^{3L_D}$ and $2^{4L_D-1}q^{4L_D}$ respectively.

Impossible Differential Characteristics. Impossible differential cryptanalysis[11] is one of the most often used cryptanalytic techniques, and it is especially powerful for the Feistel-Type structures. For example, there always exists a 5-round impossible differential for Feistel structure[4], even though 3-round Feistel structure is provably secure against differential and linear cryptanalysis[12]. Furthermore, for 4-branch Generalized Feistel structure, there usually also exists a 9-round impossible differential[5] which makes it vulnerable to impossible differential attack.

Although our structure is also a variant of Generalized Feistel structure, we have used complex permutation to enhance the diffusion effect, and hence we can achieve better immunity against impossible differential attack. After searching for the maximum number of rounds for impossible differential existed in the structure, we show that the following 6-round distinguisher is the best existed.

Proposition 5. *The maximum number of rounds for impossible differential existed in this structure is 6 rounds, and the impossible differential characteristics can be expressed as follows.*

$$(0,0,0,\alpha) \not\rightarrow_6 (\delta,\delta,0,\delta), \qquad \alpha \neq 0, \quad \delta \neq 0,$$

and,

$$(0,\alpha,0,0) \not\rightarrow_6 (0,\delta,\delta,\delta) \qquad \alpha \neq 0, \quad \delta \neq 0.$$

In detail, we can express the input and output differences of each round as follows. Note here we just describe the first characteristic as an example.

$$(0,\ 0,\ 0,\ \alpha) \rightarrow (\alpha,\ \alpha,\ 0,\ \alpha) \rightarrow (0,\ \beta,\ \beta,\ \alpha \oplus \beta) \rightarrow (*,\ *,\ 0,\ *)\ \neq\ (0,\ \gamma,\ \gamma,\ *) \leftarrow (\delta,\ \delta \oplus \gamma,\ 0,\ \delta) \leftarrow (0,\ 0,\ 0,\ \delta) \leftarrow (\delta,\ \delta,\ 0,\ \delta).$$

where α and δ denote nonzero fixed differences, β and γ denote nonzero nonfixed differences, and $*$ denotes arbitrary difference respectively.

5 Application 1: Block Cipher VSP1

As an application, we design a new 256-bit block cipher. This cipher is called VSP1 and it is based on the first structure we proposed. The following Fig. 3 illustrates the encryption procedure of this cipher in detail.

First of all, 256-bit plaintext is split into 4 sub-blocks, and each sub-block is a 64-bit word. Then the same round function is applied except the last round, and the suggested number of rounds is 8. In the last round, we will omit the final permutation P and add a layer of post-whitening key addition in the end, so that the decryption and encryption processes can be similar.

In each round, first round function F is applied to each sub-block in parallel, and then the binary matrix P described in Sect. 2 is used to mix all the sub-blocks together. The round function F is defined as follows.

$$F\,(X_{r-1,i}) = S^{-1}(\ MDS_1\ (\ S(X_{r-1,i} \oplus K_{r,0,i})\ ) \oplus K_{r,1,i}), \qquad (0 \le i \le 3).$$

Here, two layers of Sbox transformation both contain eight 8×8 Sboxes. The first layer employs the sbox of AES, and the second layer employs its inverse sbox. The diffusion layer contains an involution 8×8 MDS matrix multiplication. Here we can employ the following involution MDS matrix of block cipher KHAZAD.

$$MDS_1 = \begin{bmatrix} 01 & 03 & 04 & 05 & 06 & 08 & 0B & 07 \\ 03 & 01 & 05 & 04 & 08 & 06 & 07 & 0B \\ 04 & 05 & 01 & 03 & 0B & 07 & 06 & 08 \\ 05 & 04 & 03 & 01 & 07 & 0B & 08 & 06 \\ 06 & 08 & 0B & 07 & 01 & 03 & 04 & 05 \\ 08 & 06 & 07 & 0B & 03 & 01 & 05 & 04 \\ 0B & 07 & 06 & 08 & 04 & 05 & 01 & 03 \\ 07 & 0B & 08 & 06 & 05 & 04 & 03 & 01 \end{bmatrix}$$

Furthermore, in each round there needs eight 64-bit subkeys and in the whitening layer there needs four additional whitening keys. Therefore, our cipher needs 68 subkeys in all, and each subkey is a 64-bit word. Hence it is important for the key schedule algorithm to generate subkeys fast and securely, and also supports the involution property of the ciphers without endangering the security by being overly simplistic. Therefore, in this paper, we will leave the key schedule algorithm for future work.

Note in this cipher, we set all the internal states as 64-bit words, and many of the operations are 64-bit word oriented. This is based on the considerations that recently the 64-bit platform is getting widely use gradually, and there are also requirements of large internal states since block cipher is often used as a building block for hash function, MAC and stream cipher constructions.

5.1 Implementation Considerations

For implementation on 64-bit platform, each round of VSP1 will need 64 table lookups, 43 XOR operations and 16KB memory to store the big tables. For

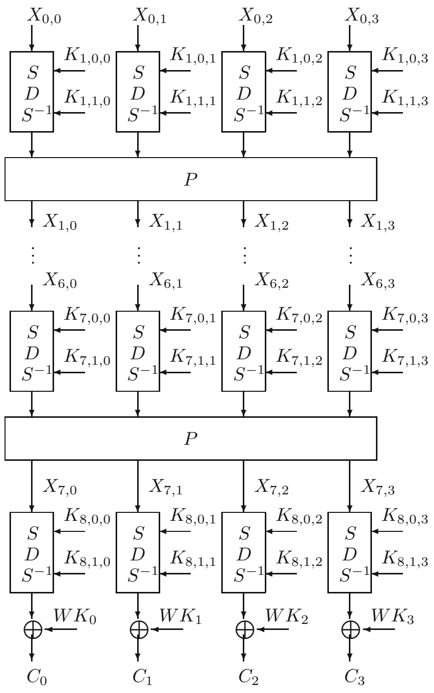


Fig. 3. Encryption of VSP1

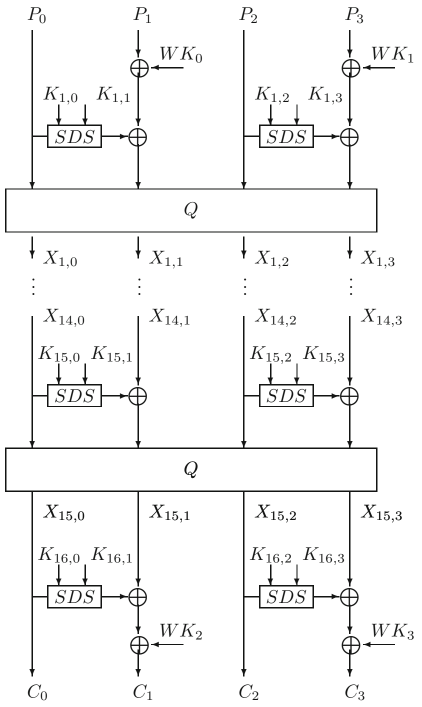


Fig. 4. Encryption of VGF2

implementation on 32-bit platform, the 8×8 MDS matrix can be split into four 4×4 small MDS matrix which can also be combined with the Sbox transform into a table. Therefore, in this case each round will need 64 table lookups, 86 XOR operations and 8KB memory to store the tables.

Compared with the 256-bit cipher Rijndael-256 which employs SP structure, its implementation on 32-bit platform needs 32 table lookups, 32 XOR operations and 8KB memory to store the tables each round. According to wide trail strategy in [2], the lower bound of active Sboxes in four rounds of Rijndael-256 is 25. While for the cipher VSP1, according to Proposition 1, there are at least 36 active Sboxes for two rounds. Thus it can be seen that this cipher is more efficient since to achieve the same level of security it will use less operations.

5.2 Security Evaluation

In this section, we will evaluate the security of VSP1 cipher against several most often used cryptanalytic techniques briefly.

Differential Cryptanalysis. For the 8×8 MDS matrix used in round function, its differential branch number is $B_D = 9$. According to the analysis in Sect. 2, the differential branch number of P is $B_P = 4$. Furthermore, for the sbox of

AES and its inverse sbox, the best differential probabilities are both $p = 2^{-6}$. Therefore, according to Proposition 1 in Sect. 3.2, for two rounds of this cipher, there are at least 36 active sboxes, and the best differential probability is less than $2^{-36*6} = 2^{-216}$. Moreover, for the full 8 rounds of this cipher, there are at least 144 active sboxes, and the best differential probability is less than $2^{-144*6} = 2^{-864}$. Therefore, full rounds of VSP1 is secure against differential cryptanalysis since there exists no differential whose probability is high enough to mount the key recovery attack.

Linear Cryptanalysis. For the 8×8 MDS matrix and binary matrix P, their linear and differential branch numbers are both equal, and the best linear bias for the sbox of AES and its inverse sbox are both $q = 2^{-4}$. Therefore, according to Proposition 2 in Sect. 3.2, for two rounds of this cipher, there are at least 36 active sboxes, and the best linear bias is less than $2^{35} \cdot 2^{-36*4} = 2^{-109}$. Moreover, for the full 8 rounds of this cipher, there are at least 144 active sboxes, and the best linear bias is less than $2^{143} \cdot 2^{-144*4} = 2^{-433}$. Therefore, the full rounds of VSP1 is practically secure against linear cryptanalysis.

Impossible Differential Cryptanalysis. For block cipher VSP1, the best impossible differential existed is the following 3-round impossible differential,

$$(0,0,0,\alpha_1) \rightarrow (\beta,\beta,\beta,0) \neq (0,*,*,*) \leftarrow (\gamma,0,0,0) \leftarrow (0,\alpha_2,\alpha_2,\alpha_2)$$

where α_1 and α_2 are nonzero fixed differences, β and γ are nonzero nonfixed differences, and $*$ is an arbitrary difference. Considering that in each round there are 512-bit subkey, hence it is unlikely that key recovery attack can work for full rounds of VSP1 using this 3-round impossible differential.

Integral Attack. For block cipher VSP1, the best integral characteristic existed is a 2-round integral distinguisher. If there is only one active byte in the first sub-block, then after F transformation in Round 2, the first sub-block is still passive, and the following P transformation will destroy this property. We can see that full rounds of VSP1 is secure against integral attack since it is unlikely that key recovery attack can work using this kind of 2-round integral characteristic.

6 Application 2: Block Cipher VGF2

Based on the second structure proposed, we design the following example cipher which is called VGF2. Its block size is also 256-bit, and all the internal states are represented as 64-bit words. Fig. 4 illustrates its encryption procedure in detail.

First of all, the second and fourth plaintext sub-blocks are XORed with the pre-whitening subkeys. Then the same round function is applied iteratively except the last round, and the suggested number of rounds is 16. In the last round, the final transformation Q is omitted, and two post-whitening subkeys are XORed to the second and fourth sub-blocks respectively to get the ciphertext.

In each round, we first apply the following round function F to the left part for every pair of sub-blocks. Then the binary matrix Q described in Sect. 2 is used to mix all the sub-blocks together. Round function F is defined as follows.

$$F\,(X_{r-1,i}) = S(\ MDS_2\ (\ S(X_{r-1,i} \oplus K_{r,i})\ ) \oplus K_{r,i+1}), \qquad i = 0,\ 2.$$

Here, two layers of Sbox transformation both contain eight 8×8 Sboxes. We can employ the same sbox of AES in both layers. Since there is no restriction for the MDS matrix used in diffusion layer, we can employ the following MDS matrix of cipher Grindahl which has low hamming weight.

$$MDS_2 = \begin{bmatrix} 01\ 04\ 01\ 01\ 02\ 0c\ 06\ 08 \\ 08\ 01\ 04\ 01\ 01\ 02\ 0c\ 06 \\ 06\ 08\ 01\ 04\ 01\ 01\ 02\ 0c \\ 0c\ 06\ 08\ 01\ 04\ 01\ 01\ 02 \\ 02\ 0c\ 06\ 08\ 01\ 04\ 01\ 01 \\ 01\ 02\ 0c\ 06\ 08\ 01\ 04\ 01 \\ 01\ 01\ 02\ 0c\ 06\ 08\ 01\ 04 \\ 04\ 01\ 01\ 02\ 0c\ 06\ 08\ 01 \end{bmatrix}$$

This cipher needs 4×16 round subkeys and 4 whitening subkeys in all, and each subkey is 64-bit. Here we also leave key schedule algorithm for future work.

6.1 Implementation Considerations

The encryption of VGF2 uses 32 Sbox calls, 4 subkey additions, 2 MDS matrix transforms and 1 binary matrix transform in each round. Therefore, when it is implemented on 64-bit platform and the operations of Sbox and MDS transforms are combined into a big table, then the operations of each round include 16 big table lookups, 14 XORs for the combination of table values, 16 small table lookups, 4 XORs for the two layer of subkey addition and 7 XORs for the Q permutation. Therefore, in each round there are 32 table lookups and 25 XOR operations altogether, and 16KB memory is needed to store the big tables. Moreover, note that in each round, the computation of round functions for every 2 sub-blocks are exactly the same. Hence they can be performed in parallel for faster encryption speed. Moreover, decryption and encryption process of our designed cipher are similar which will also reduce the cost of implementation.

6.2 Security Evaluation

In this section, we give a brief evaluation of the security of block cipher VGF2 against several widely used cryptanalytic techniques.

Differential Cryptanalysis. For the MDS matrix and binary matrix Q used in this cipher, their differential branch numbers are $B_D = 9$ and $B_Q = 4$ respectively. Therefore, according to Proposition 3 in Sect. 4.2, for six rounds of

VGF2, there are at least 36 active sboxes. Then the best differential probability of six rounds is less than $2^{-36*6} = 2^{-216}$. Moreover, for 16 rounds there will be at least 90 active sboxes, which means the best differential probability of 16 rounds is less than $2^{-90*6} = 2^{-540}$. Hence we can conclude that the full rounds of VGF2 is practically secure against differential cryptanalysis.

Linear Cryptanalysis. Similarly, according to Proposition 4 in Sect. 4.2, we can conclude that for six rounds of VGF2, there are at least 36 linear active sboxes, and for 16 rounds there are at least 90 linear active sboxes. Considering that the best linear bias of six rounds is no more than $2^{35} \cdot 2^{-36*4} = 2^{-109}$, we can conclude that full rounds of VGF2 will be secure against linear cryptanalysis.

Impossible Differential Cryptanalysis. According to Proposition 5 in Sect. 4.2, the best impossible differential existed in VGF2 cipher is the 6-round impossible differential. Considering that VGF2 contains 16 rounds and in each round 256-bit subkeys are used, it is unlikely that the key recovery attack can work against the full cipher using this kind of 6-round impossible differential.

Integral Attack. For block cipher VGF2, the best integral characteristic existed is a 4-round integral distinguisher. If there is only one active byte, then after F transformation in Round 3, two sub-blocks of the intermediate state are still balance, and the following Q transformation will destroy this property. We can see that full rounds of VGF2 is secure against integral attack since it is unlikely that key recovery attack can work using this 4-round integral characteristic.

7 Conclusion

In this paper, we have proposed two block cipher structures which are variants of SP-network and Generalized Feistel structure respectively. Then we explain the design rationales and evaluate the security of these two structures under main attack settings. In the end, we have presented two example ciphers based on the structures proposed. We also estimate their implementation performances and the security of these ciphers against main attacks briefly.

Note that the block cipher Hierocrypt has an nested-SPN structure which is similar to our first structure. It also employs the combination of SDS as round function, and then employs another MDS matrix to mix all the bytes together. However, our proposition uses a simple binary matrix which operates on 64-bit words to mix all the sub-blocks together, and this obviously requires much less operations while still keeps the security level. Furthermore, in this paper we only give a brief evaluation of the ciphers, and tests of implementation performances on various platforms and more security analysis will be our future work.

Acknowledgement

This work is supported by the National High-Tech Research and Development 863 Plan of China (No.2007AA01Z470), and the National Natural Science

Foundation of China (No.60873259). Moreover, we would like to thank the anonymous reviewers for their helpful comments and editorial suggestions.

References

1. Data Encryption Standard (DES). Federal Information Processing Standards Publication FIPS-46-3. National Bureau of Standards (1999)
2. Daemen, J., Rijmen, V.: The Design of Rijndael - AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
3. Nyberg, K.: Generlized Feistel Networks. In: Kim, K.-c., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 91–104. Springer, Heidelberg (1996)
4. Knudsen, L.R.: DEAL - A 128-bit Block Cipher. Technical Report 151, Department of Informatics, University of Bergen, Bergen, Norway (1998)
5. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-Bit Block-cipher CLEFIA (Extended Abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007)
6. Tsunoo, Y., Tsujihara, E., Shigeri, M., Saito, T., Suzaki, T., Kubo, H.: Impossible Differential Cryptanalysis of CLEFIA. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 398–411. Springer, Heidelberg (2008)
7. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and Related-Key Attack on the Full AES-256. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009. LNCS, vol. 5677, pp. 231–249. Springer, Heidelberg (2009)
8. Biryukov, A., Khovratovich, D.: Related-Key Cryptanalysis of the Full AES-192 and AES-256. In: ASIACRYPT 2009, vol. LNCS (2009) (to appear)
9. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystem (Extended Abstract). In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991)
10. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
11. 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)
12. Nyberg, K., Kundsen, L.R.: Provable Security Against Differential Cryptanalysis. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 566–574. Springer, Heidelberg (1993)

Hardware Framework for the Rabbit Stream Cipher

Deian Stefan[⋆]

S*ProCom^2// Dept. of Electrical Engineering,
The Cooper Union,
New York NY 10003, USA

Abstract. Rabbit is a software-oriented synchronous stream cipher with very strong security properties and support for 128-bit keys. Rabbit is part of the European Union's eSTREAM portfolio of stream ciphers addressing the need for strong and computationally efficient (i.e., fast) ciphers. Extensive cryptanalysis confirms Rabbit's strength against modern attacks; attacks with complexity lower than an exhaustive key search have not been found. Previous software implementations have demonstrated Rabbit's high throughput, however, the performance in hardware has only been estimated. Three reconfigurable hardware designs of the Rabbit stream cipher – direct, interleaved and generalized folded structure (GFS) – are presented. On the Xilinx Virtex-5 LXT FPGA, a direct, resource-efficient (568 slices) implementation delivers throughputs of up to 9.16 Gbits/s, a 4-slow interleaved design reaches 25.62 Gbits/s using 1163 slices, and a 3-slow 8-GFS implementations delivers throughputs of up to 3.46 Gbits/s using only 233 slices.

Keywords: FPGA, Rabbit, eSTREAM, DSP, Stream Cipher.

1 Introduction

The widespread use of embedded mobile devices poses the need for fast, hardware-oriented encryption capabilities to provide higher security and protection of private data for end users. Stream ciphers are cryptographic algorithms that transform a stream of plaintext messages of varying bit-length into ciphertext of the same length, usually by generating a *keystream* that is then XORed with the plaintext. In general, stream ciphers have very high throughput, strong security properties, and use few resources, thus making them ideal for mobile applications; well-known examples of stream ciphers include the RC4 cipher used in 802.11 Wireless Encryption Protocol [13], E0 cipher used in Bluetooth protocol [13], and the SNOW 3G cipher used by the 3GPP group in the new mobile cellular standard [26].

The European Union sponsored the four-year eSTREAM project to identify new stream ciphers which address not only strong security properties, but also

⋆ Part of this work was done while the author was visiting EPFL, Switzerland.

the need for 1) high-performance software-oriented ciphers and, 2) low-power and low-resource hardware-oriented ciphers. The Rabbit stream cipher is among four software-oriented stream ciphers which were selected for the eSTREAM software portfolio in 2008 [3]. Rabbit performs very well in software (e.g., 5.1 cycles/byte on a 1.7 GHz Pentium 4 and 3.8 cycles/byte on a 533 MHz PowerPC 440GX [6]) and detailed cryptanalysis by the designers and recent studies [2,20] found no serious weaknesses or attacks more feasible than an exhaustive key search. In [20], Lu *et al.* estimate the complexity of a time-memory-data-tradeoff (TMDT) key-recovery attack to be $2^{97.5}$ with 2^{32} memory usage, 2^{32} precomputations in addition to an exceptionally strong adversary assumption. Moreover, they also present the best distinguishing attack with complexity 2^{158}, which is considerably higher than the exhaustive key search of 2^{128}. The strong security properties of Rabbit makes the cipher a desirable candidate for both software and hardware applications. Until now there were no hardware implementations of Rabbit to evaluate its performance, only estimates of application-specific integrated circuit (ASIC) and field-programmable gate array (FPGA) designs; as part of our framework, we present three different architectures suitable for reconfigurable hardware implementations that can be used as standalone hardware or hardware/software co-designs for both cryptographic and cryptanalytic applications.

First we introduce the structure of the Rabbit stream cipher and the mathematical foundations. We then discuss the three hardware architectures of the algorithm: direct, interleaved, and generalized folded structure. The tradeoffs of each are considered along with hardware- and software-based initialization designs. Finally, FPGA implementations and performance benchmarks are presentd.

2 Structure of Rabbit

Rabbit is a *symmetric synchronous* stream cipher with a 513-bit internal state derived from the 128-bit key and an optional 64-bit initial vector (IV). From the classical definition of a synchronous stream cipher [22], the internal state during each system iteration is updated according to a *next-state* function dependent on the previous (internal) state, and similarly, the keystream is produced as a function of the internal states, independent of the plaintext or ciphertext. An *output function*, XOR in this case, is then used to combine the plaintext (ciphertext) message and keystream to produce the output ciphertext (plaintext).

The 128-bit key allows for the safe encryption of 2^{64} plaintext messages [21,6], while the optional (public) 64-bit IV provides for the safe encryption of up to 2^{64} plaintexts using the same key [8]. Many stream cipher keystream generators are based on the irregular clocking, non-linear combination, or non-linear filtering of the output(s) of linear feedback shift registers (LFSRs) and pseudo-random number generators (PRNGs) [24,22]. The Rabbit design, although counter-assisted and dependent on the highly non-linear mixing of the internal state, is a novel approach to stream cipher design, adopting random-like properties from chaos theory [7].

The Rabbit 513-bit internal state (at iteration i) is divided into eight 32-bit state variables $x_{j,i},\ 0 \leq j \leq 7$, eight 32-bit counters $c_{j,i},\ 0 \leq j \leq 7$ and a carry bit $\phi_{7,i}$. The design choice of a very large internal state makes TMDT attacks (e.g., key recovery), which rely on "off-line" precomputations to minimize "on-line" computing time, infeasible [5,6].

2.1 Internal State Update

The internal state update, i.e., a system iteration, is divided into the non-linear next-state update of the state variables $x_{j,i}$'s, and the linear update of the counter variables $c_{j,i}$'s.

Next-state update: At the core of the Rabbit algorithm is the iteration of the state variables, from which the keystream is generated. After the initialization of the internal state (explained in Section 2.2) the next-state function, depending only on the previous state, is used to iterate the system; so, the internal state at iteration $i+1$ depends solely on the non-linear mixing of the internal state at i. Formally, following the notation of [6], the eight 32-bit state variables are updated as follows:

$$x_{j,i+1} = \begin{cases} g_{j,i} + g_{j-1,i} \lll 16 + g_{j-2,i} \lll 16 & \text{for } j \text{ even} \\ g_{j,i} + g_{j-1,i} \lll 8 + g_{j-2,i} & \text{for } j \text{ odd}, \end{cases} \tag{1}$$

where $\lll \alpha$ is a bitwise-rotation by α bits, the additions are mod 2^{32} and all the indices $j-k,\ 0 \leq k \leq 2$ are mod 8 (the number of state and counter variables). The chaos-inspired function g is defined as:

$$g_{j,i} = ((x_{j,i} + c_{j,i+1})^2 \oplus ((x_{j,i} + c_{j,i+1})^2 \gg 32)) \mod 2^{32}, \tag{2}$$

where $\gg \alpha$ is a bitwise right-shift by α and the inner additions, $(x_{j,i} + c_{j,i+1})$ are mod 2^{32}. The g function is the source of the high non-linearity in the state updates — 256 bits (all the bits of the $x_{j,i}$'s) of the 513-bit internal state are non-linearly transformed; as (1) shows, each state variable is a combination of three outputs from the g function. The g function is the source of the cipher's resistance to algebraic, differential, and linear correlation attacks, which commonly take advantage of ciphers with few non-linear state updates, or the correlation between the difference of inputs and outputs. These attacks seek to determine an output's dependence on the input, find a correlation between the output and internal state or distinguish the keystream from a truly random sequence [12,1,7,6].

Counter update: Similar to the state variable updates, during each iteration the eight 32-bit counter variables are also updated, although linearly, according to:

$$c_{j,i+1} = \begin{cases} c_{0,i} + a_0 + \phi_{7,i} & \text{for } j = 0 \\ c_{j,i} + a_j + \phi_{j-1,i+1} & \text{otherwise} \end{cases} \tag{3}$$

where,

$$a_j = \begin{cases} \texttt{0x4d34d34d} & \text{for } j = 0, 3, 6 \\ \texttt{0xd34d34d3} & \text{for } j = 1, 4, 7 \\ \texttt{0x34d34d34} & \text{otherwise} \end{cases} \tag{4}$$

and the carry $\phi_{j,i+1}$ is:

$$\phi_{j,i+1} = \begin{cases} 1 & \text{if } j = 0 \text{ and } c_{0,i} + a_0 + \phi_{7,i} \geq 2^{32} \\ 1 & \text{if } j \neq 0 \text{ and } c_{j,i} + a_j + \phi_{j-1,i+1} \geq 2^{32} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

It can be shown that the 256-bit counter state (eight 32-bit counters) has a maximal period length of $2^{256} - 1$ [7], and since the counter variables are used in (2), and thus in the next-state function (1), a lower-bound on the period length of the state variables can also be guaranteed [7,6].

2.2 Initialization

Key setup: The 128-bit key K is divided into eight 16-bit sub-keys $K = k_7 || \cdots || k_0$, where $||$ is the concatenation operation, with the least significant bit (LSB) bit of k_0 and most significant bit (MSB) of k_7 corresponding to the LSB and MSB of K, respectively. The key is expanded to initialize the counter and state variables according to:

$$x_{j,0} = \begin{cases} k_{j+1} || k_j & \text{for } j \text{ even} \\ k_{j+5} || k_{j+4} & \text{for } j \text{ odd,} \end{cases} \tag{6}$$

and:

$$c_{j,0} = \begin{cases} k_{j+4} || k_{j+5} & \text{for } j \text{ even} \\ k_j || k_{j+1} & \text{for } j \text{ odd,} \end{cases} \tag{7}$$

where the indices $j + k$ are modulo 8. Additionally, the carry $\phi_{7,0}$ is initialized to zero.

Following the key expansion, the system is iterated four times according to the next-state and counter-update functions described in Section 2.1, and finally the counter variables are modified according to:

$$c_{j,4} = c_{j,4} \oplus x_{j+4,4}, \tag{8}$$

where the indices are again mod 8.

The expansion of the key is such that there is a one-to-one correspondence between the key and the 512-bit internal state, while the four system iterations and counter modifications assert both 1) the mixing of all the key bits with every state variable and 2) the combination of the counter with the non-linear state variables [6]. It is important to avoid a many-to-one mapping between the key and internal state as this drastically degrades the strength of the algorithm, for if two keys lead to the same internal state an adversary could potentially generate the same keystream with a different key. Equally essential are the counter modifications, as they prevent key-recovery attacks in which an adversary, with knowledge of the counter's state, can 'clock' the system in reverse and deduce the key. Since the next-state function is resistant to guess-and-verify and correlation attacks [6], and thus resistant to the 'reverse clocking' of the state variables, the modification of the counter variables as in (8) secures against key-recovery attacks.

IV setup: If a 64-bit IV is provided, it is divided into four 16-bit sub-IVs — $IV = iv_3||\cdots||iv_0$ — where the LSB of iv_0 and MSB of iv_3 correspond to the LSB and MSB of IV, respectively. Using the sub-IVs the counters are modified to:

$$c_{j,4} = \begin{cases} c_{j,4} \oplus iv_1||iv_0 & \text{for } j = 0,4 \\ c_{j,4} \oplus iv_3||iv_1 & \text{for } j = 1,5 \\ c_{j,4} \oplus iv_3||iv_2 & \text{for } j = 2,6 \\ c_{j,4} \oplus iv_2||iv_0 & \text{for } j = 3,7, \end{cases} \tag{9}$$

after which the system is again iterated four times, guaranteeing the non-linear combination of all the IV bits into the state variables [6].

2.3 Keystream Generation

During each iteration i, the state variables $x_{j,i}$ are split into low (L) and high (H) 16-bit sub-states $x_{j,i} = x_{j,i,\mathrm{H}}||x_{j,i,\mathrm{L}}$, from which the 128-bit keystream output, a concatenation of eight 16-bit blocks $s_i = s_{i,7}||\cdots||s_{i,0}$, is extracted according to:

$$\begin{aligned} s_{i,0} &= x_{0,i,\mathrm{L}} \oplus x_{5,i,\mathrm{H}} & s_{i,4} &= x_{4,i,\mathrm{L}} \oplus x_{1,i,\mathrm{H}} \\ s_{i,1} &= x_{0,i,\mathrm{H}} \oplus x_{3,i,\mathrm{L}} & s_{i,5} &= x_{4,i,\mathrm{H}} \oplus x_{7,i,\mathrm{L}} \\ s_{i,2} &= x_{2,i,\mathrm{L}} \oplus x_{7,i,\mathrm{H}} & s_{i,6} &= x_{6,i,\mathrm{L}} \oplus x_{3,i,\mathrm{H}} \\ s_{i,3} &= x_{2,i,\mathrm{H}} \oplus x_{5,i,\mathrm{L}} & s_{i,7} &= x_{6,i,\mathrm{H}} \oplus x_{1,i,\mathrm{L}}. \end{aligned} \tag{10}$$

It is important that adversaries gain no information from the output, that is, they should not be able to distinguish the output of the keystream generator from a truly random sequence [15]. The combination of the outputs of the non-linear g function in the keystream extraction highlights the strength of Rabbit in passing various statistical tests [6], including the NIST Test Suite that seeks to find non-randomness in a sequence [4].

3 Rabbit in Hardware

As previously mentioned, Rabbit is a software-oriented stream cipher and thus was designed to perform well on general purpose architectures, varying from 32-bit Intel processors to 8-bit microcontrollers. Estimates of ASIC and FPGA throughput and area performance are presented in [6], however the implementation details are limited. In the following sections, we consider three architecture designs of the Rabbit algorithm optimized for reconfigurable devices.

3.1 Direct Architecture and General Optimizations

The first architecture we consider is a direct implementation of the algorithm. Observing the relationship between (3) and (5), we note that the counter variables can be updated using a series of chained adders. Each adder takes inputs $c_{j,i}$, a_j and carry-in[1] $\phi_{j-1,i+1}, j > 0$, producing output $c_{j,i+1}$ and carry-out $\phi_{j,i+1}$ each cycle. Figure 1 illustrates the chaining method within the full architecture design. The updated counters $c_{j,i+1}$ and state variables $x_{j,i}$ are then used as inputs to the g function blocks, the outputs of which, $g_{j,i}$, are combined according to (1) to produce the next state variables $x_{j,i+1}$. Moreover, the next state variables are concurrently combined according to (10) to produce the 128-bit keystream output.

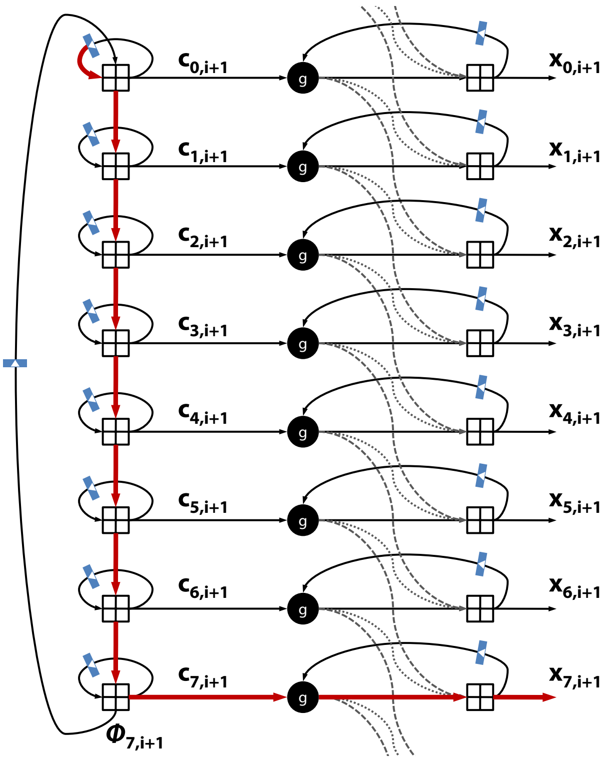


Fig. 1. Direct architecture of the Rabbit algorithm, highlighting the critical path. The $\boxplus$ is a 32-bit adder with carries, while the dotted and dashed lines indicate a variable rotate dependent on whether j is even or odd, see (1). Control logic, a_i inputs, initialization blocks and the keystream extractor are eliminated for clarity.

Below, we consider generic hardware optimizations, which are applied to all the designs in the framework, including the direct implementation.

[1] Note that the carry-in for the first adder is $\phi_{7,i}$.

Efficient squaring: In implementing the next-state function, eight parallel realizations of the g function are required. Accordingly, the implementation of g can greatly affect the overall speed performance and area usage. As Boesgaard *et al.* note [6], the most costly part of the g function, the squaring, can be efficiently implemented using three 16-bit multiplies followed by a 32-bit addition. If we let $u = x_{j,i} + c_{j,i+1}$ and split u into two 16-bit values $u = u_{\mathrm{H}}||u_{\mathrm{L}}$, then the optimization follows directly from the fact that $u^2 = u_{\mathrm{L}}^2 + 2^{32}u_{\mathrm{H}}^2 + 2^{17}u_{\mathrm{L}}u_{\mathrm{H}} \bmod 2^{32}$. Thus the full g function, as in (2), can be efficiently implemented using four (2-input) 32-bit adders, three 16-bit multipliers, 3 shifts (which have no cost in hardware, other than routing), and a 32-bit XOR.

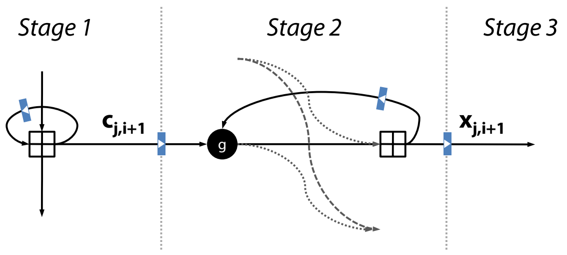


Fig. 2. Three-stage pipeline for the direct architecture of the Rabbit algorithm

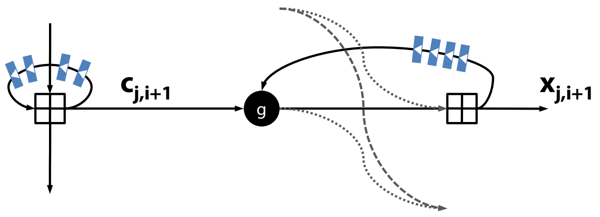
Pipelining: In addition to optimizing g, the speed of the direct design can be further increased by splitting the design into three pipeline stages. Without pipeline registers, the critical path – the path with the highest computational cost between two delay elements – consists of the eight counter adders, a g function (computing $g_{7,i}$), two 32-bit adders (computing $x_{7,i+1}$) and a 16-bit XOR (extracting keystream output); excluding the final XOR, the critical path is highlighted in Figure 1. The critical path can, however, be reduced to either eight 32-bit adders or g and two 32-bit adders[2] by introducing pipeline registers following the counter adders and preceding the keystream output XORs, see Figure 2. To retain correctness, keeping the inputs $c_{j,i+1}$ and $x_{j,i}$ to the g functions synchronized is required and can be accomplished by introducing a latency of one cycle (using clock-enables) for the $x_{j,i}$'s to match the latency introduced by the pipeline register for $c_{j,i+1}$.

***C*-slow retiming:** To further optimize the pipelined design, the critical path, which we experimentally determined to be in the second pipeline stage (the calculation of the the next state variables: g+two 32-bit adders), must be reduced with fine-grained pipelining of the g block, the costliest element in the path. We note that since $g_{j,i+1}$ depends on $x_{j,i+1}$, which is a function of the output of $g_{j,i}$, the direct design cannot take advantage of multiplier pipelining. Instead, we optimize the design with *C-slow retiming*, a DSP system-design technique

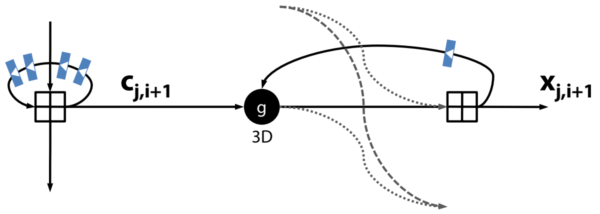
[2] Specifically, the critical path is max(eight 32-bit adders, g+two 32-bit adders).

that allows for the pipelining of structures with feedback loops [23,27]. C-slow retiming is a modification of a system design in which each register is replaced with C registers (C-slowed) after which the full structure is retimed, whilst retaining algorithmic correctness; we refer the reader to [23] for further details. For $C = 4$, Figure 3(a) illustrates the partial C-slow design before retiming, and Figure 3(b) shows it after retiming, where 3 of the 4 registers were moved into the g block. Retiming stage 2 can thus be seen as fine-grained pipelining of the g function into 3 simpler stages (addition, multiplication, and addition + XOR). Moreover, by pipelining g, the critical path is "reduced" to the eight 32-bit chained counter adders.

We note that although C-slow retiming can acutely increase the clock rate, the area usage will, in general, increase, as will the number of cycles it takes to complete a single iteration; specifically the number of cycles per iteration will increase to C. Thus to avoid zero-filling the $C - 1$ pipeline registers, it is essential that multiple streams be interleaved, running in parallel, so that during the C cycle system iteration, C independent streams are updated and C different keystream outputs are generated. Multi-stream cipher applications have been studied before (see e.g., [28,9]), and find use in many applications, including file system encryption, securing virtual private networks, and cryptanalysis.



(a) 4-slow before retiming



(b) 4-slow after retiming

Fig. 3. C-slow retiming for $C = 4$ is accomplished by first replacing each register with C of them, as shown in (a), followed by the retiming, which relocates registers to optimize the design, as shown in (b)

Initialization: Initialization of the direct architecture requires a key expansion block for (6) and (7), which consist of simple combinations of bit slices used to initialize the state and counter variables; additional control logic (multiplexers) and XORs are needed for the IV setup and modification of the counter as in (8).

For multi-stream (C-slow retimed) designs, control logic is necessary to correctly initialize the independent streams.

Alternatively, for hardware/software co-designs, the initialization can be performed in software from which the Rabbit hardware counter, state and carry registers can be loaded; the Rabbit crypto-co-processor and main CPU (e.g., MicroBlaze or PowerPC) can be interfaced using numerous bus protocols that can directly access hardware registers, including the Xilinx Fast Simplex Link (FSL), On-Chip Peripheral Bus (OPB) and the IBM-based Processor Local Bus (PLB). For many security system- and network-on-chip applications, which commonly consist of a CPU and peripherals in addition to the FPGA, initialization in software eliminates the need for additional hardware resources and further simplifies the overall design. Moreover, the saved resources can be dedicated to additional cryptographic cores in multi-stream applications, or to other hardware-assisted applications running concurrently, e.g., MPEG-4 encoder.

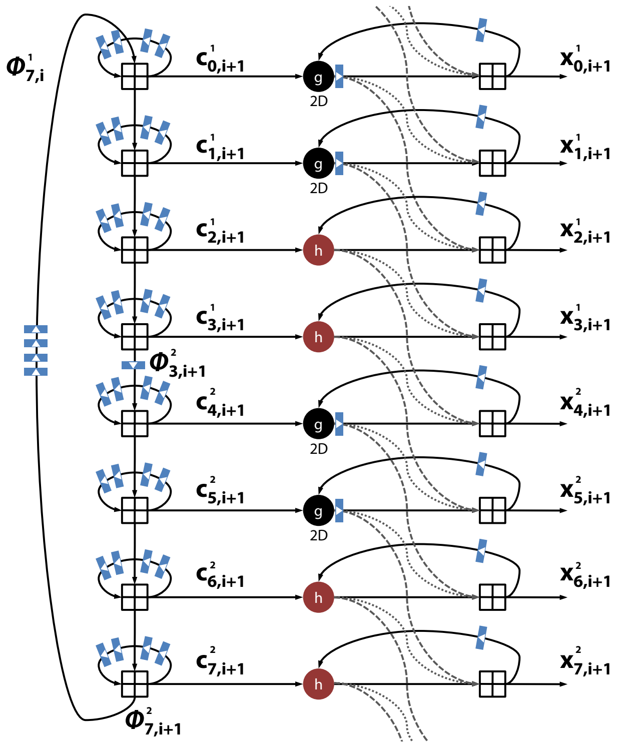
3.2 Interleaved Architecture

Although a C-slow retimed implementation is suitable for hardware, the high data-dependency between the counters (due to the percolating carries $\phi_{j,i+1}$) still poses a limitation on the clock rate. This is because a 256-bit addition[3] must be completed in a single cycle. For a 3-stage pipeline and C-slow retimed design (assuming $C \geq 2$), the cost can be reduced to that of a 128-bit addition using cut-set retiming; in this section we, however, focus on interleaved architecture (IA) design, which is a considerably more balanced structure. See Appendix A for further details on the cut-set retiming approach.

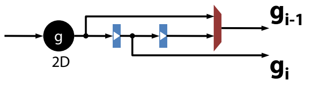
The interleaved design is a generalization of the C-slow retiming approach to fine-grained pipelining of, not only the state variable updates (stage 2 of the pipelined design in Figure 2), but the counter updates as well (stage 1). Given a C-slow design ($C = 2l, l \geq 1$), a C/k-interleaved architecture (in short C/k-IA) interleaves k independent streams in a *single clock cycle* for k cycles (ignoring the initial first cycle used to fill the pipeline), where $k \leq C$ and $k|8$. For example, a 2/2-IA consists of 2 streams which are interleaved such that during the first cycle half of the state variables of each stream are updated and during the second cycle the second half of the variables are updated. As another example, consider the 4/2-IA case; this design is equivalent to interleaving two 2/2-IA streams. We further note that the C-slow retimed design discussed in Section 3.1 is a special case for $k = 1$, i.e., $C/1$-IA.

We denote variables of different stream with a superscript, e.g., $c^m_{j,i}$ is the j-th counter variable at iteration i of stream m. For clarity we limit our discussion to the 4/2-IA design shown in Figure 4. From Figure 4 we observe that during the first cycle, half of stream 1's counters $c^1_{j,i+1}, 0 \leq j \leq 3$ and final half of stream 4's counters $c^4_{j,i+1}, 4 \leq j \leq 7$ are updated in the top and bottom of the structure, respectively. Because $\phi_{3,i+1}$ is buffered, in the following cycle we can

[3] The eight 32-bit additions with carries is equivalent to a 256-bit addition of $c_{7,j}||\cdots||c_{0,j}$ and $a_7||\cdots||a_0$ with $\phi_{7,i}$ as a carry-in.



(a) 4/2-IA design.



(b) h function

Fig. 4. 4/2-Interleaved Architecture design and corresponding h block

update $c^1_{j,i+1}, 4 \leq j \leq 7$ in the bottom half, and $c^2_{j,i+2}, 0 \leq j \leq 3$ in the top. Table 1 illustrates the update of the counter variables over time corresponding to Figure 4. With the exception of the first cycle, during every cycle a full-state update is completed.

Due to the interleaving and need to retain correctness of the algorithm, the retiming of g is slighlty more complex than that of a C-slow design. First, because we start from a 4-slow design, 2 registers can be dedicated to the fine-grained pipelining of g, while the others are used to buffer either 1) the output of g so that the next state variables can be computed according to (1) or 2) the next state variable. As the update is completed over 2 cycles, half of the g blocks need an additional register and a multiplexer (see Figure 4(b)) to select the correct g output; we denote this function by h. For example, in computing $x_{4,j+1}$, the outputs of the first two h blocks (h_2 and h_3) are the previously buffered $x_{2,j+1}$ and $x_{3,j+1}$ (and not the output of the g function).

Table 1. Example counter update of 4/2-IA over increasing time t

t	0	1	2	3	4	5	6
Top:	$c_{0,i}^1$	$c_{0,i}^2$	$c_{0,i}^3$	$c_{0,i}^4$	$\boldsymbol{c_{0,i+1}^1}$	$c_{0,i+1}^2$	$c_{0,i+1}^3$
	$c_{1,i}^1$	$c_{1,i}^2$	$c_{1,i}^3$	$c_{1,i}^4$	$\boldsymbol{c_{1,i+1}^1}$	$c_{1,i+1}^2$	$c_{1,i+1}^3$
	$c_{2,i}^1$	$c_{1,i}^2$	$c_{2,i}^3$	$c_{2,i}^4$	$\boldsymbol{c_{2,i+1}^1}$	$c_{2,i+1}^2$	$c_{2,i+1}^3$
	$c_{3,i}^1$	$c_{3,i}^2$	$c_{3,i}^3$	$c_{3,i}^4$	$\boldsymbol{c_{3,i+1}^1}$	$c_{3,i+1}^2$	$c_{3,i+1}^3$
Bottom:	—	$c_{4,i}^1$	$c_{4,i}^2$	$c_{4,i}^3$	$\boldsymbol{c_{4,i}^4}$	$c_{4,i+1}^1$	$c_{4,i+1}^2$
	—	$c_{5,i}^1$	$c_{5,i}^2$	$c_{5,i}^3$	$\boldsymbol{c_{5,i}^4}$	$c_{5,i+1}^1$	$c_{5,i+1}^2$
	—	$c_{6,i}^1$	$c_{6,i}^2$	$c_{6,i}^3$	$\boldsymbol{c_{6,i}^4}$	$c_{6,i+1}^1$	$c_{6,i+1}^2$
	—	$c_{7,i}^1$	$c_{7,i}^2$	$c_{7,i}^3$	$\boldsymbol{c_{7,i}^4}$	$c_{7,i+1}^1$	$c_{7,i+1}^2$

We further note that for the 4/2-IA, in addition to registers which buffer the next state variables, two keystream extractors are needed in order to produce four 128-bit outputs in four cycles.

3.3 Generalized Folded Structure

Although FPGAs contain digital signal processing (DSP) slices[4] that can be used in implementing an optimized direct or IA design, with the exception of the DSP-enhanced FPGAs (such as the Xilinx Spartan-3A, Virtex-5 SXT and Virtex-4 SX [29]), most FPGAs have a small number of DSP slices which may be necessary for applications other than the encryption module (e.g. Fast Fourier Transform block used for image processing). As such, we seek a more compact implementation of the Rabbit stream cipher.

From (1), (2), (3) and Figure 1, we observe the repeated use of identical circuit blocks in the design (e.g. block g followed by addition), which can be reduced to fewer shared copies at the cost of additional control logic and intermediate state registers. Specifically, the g block, adders and rotation blocks used to update a state variable can be shared to compute all the eight state variables at the cost of 1/8-th the time each computing block is used to update a state variable. Similar to the sharing of resources to update the state variables, the calculation of the eight counter variables at 1/8-th the time per resource can be accomplished by sharing a single adder and carry register.

In DSP terminology, the general design optimization is referred to as a n*-folded* or n*-rolled* design [23], reducing the number of used computational resources (e.g., g blocks) to $1/n$ at the cost of taking n cycles to complete a full iteration. It is constructive to think of folded designs as n threads running on a pipelined system sharing the same computational units, and during every cycle a different thread, cycled in a round–robin fashion, gets a chance to use the computational

[4] The design of a DSP slice is FPGA-family-specific, however the most common design is a 18×18 multiplier followed by an adder/accumulator and a small number of registers and multiplexers.

units (and advance in the pipeline) [16], such that after n cycles all the threads have finished their necessary computations and the iteration is complete.

Although a directly folded design of Rabbit is realizable, it is inefficient because each iteration requires $g_{6,i}$ and $g_{7,i}$ to compute the first two next-state variables, $x_{0,i+1}$ and $x_{1,i+1}$, and as such, an elegant solution buffering only the last two g values is not feasible without the use of an additional g block. Instead, we propose a generalized filter structure that allows access to intermediate values—following the threading analogy: the threads are no longer independent and can share data. Moreover, an n-GFS implementation only requires $1/n$ of the number of computational elements (e.g., adders and g functions) used by a direct implementation. As the counter implementation in an n-GFS architecture is the same as that of a folded design (i.e., in an n-GFS design, the counter system is simply the chaining of $8/n$ adders whose (partial) inputs are n delayed counter variables that need to be updated sequentially), we limit our discussion to the more interesting case of the state updates.

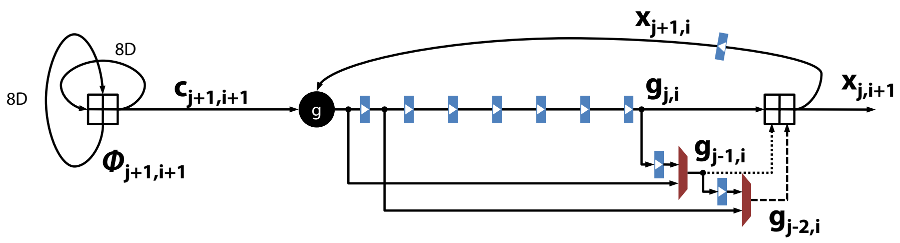


Fig. 5. 8-GFS design. Every 8-th cycle, the multiplexers select the $g_{7,i}$ and $g_{6,i}$ results for the $g_{j-1,i}$ and $g_{j-2,i}$ inputs of the 32-bit adder. The dashed and dotted lines highlight rotations dependent on j.

As shown in Figures 5 the 8-GFS design uses a minimal number of resources, both in terms of the register usage and computational elements (g function and adders). Only two additional registers, which buffer $g_{j-1,i}$ and $g_{j-2,i}$, are needed when computing $x_{j,i+1}$ according to (1). We note that every 8 cycles all intermediate terms, $g_{0,i}$ through $g_{7,i}$, are available and thus any of the next-state variables can be updated, including $x_{0,i+1}$. Similarly, Figure 6 shows the compact 4-GFS design split into a top and bottom pipeline, each computing even and odd next-state variables, respectively. As with the 8-GFS, every $n = 4$ cycles, all the intermediate terms are available and thus $x_{0,i+1}$ and $x_{2,i+1}$ can be computed. A 2-GFS design follows directly from these.

From the figures, we observe that a straight-forward GFS implementation will be limited by the rate at which it can be clocked (due to the fact that the critical path consists of a g block and two 32-bit adders). However, the pipelining and C-slow retiming techniques presented in Section 3.1 are adopted to further speed up the compact n-GFS designs.

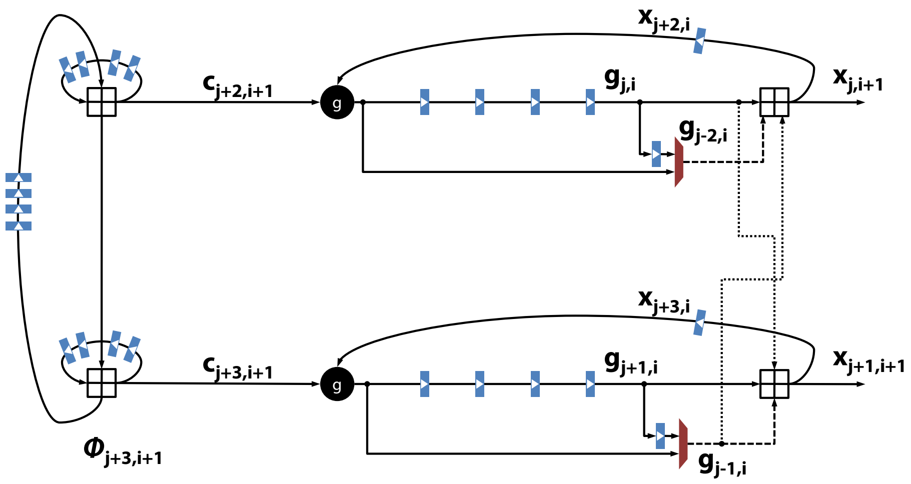


Fig. 6. 4-GFS design with the top pipeline computing every even state variable, and the bottom every odd. Every 4-th cycle, the top and bottom multiplexers select the $g_{6,i}$ and $g_{7,i}$ results, respectively. The dashed and dotted lines highlight rotations dependent on j.

Keystream extraction: To extract the keystream output according to (10), a time division demultiplexer (TDD) is needed so that $x_{j,i}$, $0 \leq j \leq 7$ are simultaneously available for the calculation of the s_i's. Since a TDD uses a considerable number of registers, applications of 8-GFS where variable output lengths and out-of-order keystreams are acceptable (such as random number generators), the TDD (and following XORs) can be replaced by two 16-bit XORs producing the following output sequence: $s_{i,0}||s_{i,1}$, $s_{i,7}||s_{i,4}$, $s_{i,2}||s_{i,3}$, $s_{i,6}$, $s_{i,5}$. As the 4-GFS does not directly benefit from this optimization, the keystream extractor of 4-GFS consists of a 2-to-8 TDD followed by a series of XORs to generate the output.

Initialization: The generalized filter structure has a very flexible initialization process. For an 8-GFS, the hardware initialization requires additional 1) four registers so that $x_{0,4}$ is available for the modification of $c_{4,4}$ according to (8), 2) two XORs for the mixing of the counters with the state variables and IV, 3) set of control logic. Similar requirements follow for the 4-GFS. We note that although minimal additions are needed for the hardware initialization, software initialization (as discussed in Section 3.1) can be used without the need for any additional resources.

4 Implementation and Discussion

Three direct designs, a 4/2-IA design, and various 4- and 8-GFS designs of the Rabbit cipher were implemented using System Generator and synthesized using

Xilinx XST (ISE 11.1). We targeted the Xilinx Virtex-5 LXT (XC5VLX50TFFG 1136) FPGA hosted on the Xilinx ML 501 development board, consisting of 7,200 slices, 60 Block RAMs and 48 DSP48 slices. Table 2 summarizes the post-place and route results, where the suffix V is used to identify the implementations with variable output rate (see Section 3.3). We stress the advantage of using C-slow retiming by observing that a direct design can be maximally clocked at 71.58 MHz, while the fine-grained pipelining of the g function increases the clock rate to 141.38 MHz. This nearly doubles the throughput from 9.16 Gbps to 18.10 Gbps, in addition to increasing throughput/area ratio. Although using SLICEM and SLICEL slices (memory- and logic-enhanced slices) for more efficient carry propagation endures a clock rate of 71.58 MHz, we notice the advantage of pipelining the adders in the very high throughput (25.62 Gbps) of the 4/2-IA design; we expect that using C/k-IA designs with $k > 2$ will further allow for an increase in the clock rate, and thus throughput. Furthermore, our results confirm that the estimates made in [6] are reasonably accurate.

Table 2. Rabbit Resource Usage and Performance Evaluation

Design	**Freq (MHz)**	**Slices (%)**	**DSP Slices(%)**	**Block RAMs(%)**	**Thruput (Gbps)**	**Mbps/ Slice**
(Rabbit)						
Direct	71.582	568 (7.88%)	24 (50%)	0 (0.00%)	9.16	16.10
Direct, 3-slow	137.155	884 (12.28%)	24 (50%)	0 (0.00%)	17.56	19.86
Direct, 4-slow	141.383	961 (13.35%)	24 (50%)	0 (0.00%)	18.10	18.83
4/2-IA	200.120	1163 (16.15%)	24 (50%)	0 (0.00%)	25.62	22.03
8-GFS	83.724	260 (3.61%)	3 (6%)	0 (0.00%)	1.34	5.15
8-GFS, 2-slow	138.198	368 (5.11%)	3 (6%)	0 (0.00%)	2.21	6.01
8-GFS, 2-slow, V	142.227	239 (3.32%)	3 (6%)	0 (0.00%)	2.28	9.52
8-GFS, 3-slow	214.638	351 (4.88%)	3 (6%)	0 (0.00%)	3.43	9.78
8-GFS, 3-slow, V	216.450	233 (3.24%)	3 (6%)	0 (0.00%)	3.46	14.86
4-GFS	85.697	360 (5.00%)	6 (12%)	0 (0.00%)	2.74	7.62
4-GFS 2-slow	155.982	602 (8.36%)	6 (12%)	0 (0.00%)	4.99	8.29
4-GFS 3-slow	195.198	588 (8.17%)	6 (12%)	0 (0.00%)	6.25	10.62
Estimate [6]	—	—	24	—	17.8	—
(eSTREAM)						
Mickey128 [25]	280.5	392 (2.86%)	0 (0.00%)	0 (0.00%)	0.56	1.43
Grain [14]	155	356 (46.35%)	—	—	2.48	6.97
Grain-128 [10]	181	48 (0.14%)	—	—	0.18	3.77
Trivium [14]	190	388 (10.83%)	—	—	12.16	31.34
(other)						
AES [11]	350	400 (—%)	0 (0.00%)	0 (0.00%)	4.1	10.2
AES [17]	168.3	5177 (37.8%)	—	84 (61.7%)	21.5	4.2
RC4 [18]	64	138 (8.98%)	—	3 (12.5%)	0.22	0.16
LILI-II [19]	—	866 (2.56%)	—	1 (0.69%)	0.24	0.28
SNOW 2.0 [19]	—	1015 (3.00%)	—	3 (2.08%)	5.659	5.57

Table 2 also shows the performances of the more compact n-GFS designs. The ascent from an 8- to 4-GFS shows a linear increase in the throughput, with only a slight increase in slice count. The single stream 4-GFS and 3-slow 8-GFS are ideal for resource-constrained environments, while delivering reasonably high throughputs (2.74 and 3.43 Gbps, respectively). For cases where variable rate and out-of-order keystream output is acceptable, we recommend the use of the 3-slow 8-GFS, as it outperforms the 4-GFS by more than 26% while using approximately 35% fewer slices, and half the number of DSP slices.

We measured the performance penalty and additional resource of using hardware-initialized designs as compared to hardware/software co-designs to be less than 5% and 10%, respectively. Moreover, since the initialization circuit will not be needed after initialization, we recommend the hardware/software co-design as a very resource efficient design approach.

For completeness, we also compare our results to other stream cipher implementations in Table 2. The table shows previous results of the three eSTREAM hardware-oriented ciphers; a direct comparison is difficult, since [14,10,25] are based on the Spartan-3, Virtex-II, and Virtex-II Pro FPGAs and we present results on the Virtex-5 (which is based on the new-generation 6-input LUT architecture). However, we observe that, in general, the throughput/slice ratio of our results is greater than that of Mickey 128 2.0 and comparable with that of Grain. Trivium's throughput/slice is higher than the compared stream ciphers, including our 4/2-IA, whose throughput is much higher than all three eSTREAM candidates. We stress that although Rabbit is a software-oriented stream cipher, its performance in hardware is commendable in terms of both throughput and area-usage.

Finally, we compare our results to the Advanced Encryption Standard (AES, Rijndael) and various well-known stream ciphers. In terms of speed, the compact 4-GFS 3-slow Rabbit outperforms all these ciphers, including the Virtex-5 implementation of AES [11], in addition to maintaining the highest throughput/area ratio of 10.62. Similarly, the 4/2-IA outperforms one of the fastest AES implementations [17]; again, a direct comparison is difficult since the AES block cipher of [17] was implemented on older generation Virtex-II Pro FPGAs. In addition to the very high speed performance of Rabbit in hardware, with the exception of RC4, the compact n-GFS implementations outperform the compared stream ciphers in terms of slices used as well; however we also expect the slice count of the compared ciphers to be lower on a Virtex 5.

5 Conclusion

The first hardware standalone and hardware/software co-designs of the Rabbit stream cipher were presented and optimized using DSP system design techniques. As part of the generalized hardware framework, three different architectures were presented: a direct, interleaved and generalized folded structure. These implementations on the Virtex-5 LXT FPGA outperform previous FPGA implementations of stream ciphers such as MICKEY-128, RC4 and LILI-II, while also maintaining area-efficiencies above 5 Mbps/slice. Future work includes further optimization of Rabbit for ASICs, low-power Spartan-6 FPGAs, and implementation of additional IA and GFS variants.

Acknowledgment. The author would like to thank Om Agrawal, David Nummey, and anonymous reviewers for their insightful comments and suggestions. The support of Fred L. Fontaine and S*ProCom^2, and Arjen K. Lenstra and LACAL is also appreciated.

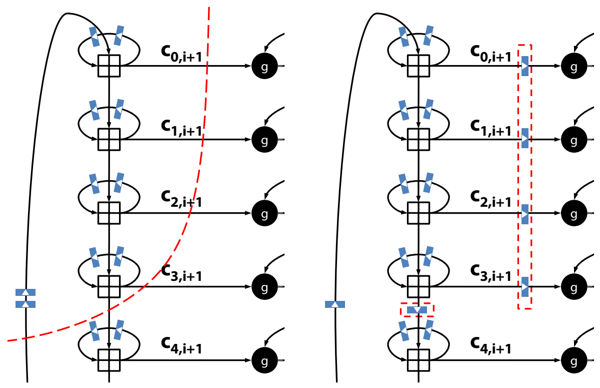
References

1. Cryptico A/S. Differential properties of the g-function (2003), http://www.cryptico.com/Files/filer/wp_differential_properties_gfunction.pdf
2. Aumasson, J.P.: On a bias of Rabbit. In: State of the Art of Stream Ciphers Workshop (SASC 2007), eSTREAM, ECRYPT Stream Cipher Project, Report (2007)
3. Babbage, S., Canniere, C., Canteaut, A., Cid, C., Gilbert, H., Johansson, T., Parker, M., Preneel, B., Rijmen, V., Robshaw, M.: The eSTREAM Portfolio. In: eSTREAM, ECRYPT Stream Cipher Project (2008)
4. Barker, E.B., Nechvatal, M.S., Barker, E., Leigh, S., Levenson, M., Vangel, M., Discussion, G., Studies, E.: A Statistical Test Suite For Random And Pseudorandom Number Generators For Cryptographic Applications
5. Biryukov, A., Shamir, A.: Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. LNCS, pp. 1–13. Springer, Heidelberg (2000)
6. Boesgaard, M., Vesterager, M., Christensen, T., Zenner, E.: The Stream Cipher Rabbit. In: ECRYPT Stream Cipher Project Report 6 (2005)
7. Boesgaard, M., Vesterager, M., Pedersen, T., Christiansen, J., Scavenius, O.: Rabbit: A new high-performance stream cipher. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 307–329. Springer, Heidelberg (2003)
8. Boesgaard, M., Vesterager, M., Zenner, E.: The Stream Cipher Rabbit. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 69–83. Springer, Heidelberg (2008)
9. Bos, J.W., Casati, N., Osvik, D.A.: Multi-stream hashing on the playstation 3. In: International Workshop on State-of-the-Art in Scientific and Parallel Computing 2008, Minisymposium on Cell/B.E. Technologies (2008)
10. Bulens, P., Kalach, K., Standaert, F.X., Quisquater, J.J.: FPGA implementations of eSTREAM phase-2 focus candidates with hardware profile. In: State of the Art of Stream Ciphers Workshop (SASC 2007), eSTREAM, ECRYPT Stream Cipher Project, Report (2007)
11. Bulens, P., Standaert, F.X., Quisquater, J.J., Pellegrin, P., Rouvroy, G.: Implementation of the AES-128 on Virtex-5 FPGAs. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 16–26. Springer, Heidelberg (2008)
12. Courtois, N.: Fast algebraic attacks on stream ciphers with linear feedback. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 176–194. Springer, Heidelberg (2003)
13. Ferro, E., Potorti, F.: Bluetooth and Wi-Fi wireless protocols: a survey and a comparison. IEEE Wireless Communications 12(1), 12–26 (2005)

14. Gaj, K., Southern, G., Bachimanchi, R.: Comparison of hardware performance of selected Phase II eSTREAM candidates. In: State of the Art of Stream Ciphers Workshop (SASC 2007), eSTREAM, ECRYPT Stream Cipher Project, Report (2007)
15. Goldreich, O.: Foundations of Cryptography: Basic Tools. Cambridge University Press, New York (2000)
16. Hauck, S., DeHon, A.: Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation. Morgan Kaufmann, San Francisco (2007)
17. Hodjat, A., Verbauwhede, I.: A 21.54 Gbits/s fully pipelined AES processor on FPGA. In: 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM 2004, pp. 308–309 (2004)
18. Kitsos, P., Kostopoulos, G., Sklavos, N., Koufopavlou, O.: Hardware implementation of the RC4 stream cipher. In: Proceedings of the 46th IEEE International Midwest Symposium on Circuits and Systems, MWSCAS 2003, vol. 3 (2003)
19. Leglise, P., Standaert, F.X., Rouvroy, G., Quisquater, J.J.: Efficient implementation of recent stream ciphers on reconfigurable hardware devices. In: 26th Symposium on Information Theory in the Benelux, pp. 261–268 (2005)
20. Lu, Y., Wang, H., Ling, S.: Cryptanalysis of Rabbit. In: Proceedings of the 11th International Conference on Information Security, pp. 204–214. Springer, Heidelberg (2008)
21. Mao, W.: Modern Cryptography: Theory and Practice. Prentice Hall Professional Technical Reference (2003)
22. Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC Press, Boca Raton (1997)
23. Parhi, K.K.: VLSI Digital Signal Processing Systems: Design and Implementation. Wiley, Chichester (1999)
24. Schneier, B.: Applied Cryptography Second Edition: protocols, algorithms, and source code in C. John Wiley and Sons, Chichester (1996)
25. Stefan, D., Mitchell, C.: Parallelized Hardware Implementation of the MICKEY-128 2.0 Stream Cipher. In: State of the Art of Stream Ciphers Workshop (SASC 2007), eSTREAM, ECRYPT Stream Cipher Project, Report (2007)
26. I.A. UEA2&UIA. Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2& UIA2. Document 2: SNOW 3G Specifications. Version: 1.1. ETSI/SAGE Specification (2006)
27. Weaver, N., Markovskiy, Y., Patel, Y., Wawrzynek, J.: Post-placement C-slow retiming for the Xilinx Virtex FPGA. In: Proceedings of the 2003 ACM/SIGDA Eleventh International Symposium on Field Programmable Gate Arrays, pp. 185–194. ACM, New York (2003)
28. Wee, C.M., Sutton, P.R., Bergmann, N.W., Williams, J.A.: Multi stream cipher architecture for reconfigurable system-on-chip. In: International Conference on Field Programmable Logic and Applications, FPL 2006, pp. 1–4 (August 2006)
29. Xilinx. DSP Solutions Using FPGAs (2009), http://www.xilinx.com/products/design_resources/dsp_central/grouping/fpgas4dsp.htm

A Cut-Set Retiming

Given a data flow graph G, cut-set retiming is a technique in which the graph is split into two disconnected subgraphs G_0 and G_1. Further, for every edge from



(a) Determining the cut-set (b) Cut-se retiming

Fig. 7. Example of cut-set retiming to pipeline the chained counter adders

G_0 to G_1, k delays are added and, similarly, for every edge from G_1 to G_0 k delays are removed (note that this assumes the existence of the k delays). We refer to [23] for additional details. Figure A shows part of the chained counter adders of a 4-slow Rabbit cipher with an example of a cut-set (Figure 7(a)) and the respective retiming (Figure 7(b)). This particular example shows a reduction from a 256-bit addition to two 128-bit additions. Similarly, for $C = 4$ and $C = 8$-slow designs, the 256-bit addition can be further reduced to four 64-bit or eight 32-bit additions, respectively. We note that the IA design of Section 3.2 can be similarly pipelined, however unlike the latter, the cut-set retimed design leads to an unbalanced design with a buildup of many registers between $c_{0,j+1}$ and the g function. As such, we prefer the IA design approach.

Linearity within the SMS4 Block Cipher

Muhammad Reza Z'aba, Leonie Simpson,
Ed Dawson, and Kenneth Wong

Information Security Institute, Queensland University of Technology,
GPO Box 2434, Brisbane, Queensland 4001, Australia
m.zaba@isi.qut.edu.au, {lr.simpson,e.dawson,kk.wong}@qut.edu.au

Abstract. We present several new observations on the SMS4 block cipher, and discuss their cryptographic significance. The crucial observation is the existence of fixed points and also of simple linear relationships between the bits of the input and output words for each component of the round functions for some input words. This implies that the non-linear function T of SMS4 does not appear random and that the linear transformation provides poor diffusion. Furthermore, the branch number of the linear transformation in the key scheduling algorithm is shown to be less than optimal. The main security implication of these observations is that the round function is not always non-linear. Due to this linearity, it is possible to reduce the number of effective rounds of SMS4 by four. We also investigate the susceptibility of SMS4 to further cryptanalysis. Finally, we demonstrate a successful differential attack on a slightly modified variant of SMS4. These findings raise serious questions on the security provided by SMS4.

Keywords: SMS4, block cipher, round function, fixed point, encryption, key scheduling algorithm, linearity, cryptanalysis.

1 Introduction

SMS4 [14,7] is a 32-round block cipher with 128-bit input block and 128-bit master key. It is used in the Chinese Wireless LAN Wired Authentication and Privacy Infrastructure (WAPI). Using the terminology of Schneier and Kelsey [16], the cipher employs a homogeneous, complete, source-heavy unbalanced Feistel network structure. The encryption and the key scheduling algorithms are nearly identical. The only difference between the structures of these two algorithms is the linear transformation used in each round function.

Since SMS4 was made public in January 2006, the cipher has endured extensive cryptanalysis. Reduced-round versions of the cipher have been cryptanalyzed using integral [12], rectangle [13,17,19,10], impossible differential [13,17], boomerang [10], differential [10,19] and linear [10,8] attacks. The best attack so far is a differential attack on 22 rounds by Zhang et al. [18]. In the same paper, they observe that the number of rotations and XOR operations used in the linear transformation of the SMS4 block cipher is the minimum required to achieve an optimal branch number. They also show that the linear transformation is bijective and present the distribution of input and output patterns of this transformation to assist in differential attacks.

In this paper, we present further observations on both the encryption and the key scheduling algorithms of the SMS4 block cipher. The crucial observation is the existence of fixed points and also of simple linear relationships between the bits of the input and output words for each component in the round functions. In particular, we show that the non-linear function T has 11 fixed points. Note that the expected number of fixed points for a random permutation is one [9, Chap. 6]. Therefore, the function T does not behave like a random permutation. We also identified a set of input words for which the round functions of both the encryption and the key scheduling algorithms produce the same output words. Furthermore, we show that the branch number of the linear transformation in the key scheduling algorithm is four, which is less than optimal.

One of the implications of these observations is that the first four round functions of SMS4 are not always non-linear. Under this condition, the number of effective rounds is reduced by four: from 32 to 28. We briefly explore the susceptibility of SMS4 against algebraic and advanced variants of the slide attacks. Finally, we demonstrate that if the linear transformation in the key scheduling algorithm was used in the encryption algorithm, then this variant of SMS4, reduced to 27 rounds, is vulnerable to a differential attack. In contrast, the best differential attack on the original SMS4 is on 22 rounds [18], which is also the best existing attack so far. These observations might potentially be useful in attacking SMS4 itself.

This paper is organized as follows. Section 2 describes the specification of the SMS4 block cipher. The observations on the components in the round functions of both the encryption and the key scheduling algorithms are analyzed in Section 3. Section 4 discusses the cryptographic significance of these observations. Section 5 presents a differential attack on a slightly modified variant of SMS4. A summary of our observations and conclusions are given in Section 6.

2 Specification of SMS4

SMS4 [14,7] is a block cipher that accepts a 128-bit plaintext block P, and a 128-bit master key K. The master key is used as input to the key scheduling algorithm to produce a set of thirty-two 32-bit round subkeys. The plaintext block and the round subkeys are used as input to the encryption algorithm to produce the ciphertext block C. The encryption algorithm consists of 32 applications of the round function.

2.1 Round Function of the Encryption Algorithm

Let $P = (X_0, X_1, X_2, X_3)$ denote the 128-bit plaintext block formed from the concatenation of four 32-bit words X_i. Let K_i denote the 32-bit i-th round subkey derived from the 128-bit master key K. The derivation of these subkeys is explained in Section 2.2. Let $T = L \circ S$ denote the function composed of the non-linear transformation S and the linear transformation L. Both S and L are described in detail later. The i-th round function of the encryption algorithm can be described as follows:

$$X_{i+4} = X_i \oplus T(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus K_i),\ i = 0, 1, \ldots, 31$$

and is depicted in Figure 1. The ciphertext consists of the concatenation of the four 32-bit words $C = (X_{35}, X_{34}, X_{33}, X_{32})$, which is obtained in the reverse order from the output of the final round function to facilitate decryption.

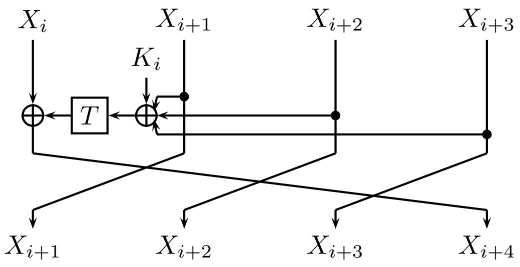


Fig. 1. Round Function of SMS4 in Round i

Decryption is the same as encryption with the only difference being the order in which the subkeys are used; this is in the reverse order as follows:

$$X_i = X_{i+4} \oplus T(X_{i+3} \oplus X_{i+2} \oplus X_{i+1} \oplus K_i),\ i = 31, 30, \ldots, 0.$$

The function T is the composition of the two transformations S and L, where S is applied first, followed by L. These transformations operate on 32-bit words. Let $X_i = (X_{i,0}, X_{i,1}, X_{i,2}, X_{i,3})$ denote a 32-bit word formed from the concatenation of four 8-bit words $X_{i,j}$. The application of the non-linear transformation S to X_i consists of the application of a single 8×8 S-box s to $X_{i,j}$ as follows:

$$S(X_i) = (s(X_{i,0}), s(X_{i,1}), s(X_{i,2}), s(X_{i,3})).$$

Let $X_i \lll k$ denote the rotation of X_i by k bits to the left. The linear transformation L is defined as:

$$L(X_i) = X_i \oplus (X_i \lll 2) \oplus (X_i \lll 10) \oplus (X_i \lll 18) \oplus (X_i \lll 24).$$

2.2 Round Function of the Key Scheduling Algorithm

In the initialization phase of the key scheduling algorithm, a 128-bit constant FK is XORed with the 128-bit master key K to produce the initial inputs for the key scheduling algorithm. Let $K = (MK_0, MK_1, MK_2, MK_3)$ denote the master key formed from the concatenation of four 32-bit words MK_i. Similarly, let $FK = (FK_0, FK_1, FK_2, FK_3)$ denote the constant as the concatenation of four 32-bit words FK_i, where $FK_0 =$ `A3B1BAC6`, $FK_1 =$ `56AA3350`, $FK_2 =$ `677D9197` and $FK_3 =$ `B27022DC` (in hexadecimal). Then, the initial input words to the key scheduling algorithm are $K_{i-4} = MK_i \oplus FK_i$ for $i = 0, 1, 2, 3$. Note that this initialization phase has no cryptographic significance because the operation is linear and the constants are known.

Let $T' = L' \circ S$ denote the function composed of the non-linear transformation S and the linear transformation L' (L' is described later). Note that this transformation L' is the only difference between the round functions of the encryption and the key scheduling algorithms. Let K_i and CK_i denote the i-th round 32-bit subkey and constant, respectively. The i-th round function of the key scheduling algorithm can be described as follows:

$$K_i = K_{i-4} \oplus T'(K_{i-3} \oplus K_{i-2} \oplus K_{i-1} \oplus CK_i),\ i = 0, 1, \ldots, 31.$$

The round constants $CK_i = (CK_{i,0}, CK_{i,1}, CK_{i,2}, CK_{i,3})$, which are composed of the concatenation of four 8-bit words $CK_{i,j}$, are defined as

$$CK_{i,j} = (28i + 7j) \bmod 256,\ i = 0, 1, \ldots, 31 \text{ and } j = 0, 1, 2, 3.$$

The linear transformation L' is defined as:

$$L'(X) = X \oplus (X \lll 13) \oplus (X \lll 23).$$

3 Observations on Components in the Round Functions

This section presents several new observations on each component in the round functions of both the encryption and the key scheduling algorithms of SMS4.

3.1 Simple Linear Relationships between Input and Output Words

We observe the existence of a simple linear relationship between the bits of some input and output words of each component in the encryption and the key scheduling algorithms. For a component F, there exist a set of output words of F which are equivalent to a simple rotation of the input word. That is, for some 32-bit words X_i,

$$F(X_i) = X_i \lll j \tag{1}$$

for some particular rotation values of $j \in \{0, 1, \ldots, 31\}$. A fixed point is a special case of this relationship when $j = 0$. For example, consider the linear transformation L, i.e. $F(X_i) = L(X_i)$ and the input word $X_i =$ `02020202`. The output word is $F($`02020202`$) =$ `08080808`, so Equation 1 is valid for $j = 2, 10, 18, 26$.

In the remainder of this section, N_F denotes the total number of distinct values X_i that satisfy the relationship described in Equation 1 for a particular component F. The set containing these input words X_i is denoted by Θ_F.

Additionally, $N_{F,j}$ denotes the number of individual values that satisfy this relationship for a specific rotation value j. Note that the sum $\sum_{j=0}^{31} N_{F,j}$ may be higher than N_F because some input words satisfy this relationship for multiple values of j. For instance, in the previous example, i.e. $F($`02020202`$) =$ `08080808`, the input word `02020202` is counted four times.

Non-linear Transformation *S*. Recall that the non-linear transformation S consists of the application of a single 8×8 S-box s, applied four times in parallel. By reverse engineering, Liu et al. [12] managed to deduce how the S-box for SMS4 is constructed. They found that the S-box s uses an inversion in the finite field, which is similar to that of the AES. Note that the design of the S-box for the AES explicitly avoids fixed points [6]. However, we identified one fixed point in s. The fixed point is the 8-bit value `AB` (in hexadecimal). Thus, the non-linear transformation S also has a fixed point, which is the hexadecimal value `ABABABAB`.

In addition to this fixed point, there also exist other input words X_i that satisfy the relationship $S(X_i) = X_i \lll j$ for some $j > 0$. For these particular input words, the transformation S is basically linear. There are $N_S = 39$ (including the fixed point) distinct input words X_i that have a relationship of this form. Let Θ_S denote the set containing the exact values of these X_i, which are given in Table 3 in the Appendix. The number, $N_{S,j}$, of values that satisfy this relationship for S, for each rotation value j is given in Table 1.

Linear Transformation *L*. We identified four fixed points ($j = 0$) and 1020 other ($j > 0$) input words X_i that satisfy the relationship $L(X_i) = X_i \lll j$, i.e. $N_L = 1024$. Let Θ_L denote the set containing the exact values of these X_i. For these input words, the linear transformation L provides poor diffusion because the input bits of these words are not well scattered by L when producing the output words. The number, $N_{L,j}$, of values that satisfy this relationship for L, for each rotation value j is given in Table 1.

Function *T*. As a non-linear cryptographic component, the function T of SMS4 should behave like a random permutation. The probability that a given permutation of n elements has c fixed points is given by [15, Chap. 3]

$$p_{n,c} = \frac{1}{n!} \cdot \binom{n}{c} \cdot (n-c)! \cdot \sum_{k=0}^{n-c} \frac{(-1)^k}{k!} \approx \frac{1}{c!e}.$$

For both $c = 0$ and $c = 1$, as n tends to infinity, the probabilities $p_{n,0}$ and $p_{n,1}$ approach $e^{-1} = 0.3679$. Therefore, the number of permutations having at least 2 fixed points is approximately $1 - 2(0.3679) = 0.2642$. Note that the expected number of fixed points for a random permutation is one [9, Chap. 6].

By exhaustive search, we found 11 fixed points in the function T of SMS4, i.e. values X_i such that $T(X_i) = X_i$ (for $j = 0$). The fixed points are `0B0B0B0B`, `3E973E97`, `3AE2C6AD`, `62D367B9`, `973E973E`, `E2C6AD3A`, `D367B962`, `C6AD3AE2`, `67B962D3`, `AD3AE2C6` and `B962D367`. For a random permutation, the probability

Table 1. Number of output words which are equivalent to the rotation of the input word by j bits to the left ($0 \leq j \leq 31$), for each component function

j	$N_{S,j}$	$N_{L,j}$	$N_{T,j}$	$N_{L',j}$	$N_{T',j}$
0	1	4	11	4	0
1	16	2	4	2	6
2	7	1024	7	8	8
3	0	2	2	2	4
4	1	4	1	4	5
5	3	2	0	2	3
6	1	16	1	8	6
7	0	2	1	2	3
8	3	4	1	4	2
9	2	2	4	2	2
10	1	256	1	8	4
11	0	2	2	2	4
12	1	4	1	4	3
13	1	2	2	2	1
14	1	16	5	8	6
15	0	2	7	2	1
16	9	4	3	4	0
17	4	2	4	2	2
18	3	1024	3	8	4
19	0	2	2	2	4
20	1	4	1	4	5
21	3	2	0	2	3
22	1	16	1	8	2
23	0	2	1	2	7
24	3	4	1	4	2
25	2	2	0	2	2
26	1	256	9	8	4
27	0	2	2	2	12
28	1	4	1	4	3
29	1	2	2	2	1
30	1	16	1	8	6
31	0	2	11	2	1

of having 11 fixed points is approximately $p_{n,11} = 1/(11! \cdot e) \approx 9.216\text{E}-9$, which is quite low. Interestingly, if the S-box of SMS4 is replaced by the S-box of the AES, there are no fixed points in the resulting function T.

Similarly, there exist input words X_i that satisfy the relationship $T(X_i) = X_i \lll j$ for $j > 0$. In total, there are $N_T = 59$ distinct input words X_i (including the fixed points) that satisfy this relationship. Let Θ_T denote the set containing the exact values of these X_i, which are given in Table 4 in the Appendix. The number $N_{T,j}$ for each value of j is given in Table 1.

Recall that the function T is composed of S and L, i.e. $T = L \circ S$. The 39 input words contained in the set Θ_S do not all appear in the set Θ_T. However, there are seven input words that appear in the intersection of these two sets, $\Theta_S \cap \Theta_T$. These input words are `0A0A0A0A`, `0B0B0B0B`, `21212121`, `26262626`, `ABABABAB`, `E7E7E7E7` and `FAFAFAFA`.

Linear Transformation L'. We found, by exhaustive search, that there are no fixed points for L'. However, we found $N'_L = 8$ distinct input words X_i that satisfy the relationship $L'(X_i) = X_i \lll j$ for some $j > 0$. Let $\Theta_{L'}$ denote the set containing the exact values of these X_i. As a linear transformation, the diffusion provided by L' is poor for these input words. Note that the size of the set $\Theta_{L'}$ is smaller than the size of Θ_L, despite the fact that L' has fewer rotations than L. The number $N_{L',j}$ of values for each rotation value j is given in Table 1.

Function T'. Unlike the function T, the function T' has no fixed points. However, there still exist some input words X_i that satisfy the relationship $T'(X_i) = X_i \lll j$ for some $j > 0$. In total, there are $N_{T'} = 59$ distinct input words X_i that satisfy this relationship. Let $\Theta_{T'}$ denote the set containing the exact values of these X_i, which are given in Table 5 in the Appendix. The number $N_{T',j}$ for each value of j is given in Table 1.

Recall that the function T' is composed of S and L', i.e. $T' = L' \circ S$. The number $N_{T'}$ of input words in the set $\Theta_{T'}$ is about 7 times more than the same number for $\Theta_{L'}$, and 20 more than Θ_S. Unlike the function T, the input words contained in the set Θ_S do not appear at all in the set $\Theta_{T'}$, i.e. $\Theta_S \cap \Theta_{T'} = \emptyset$. However, there exist a set of input words for which the functions T and T' produce the same output words. This relationship is discussed in the following section.

3.2 Relationship between T and T'

As noted in Section 2, the encryption and the key scheduling algorithms are nearly identical, differing only in the linear transformation. We identified eight input words for which the transformation L and L' produce the same output words, i.e. $L(Y_i) = L'(Y_i)$. These input words Y_i are `00000000`, `33333333`, `55555555`, `66666666`, `99999999`, `AAAAAAAA`, `CCCCCCCC` and `FFFFFFFF`.

Recall that the non-linear transformation S is the same in both the functions T and T'. If there exist some input words Y_i such that $L(Y_i) = L'(Y_i)$, then there exist words $X_i = S^{-1}(Y_i)$ such that $T(X_i) = L(S(X_i)) = L'(S(X_i)) = T'(X_i)$. The eight input words X_i are `71717171`, `28282828`, `97979797`, `A5A5A5A5`, `1F1F1F1F`, `18181818`, `04040404` and `B9B9B9B9`.

3.3 On the Branch Number of L'

A commonly used measure of diffusion for Substitution-Permutation-Network (SPN) block ciphers is the notion of the branch number [6]. For an SPN cipher, this number denotes the minimum number of active S-boxes for any two consecutive rounds. However, in the context of a generic Feistel cipher such as SMS4, this is not always true. Therefore, the branch number of a linear transformation L, denoted $\mathcal{B}(L)$, can be defined as the minimum number of non-zero subword differences for any input and output pair of L. If the input word to L is partitioned into m sub-words, then the optimal branch number for L is $\mathcal{B}(L) = m + 1$ [6].

The branch number is calculated as follows. Let $X_i = (X_{i,0}, X_{i,1}, \ldots, X_{i,m-1})$ denote a mb-bit word formed from the concatenation of m b-bit words. Let $\Gamma_{X_i} = \Gamma_{X_{i,0}} \Gamma_{X_{i,1}} \ldots \Gamma_{X_{i,m-1}}$ denote a binary vector of length m where $\Gamma_{X_{i,j}} = 1$ if $X_{i,j}$ is nonzero and $\Gamma_{X_{i,j}} = 0$ otherwise. Let $wt(\Gamma_{X_i})$ denote the Hamming weight (i.e. the number of non-zero bits) of Γ_{X_i} . The branch number of L, denoted $\mathcal{B}(L)$, is defined as

$$\mathcal{B}(L) = \min\{wt(\Gamma_{X_i}) + wt(\Gamma_{Y_i}) : X_i \neq 0 \text{ and } Y_i = L(X_i)\}.$$

For SMS4, the input word to both L and L' is partitioned into $m = 4$ subwords. Therefore, the optimal branch number for both L and L' is 5. Zhang et al.

Table 2. The input-output pattern distribution of L'

Γ_{X_i}	Γ_{Y_i}															
	0	1	2	4	8	3	5	6	9	A	C	7	B	D	E	F
0	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	.	.	.	.	.	.	.	.	.	.	.	1	3	31	.	220
2	.	.	.	.	.	.	.	.	.	.	.	3	31	.	1	220
4	.	.	.	.	.	.	.	.	.	.	.	31	.	1	3	220
8	.	.	.	.	.	.	.	.	.	.	.	.	1	3	15	236
3	.	.	.	.	.	7	1	1	3	1	.	242	210	220	252	n_{22}
5	.	.	.	.	.	1	3	1	1	.	1	218	250	218	250	n_{21}
6	.	.	.	.	.	3	1	7	.	1	1	210	220	252	242	n_{22}
9	.	.	.	.	.	1	1	.	7	1	3	380	370	338	236	n_0
A	.	.	.	.	.	1	.	.	1	3	1	251	218	250	235	n_{19}
C	.	.	.	.	.	.	1	1	1	1	7	222	252	242	228	n_{20}
7	.	1	3	1	1	240	248	242	249	249	251	n_{20}	n_{18}	n_{16}	n_{11}	n_{23}
B	.	1	.	.	1	245	252	254	242	249	250	n_1	n_5	n_4	n_{12}	n_{29}
D	.	.	.	1	1	253	249	252	245	252	242	n_3	n_2	n_5	n_{13}	n_{30}
E	.	.	1	3	.	250	250	243	253	249	243	n_{17}	n_{14}	n_9	n_{15}	n_{24}
F	.	253	251	250	252	n_8	n_6	n_8	n_7	n_6	n_{10}	n_{28}	n_{27}	n_{26}	n_{25}	n_{31}

[18] showed that the branch number of L is indeed optimal, and noted that the number of rotations and XOR operations used in L are the minimum needed to reach this optimal branch number. However, they did not investigate the branch number for L'. We determine the branch number for L' using a computer program and by observing the input-output pattern distribution table defined as follows.

Let both Γ_{X_i} and Γ_{Y_i} denote binary vectors of length $m = 4$. Furthermore, let $W[\Gamma_{X_i}][\Gamma_{Y_i}]$ denote the Γ_{X_i}-th row and Γ_{Y_i}-th column entry for the input-output pattern distribution table. The entries for this table are computed as follows. Initialize the counter W to all-zero. For every input $X_i = 0, 1, \ldots, 2^{32}-1$, calculate the output $Y_i = L'(X_i)$ and increment the counter $W[\Gamma_{X_i}][\Gamma_{Y_i}]$. The resulting table for L' is given by Table 2 where the entry '.' denotes zero, for simplicity. Due to size constraints, some values are denoted by n_i given as follows.

$$\begin{array}{lllll}
n_0 = 63688, & n_7 = 64023, & n_{14} = 64049, & n_{21} = 64082, & n_{28} = 16323877, \\
n_1 = 63894, & n_8 = 64024, & n_{15} = 64050, & n_{22} = 64088, & n_{29} = 16324086, \\
n_2 = 63895, & n_9 = 64025, & n_{16} = 64051, & n_{23} = 16323681, & n_{30} = 16324087, \\
n_3 = 63919, & n_{10} = 64026, & n_{17} = 64057, & n_{24} = 16323702, & n_{31} = 4229286763. \\
n_4 = 63930, & n_{11} = 64027, & n_{18} = 64061, & n_{25} = 16323764, & \\
n_5 = 63939, & n_{12} = 64032, & n_{19} = 64065, & n_{26} = 16323875, & \\
n_6 = 64019, & n_{13} = 64040, & n_{20} = 64070, & n_{27} = 16323876, &
\end{array}$$

The branch number of L' can be determined by first searching in Table 2 for a non-zero entry $W[\Gamma_{X_i}][\Gamma_{Y_i}]$ with $\Gamma_{X_i} \neq 0$ for which the sum of the Hamming weight for Γ_{X_i} and Γ_{Y_i} is the lowest among other entries. Then, the branch number is calculated as $\mathcal{B}(L') = wt(\Gamma_{X_i}) + wt(\Gamma_{Y_i})$. An example of such an entry is $W[\mathtt{1}][\mathtt{7}]$ and thus, the branch number of L' is $\mathcal{B}(L') = wt(\mathtt{1}) + wt(\mathtt{7}) = 1+3 = 4$, which is not optimal.

The input-output pattern distribution table also gives information regarding possible and impossible subword difference paths propagated by L'. This is useful for differential-type attacks. The sub-optimal branch number for L' is an indication of a potential weakness. This is exploited in Section 5 in a differential attack on a slightly modified variant of SMS4.

4 Cryptographic Significance

This section discusses the cryptographic significance of the observations made in Section 3.

4.1 Implications for the Key Scheduling Algorithm

The length of the master key for SMS4 is 128 bits, hence there are 2^{128} possible values of the master key. The key scheduling algorithm produces 32 subkeys, each of 32 bits, thus the sequence of subkeys forms a $32 \times 32 = 1024$-bit binary sequence. Clearly, there are extremely many sequences of subkeys that are impossible.

Note that the function T', which is a 32-bit to 32-bit map, is bijective (using the theorem provided by Zhang et al. [18]). In every round, the value of a single 32-bit word is updated using the output of T', a function which takes the other three 32-bit words as input. After four rounds, all 128 bits of the master key are completely updated by the round functions. Therefore, we can reasonably conjecture that all possible values of the first four subkeys are equally likely to occur (statistically independent), whereas the values for the remaining 28 subkeys are determined entirely by these four subkeys. This conjecture allows us to make the following claim.

We know from Section 3.1 that there are 59 distinct words X_i contained in the set $\Theta_{T'}$. Recall that the value of the master key after the initialization phase is partitioned into four 32-bit words $(K_{-4}, K_{-3}, K_{-2}, K_{-1})$ and the i-th round constant is denoted by CK_i. If the input words to the first four consecutive functions T' of the key scheduling algorithm are in the set $\Theta_{T'}$, then the first four subkeys consist of merely linear combinations of the master key[1]. This event is illustrated as follows. If $(K_{-3} \oplus K_{-2} \oplus K_{-1} \oplus CK_0) \in \Theta_{T'}$, then

$$K_0 = K_{-4} \oplus [(K_{-3} \oplus K_{-2} \oplus K_{-1} \oplus CK_0) \lll j_0].$$

Similarly, if $(K_{-2} \oplus K_{-1} \oplus K_0 \oplus CK_1) \in \Theta_{T'}$, then

$$K_1 = K_{-3} \oplus [(K_{-2} \oplus K_{-1} \oplus K_{-4} \oplus [(K_{-3} \oplus K_{-2} \oplus K_{-1} \oplus CK_0) \lll j_0] \oplus CK_1) \lll j_1].$$

[1] Note that the initialization phase does not have any cryptographic significance. Therefore, if we know the value of the resulting key after this phase, then we also know the value of the master key.

Furthermore, if $(K_{-1} \oplus K_0 \oplus K_1 \oplus CK_2) \in \Theta_{T'}$, then

$$\begin{aligned} K_2 = K_{-2} \oplus [(&K_{-1} \oplus K_{-4} \oplus [(K_{-3} \oplus K_{-2} \oplus K_{-1} \oplus CK_0) \lll j_0] \oplus \\ &K_{-3} \oplus [(K_{-2} \oplus K_{-1} \oplus K_{-4} \oplus [(K_{-3} \oplus K_{-2} \oplus K_{-1} \oplus CK_0) \lll j_0] \oplus \\ &CK_1) \lll j_1] \oplus CK_2) \lll j_2]. \end{aligned}$$

Finally, if $(K_0 \oplus K_1 \oplus K_2 \oplus CK_3) \in \Theta_{T'}$, then

$$\begin{aligned} K_3 = K_{-1} \oplus [(&K_{-4} \oplus [(K_{-3} \oplus K_{-2} \oplus K_{-1} \oplus CK_0) \lll j_0] \oplus \\ &K_{-3} \oplus [(K_{-2} \oplus K_{-1} \oplus K_{-4} \oplus [(K_{-3} \oplus K_{-2} \oplus K_{-1} \oplus CK_0) \lll j_0] \oplus \\ &CK_1) \lll j_1] \oplus \\ &K_{-2} \oplus [(K_{-1} \oplus K_{-4} \oplus [(K_{-3} \oplus K_{-2} \oplus K_{-1} \oplus CK_0) \lll j_0] \oplus \\ &K_{-3} \oplus [(K_{-2} \oplus K_{-1} \oplus K_{-4} \oplus [(K_{-3} \oplus K_{-2} \oplus K_{-1} \oplus CK_0) \lll j_0] \oplus \\ &CK_1) \lll j_1] \oplus CK_2) \lll j_2] \oplus CK_3) \lll j_3]. \end{aligned}$$

The above linear equations are valid for specific values of $j_i \in \{0, 1, \ldots, 31\}$. This event occurs with probability $(59/2^{32})^4 \approx 2^{-104.5}$ and thus, there are approximately $2^{23.5}$ values of the master key which cause such an event to happen.

4.2 Implications for the Encryption Algorithm

As noted in Section 3.1, there are 59 distinct words X_i contained in the set Θ_T. If the input words to the first four consecutive functions T of the encryption algorithm are in the set Θ_T, then the output block after four rounds consist of merely linear combinations of the plaintext block and subkeys. In general, this event is similar to that described in Section 4.1. Let us demonstrate the specific case in which only fixed points occur in the first four consecutive rounds. Let $\hat{\Theta}_T$ denote a subset of Θ_T containing the 11 fixed points for T (Refer to Section 3.1). This event is shown as follows for the plaintext block $P = (X_0, X_1, X_2, X_3)$ and subkeys K_0, K_1, K_2 and K_3. If $(X_1 \oplus X_2 \oplus X_3 \oplus K_0) \in \hat{\Theta}_T$ then

$$X_4 = X_0 \oplus X_1 \oplus X_2 \oplus X_3 \oplus K_0.$$

Similarly, if $(X_2 \oplus X_3 \oplus X_4 \oplus K_1) \in \hat{\Theta}_T$, then

$$X_5 = X_0 \oplus K_0 \oplus K_1. \tag{2}$$

Furthermore, if $(X_3 \oplus X_4 \oplus X_5 \oplus K_2) \in \hat{\Theta}_T$, then

$$X_6 = X_1 \oplus K_1 \oplus K_2. \tag{3}$$

Finally, if $(X_4 \oplus X_5 \oplus X_6 \oplus K_3) \in \hat{\Theta}_T$, then

$$X_7 = X_2 \oplus K_2 \oplus K_3. \tag{4}$$

Clearly, for the specific case of fixed points, the linear relationships above are much simpler than the general case because some words X_i and subkeys K_i cancel. This specific event occurs with probability $(11/2^{32})^4 \approx 2^{-114.2}$ and thus, there are approximately $2^{13.8}$ values of the plaintext block that cause such an event to happen for the full SMS4. In the general case, there are $2^{23.5}$ values of the plaintext block that cause the four-round linearity to happen.

4.3 Further Implications for Both the Key Scheduling and the Encryption Algorithms

The points discussed in Sections 4.1 and 4.2 have further security implications for SMS4. In the (admittedly rare) event that both the key scheduling and the encryption algorithms behave linearly for the first four rounds, the output block after four rounds of SMS4 is composed of merely linear combinations of the plaintext block and subkeys. The subkeys, in turn, are composed of linear combinations of the master key. Theoretically, if both of these events occur at the same time, then the number of effective rounds for SMS4 is reduced by four, from 32 to 28.

The above discussions only consider the case for which the linearity occurs in the key scheduling and the encryption algorithms in the first four consecutive rounds. Note that it may be possible for the linearity to occur in any four of the 32 rounds of SMS4. Furthermore, for certain particular combinations of plaintext block and master key, the linearity might possibly exist in more than four rounds. In this case, the number of effective rounds for SMS4 can be further reduced.

4.4 Susceptibility to Algebraic Attack

The algebraic attack [5] introduced by Courtois and Pieprzyk consists of building a system of binary equations that link the plaintext block, subkeys and ciphertext block. The binary equations describing an S-box that uses a finite field inversion, such as the AES and SMS4, are quadratic whereas the remaining equations are linear. The system is then solved to obtain the key bits. One of the obstacles in solving the system of equations for ciphers such as the AES and SMS4 is the existence of quadratic equations. The claimed advantage of this attack is that it only needs very few number of plaintext and ciphertext pairs.

As discussed in Sections 4.1, 4.2 and 4.3, there exist a few exceptional cases in which the non-linear functions T and T' are linear in the first four rounds of SMS4. Under these conditions, the binary equations describing the first four rounds are also entirely linear. Therefore, there is no need to describe the S-boxes in these rounds as systems of quadratic equations [12]. Since the occurrence of this event is statistical in nature, we may need more plaintext and ciphertext pairs compared to a conventional algebraic attack. However, the removal of some quadratic equations might help in reducing the complexity of solving the equation system.

4.5 Susceptibility to Advanced Variants of the Slide Attack

The slide attack was introduced by Biryukov and Wagner [3,4]. Given two different plaintexts, the attack permits the *sliding* of the two encryptions by a certain number of rounds. This is due to the similarity that exists between the structure of the two encryptions. The attack also allows the sliding of encryption with decryption [4].

We have shown in Section 3.2 that there are eight input words for which the functions T and T' produce the same output words. This similarity might provide an avenue for advanced variants of the slide attack. However, it is an open problem to determine whether it is useful to slide the encryption algorithm with the key scheduling algorithm if both algorithms are nearly identical, as is the case for SMS4.

4.6 Subkeys and Related-Keys

As discussed in Section 4.1, we conjecture that all possible 32-bit subkey values of the first four rounds of SMS4 are equally likely to occur. This allows us to explore the relationship between subkeys in these rounds and subkeys in the subsequent rounds. One possible relationship is described as follows. If the first four 32-bit round subkeys are identical (that is $K_i = \hat{K}$ for $i = 0, 1, 2, 3$ where $\hat{K}$ denotes an arbitrary 32-bit value), then a total of 2^{32} (out of 2^{128}) master keys have the following forms: $K_{-1} = \hat{K} \oplus T'(\hat{K} \oplus CK_3)$, $K_{-2} = \hat{K} \oplus T'(K_{-1} \oplus CK_2)$, $K_{-3} = \hat{K} \oplus T'(K_{-2} \oplus K_{-1} \oplus \hat{K} \oplus CK_1)$ and $K_{-4} = \hat{K} \oplus T'(K_{-3} \oplus K_{-2} \oplus K_{-1} \oplus CK_0)$. If this event and the event discussed in Section 4.2 occur at the same time, then the subkeys that exist in Equations 2, 3 and 4 will cancel and the subkeys in the first four rounds will have no effect on the intermediate words X_5, X_6 and X_7.

Similarly, if the subkeys in the first four rounds are identical, then the subkeys in rounds four (K_4) and five (K_5) have the following form:

$$K_4 = \hat{K} \oplus T'(\hat{K} \oplus \hat{K} \oplus \hat{K} \oplus CK_4) = \hat{K} \oplus T'(\hat{K} \oplus CK_4)$$

Suppose that $K_4 = \hat{K}$, which implies that $K_4 = \hat{K} = \hat{K} \oplus T'(\hat{K} \oplus CK_4)$ and $T'(\hat{K} \oplus CK_4) = 0$. Since CK_4 is a known fixed round constant, only one value of $\hat{K}$ can satisfy this equation, that is $\hat{K} = CK_4 \oplus \texttt{71717171} = \texttt{1060FF4}$. Therefore, for all 2^{32} master keys that have the form $K_i = \hat{K}$ for $i \in \{0, 1, \ldots, 4\}$, only one master key satisfies the relationship $K_3 = K_4$. The remaining $2^{32} - 1$ master keys have the relationship $K_3 \neq K_4$. Stated differently, if we are given a sequence of subkeys containing five identical words $K_i = \hat{K}$ for $i \in \{0, 1, \ldots, 4\}$, and $K_i \neq$ `1060FF4`, then we know that the subkeys are not the first four subkeys derived from the SMS4 key scheduling algorithm. These kinds of relationships can be further investigated beyond the first four rounds by taking into consideration the relationship between the round constants. The algorithm to derive these constants is already given in Section 2.2. In a key recovery attack, if the attacker knows the relationship of the words in the master key beforehand, then guesses that are impossible can be skipped. This reduces the key space that the attacker needs to guess.

The previous single-key discussion may be extended to the related-key model. Related-key attacks [1,11] allow the attacker to choose the relationship between two different master keys but not the actual value of the keys. The relationship is chosen such that the round subkeys of the first master key are related in some way to the round subkeys of the second master key. Then, several (known or chosen) plaintexts are encrypted using these related master keys to obtain the corresponding ciphertexts. The ciphertexts are then used to recover both master keys. This is an area for further investigation.

5 A Differential Attack on Modified SMS4

This section presents a differential attack [2] on a modified variant of SMS4, created by replacing the linear transformation L in the encryption algorithm with L'. This basically means that we are attacking the key scheduling algorithm, if it was used for encryption. We demonstrate that a differential attack is possible on a 27-round version of this variant.

5.1 23-Round Characteristic

We use a 5-round self-iterating differential characteristic based on previous differential attacks on SMS4 [10,18,19]. The characteristics used in these attacks have six active S-boxes: three in the fourth round and three in the fifth. Based on the entries of the input-output pattern distribution of L' given in Table 2, we know that there exist a number of differential paths where only two S-boxes are active in one round. An example of such a path is the entry $W[\mathtt{3}][\mathtt{3}]$.

Let $\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ denote a 32-bit difference formed from the concatenation of four 8-bit differences α_i. The 5-round self-iterating characteristic satisfies $0 \xrightarrow{T} 0$ in the first, second and third rounds; and $\alpha \xrightarrow{T} \alpha$ in the fourth and fifth rounds. This characteristic is given as follows: $(\alpha, \alpha, \alpha, 0) \rightarrow (\alpha, \alpha, 0, \alpha) \rightarrow (\alpha, 0, \alpha, \alpha) \rightarrow (0, \alpha, \alpha, \alpha) \rightarrow (\alpha, \alpha, \alpha, \alpha) \rightarrow (\alpha, \alpha, \alpha, 0)$.

By exhaustive search, we found six values of α that satisfy the above 5-round self-iterating characteristic such that only two bytes of α are nonzero (i.e. two bytes are active). The values are `0000900C`, `00C900C9`, `00900C00`, `0C000090`, `900C0000` and `C900C900`. The probability that $\alpha \xrightarrow{T} \alpha$ for each of these values is 2^{-14}. The probability for the 5-round self-iterating characteristics is therefore $(2^{-14})^2 = 2^{-28}$. This characteristic can be concatenated four and a half times to produce a 23-round differential characteristic with total probability $(2^{-28})^4 = 2^{-112}$ given below.

$$(\alpha, \alpha, \alpha, 0) \xrightarrow{\text{5 Rounds}} (\alpha, \alpha, \alpha, 0) \xrightarrow{\text{5 Rounds}} (\alpha, \alpha, \alpha, 0) \xrightarrow{\text{5 Rounds}}$$
$$(\alpha, \alpha, \alpha, 0) \xrightarrow{\text{5 Rounds}} (\alpha, \alpha, \alpha, 0) \xrightarrow{\text{3 Rounds}} (0, \alpha, \alpha, \alpha)$$

In comparison, the best 5-round differential characteristic on the original SMS4 has probability 2^{-38} and can only be concatenated up to three and a half times (to construct a 18-round differential characteristic) with total probability 2^{-114} [18].

5.2 27-Round Key Recovery Attack

The previous 23-round differential characteristic can be used in a 27-round key recovery attack on the modified variant of SMS4. Since the attack is heavily based on previous differential attacks [10,18,19], we only briefly describe the attack.

Choose $\alpha = (\mathtt{00,00,90,0C})$ and let Λ be the set of all output differences of T' where only 2 S-boxes are active. For each S-box, there is only 127 possible output differences. Therefore, the set contains $127 \cdot 2 \approx 2^8$ possible values.

Let P and P^* denote a plaintext pair and let C and C^* denote the corresponding ciphertext pair after 27 rounds, where $P = (X_0, X_1, X_2, X_3)$, $P^* = (X_0^*, X_1^*, X_2^*, X_3^*)$, $C = (X_{27}, X_{28}, X_{29}, X_{30})$ and $C^* = (X_{27}^*, X_{28}^*, X_{29}^*, X_{30}^*)$. The attack proceeds as follows.

1. Generate $m \cdot (2^{16})^3 = m \cdot 2^{48}$ plaintext blocks where bytes 2, 3, 6, 7, 10 and 11 are set to all possible values whereas the remaining bytes are fixed. These propose $m \cdot 2^{48}/2 = m \cdot 2^{47}$ plaintext pairs (P, P^*) having the difference $(\alpha, \alpha, \alpha, 0)$.
2. Encrypt the plaintexts using 27 rounds of the modified SMS4.
3. Filter the ciphertexts so that we only choose $(X_{27} \oplus X_{27}^*) \in \Lambda$. This filtering causes about $m \cdot 2^{47} \cdot 2^{-8} = m \cdot 2^{39}$ pairs to remain.
4. Let $\gamma_{i,j} = s(X_{i,j} \oplus X_{i+1,j} \oplus X_{i+2,j} \oplus K_{i-1,j}) \oplus s(X_{i,j}^* \oplus X_{i+1,j}^* \oplus X_{i+2,j}^* \oplus K_{i-1,j})$ and $\delta_{i,j} = L'(X_{i+3,j} \oplus X_{i+3,j}^* \oplus \alpha_j)$.
5. For each round $i = 27, 26, 25$, do the following
   (a) For each byte $j = 0, 1, 2, 3$, do the following
      i. For each byte guess $K_{i-1,j} = \texttt{0}, \texttt{1}, \ldots, \texttt{FF}$, do the following
         A. Calculate $\gamma_{i,j}$ and $\delta_{i,j}$.
         B. If $\gamma_{i,j} = \delta_{i,j}$, then store $K_{i-1,j}$ as a possible correct candidate key byte.
      ii. After all values have been guessed for this byte, wrong pairs are expected to be discarded by a factor of 2^{-8}.
6. After Step (5), we have guessed 12 bytes of key material and about $m \cdot 2^{39} \cdot (2^{-8})^{12} = m \cdot 2^{-57}$ pairs are expected to remain.
7. For round $i = 24$, do the following
   (a) For each byte guess $K_{23,0} = \texttt{0}, \texttt{1}, \ldots, \texttt{FF}$, calculate $\gamma_{24,0}$ and $\delta_{24,0}$. If $\gamma_{24,0} = \delta_{24,0}$, then store $K_{23,0}$ as a possible correct candidate key byte.
   (b) After all values have been guessed for this byte, wrong pairs are expected to be discarded by a factor of 2^{-8}.
8. After Step (7), about $m \cdot 2^{-57} \cdot (2^{-8}) = m \cdot 2^{-65}$ pairs are expected to remain. If $m = 2^{68}$, then for a wrong key guess, the expected number of remaining ciphertext pairs is approximately $2^{68} \cdot 2^{-65} = 2^3 = 8$. However, for a right key guess, the expected number of remaining ciphertext pairs is approximately $2^{68} \cdot 2^{48} \cdot 2^{-112} = 2^4 = 16$.
9. If the guesses for $K_{23,0}$, K_{24}, K_{25} and K_{26} suggest more than 16 remaining ciphertext pairs, then the guesses are candidates for correct subkeys.

The data complexity of this 27-round attack is $2^{68} \cdot 2^{48} = 2^{116}$ chosen plaintexts. The time complexity of the attack is dominated by Steps (5) and (7). At the beginning of Step (5), there are about $2^{68} \cdot 2^{39}$ pairs of texts. We guess 12 bytes of key material and for each guess, wrong pairs are discarded by a factor of 2^{-8}. At the beginning of Step (7), there are roughly $2^{68} \cdot 2^{-57}$ pairs of texts and we only guess one byte of key material. Adding these two complexities together, we obtained the time complexity of approximately $(\sum_{k=0}^{11} 2^8 \cdot 2^{68} \cdot 2^{39} \cdot 2^{-8k}) + 2^8 \cdot 2^{68} \cdot 2^{-57} \approx 2^{115}$ encryptions. In contrast, the best existing cryptanalysis on the original SMS4 is a differential attack on 22 rounds with a data complexity of 2^{117} chosen plaintexts and time complexity of $2^{112.3}$ 22-round encryptions [18].

5.3 Comments on the Security of SMS4

As mentioned at the beginning of Section 5, the attack described above is the same as attacking the key scheduling algorithm, as if it was used for encryption. We use the original components of the SMS4 and did not modify the function of these components. The key scheduling algorithm might therefore be exploited in related-key differential attacks.

In the light of our discussion in Section 4.3, there is a small possibility that the first four rounds of SMS4 is deprived of non-linearity. Under these conditions, the number of effective rounds for SMS4 is theoretically reduced by four, from 32 to 28. In this section, we have demonstrated an attack against 27 rounds of a slightly modified variant of SMS4. This is only one round short of the effective 28 rounds. Note that the four-round linearity event discussed in Section 4.3 refers to the event in which the function T was used in the encryption, instead of T', as is the case here. However, if T' was used in the encryption, the probability of this event to occur for T', in the general case, is the same as if T was used in the encryption. This is because the number of input words in the set Θ_T is the same as the set $\Theta_{T'}$.

Recall that the best attack on the original SMS4 is on 22-rounds [18], which is six rounds short of the effective 28 rounds. However, note that the security margin is reduced from 32 to 28 rounds only if the linearity in the first four rounds can be detected and utilized in an attack. A method to detect this remains an open problem.

6 Summary and Conclusion

This paper presents several new observations on both the encryption and the key scheduling algorithms of the SMS4 block cipher. We have shown the existence of fixed points and of simple linear relationships between the bits of the input and output words for each component of the round functions for some input words. Furthermore, we show that the branch number of the linear transformation in the key scheduling algorithm is less than optimal.

The major security implication of these observations is that the round function is not always non-linear. Due to this linearity, for some combinations of plaintext block and master key, the number of effective rounds of SMS4 is theoretically reduced by four, from 32 to 28. We also briefly explored the susceptibility of SMS4 against algebraic and advanced variants of the slide attacks.

Finally, we demonstrated that if the linear transformation L of the encryption algorithm is replaced with the linear transformation L' of the key scheduling algorithm, then this variant of SMS4 is weaker than the original SMS4 with regard to differential cryptanalysis. We show this by attacking four more rounds than the best existing differential attack on SMS4. This is possible due to the sub-optimal branch number of L'. This property of L' might be an indication of further weakness that can be exploited in an attack. We strongly believe that this variant is also weaker than SMS4 against other differential-type attacks.

Given the number of expected fixed points, it is unlikely that the components in the round functions are generated randomly, that is, they were selected specifically. However, the criteria for selecting the components are not known. The findings made in this paper raise serious questions on the security provided by SMS4, and might provide clues on the existence of a flaw in the design of the cipher.

References

1. 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)
2. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer, Heidelberg (1993)
3. Biryukov, A., Wagner, D.: Slide Attacks. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 245–259. Springer, Heidelberg (1999)
4. Biryukov, A., Wagner, D.: Advanced Slide Attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 589–606. Springer, Heidelberg (2000)
5. Courtois, N.T., Pieprzyk, J.: Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)
6. Daemen, J., Rijmen, V.: The Design of Rijndael, AES – The Advanced Encryption Standard. Springer, Heidelberg (2002)
7. Diffie, W., Ledin, G.: SMS4 Encryption Algorithm for Wireless Networks. In: Cryptology ePrint Archive, Report 2008/329 (2008)
8. Etrog, J., Robshaw, M.J.B.: Improved Cryptanalysis of Reduced-Round SMS4. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 51–65. Springer, Heidelberg (2009)
9. Grinstead, C.M., Snell, J.L.: Introduction to Probability, 2nd revised edn. American Mathematical Society, Providence (1997)
10. Kim, T., Kim, J., Hong, S., Sung, J.: Linear and Differential Cryptanalysis of Reduced SMS4 Block Cipher. In: Cryptology ePrint Archive, Report 2008/281 (2008)
11. Knudsen, L.: Cryptanalysis of LOKI91. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 22–35. Springer, Heidelberg (1993)
12. Liu, F., Ji, W., Hu, L., Ding, J., Lv, S., Pyshkin, A., Weinmann, R.-P.: Analysis of the SMS4 Block Cipher. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 158–170. Springer, Heidelberg (2007)
13. Lu, J.: Attacking Reduced-Round Versions of the SMS4 Block Cipher in the Chinese WAPI Standard. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 306–318. Springer, Heidelberg (2007)
14. Office of State Commercial Cryptography Administration, P.R. China: The SMS4 Block Cipher (2006) (in Chinese), http://www.oscca.gov.cn/UpFile/200621016423197990.pdf
15. Riordan, J.: An Introduction to Combinatorial Analysis. Princeton University Press, Princeton (1980)
16. Schneier, B., Kelsey, J.: Unbalanced Feistel Networks and Block Cipher Design. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 121–144. Springer, Heidelberg (1996)

17. Toz, D., Dunkelman, O.: Analysis of Two Attacks on Reduced-Round Versions of the SMS4. In: Chen, L., Ryan, M.D., Wang, G. (eds.) ICICS 2008. LNCS, vol. 5308, pp. 141–156. Springer, Heidelberg (2008)
18. Zhang, W., Wu, W., Feng, D., Su, B.: Some New Observations on the SMS4 Block Cipher in the Chinese WAPI Standard. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 324–335. Springer, Heidelberg (2009)
19. Zhang, L., Zhang, W., Wu, W.: Cryptanalysis of Reduced-Round SMS4 Block Cipher. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 216–229. Springer, Heidelberg (2008)

A Appendix

Table 3. Values of X_i (in the set Θ_S) and j such that $S(X_i) = X_i \lll j$

X_i	j	X_i	j	X_i	j
`0A0A0A0A`	1, 9, 17, 25	`21210A0A`	1	`ABB4ABDE`	16
`0A0A0A21`	1	`21210A21`	1	`ABDEABB4`	16
`0A0A210A`	1	`2121210A`	1	`B4ABDEAB`	16
`0A0A2121`	1	`21212121`	1, 9, 17, 25	`B4B4DEDE`	16
`0A210A0A`	1	`245C245C`	2, 18	`B4DEB4DE`	8, 24
`0A210A21`	1, 17	`245C2626`	2	`B4DEDEB4`	16
`0A21210A`	1	`26245C26`	2	`D056D056`	5, 21
`0A212121`	1	`2626245C`	2	`DEABB4AB`	16
`0B0B0B0B`	6, 14, 22, 30	`26262626`	2, 10, 18, 26	`DEB4B4DE`	16
`210A0A0A`	1	`56D056D0`	5, 21	`DEB4DEB4`	8, 24
`210A0A21`	1	`5C245C24`	2, 18	`DEDEB4B4`	16
`210A210A`	1, 17	`5C262624`	2	`E7E7E7E7`	4, 12, 20, 28
`210A2121`	1	`ABABABAB`	0, 8, 16, 24	`FAFAFAFA`	5, 13, 21, 29

Table 4. Values of X_i (in the set Θ_T) and j such that $T(X_i) = X_i \lll j$

X_i	j	X_i	j	X_i	j
02740274	2, 18	4F13E4B4	2	BB06C4A3	26
039A039A	1, 17	58434DF7	26	BE6CBE6C	15, 31
06C4A3BB	26	5CDE9B16	14	C4A3BB06	26
0A0A0A0A	3, 11, 19, 27	62D367B9	0	C6AD3AE2	0
0B0B0B0B	0, 8, 16, 24	67B962D3	0	C7E7C7E7	13, 29
1079D3A1	31	6CBE6CBE	15, 31	D367B962	0
13E4B44F	2	74027402	2, 18	D3A11079	31
165CDE9B	14	79D3A110	31	DE9B165C	14
16AF4D4B	15	973E973E	0, 16	E0E1F7E3	9
1A2A1A2A	1, 17	9A039A03	1, 17	E1F7E3E0	9
21212121	3, 11, 19, 27	9B165CDE	14	E2C6AD3A	0
22E59CB6	31	9CB622E5	31	E3E0E1F7	9
26262626	4, 12, 20, 28	A11079D3	31	E4B44F13	2
2A1A2A1A	1, 17	A3BB06C4	26	E59CB622	31
3AE2C6AD	0	ABABABAB	2, 10, 18, 26	E7C7E7C7	13, 29
3E973E97	0, 16	AD3AE2C6	0	E7E7E7E7	6, 14, 22, 30
434DF758	26	AF4D4B16	15	F758434D	26
4B16AF4D	15	B44F13E4	2	F7E3E0E1	9
4D4B16AF	15	B622E59C	31	FAFAFAFA	7, 15, 23, 31
4DF75843	26	B962D367	0		

Table 5. Values of X_i (in the set $\Theta_{T'}$) and j such that $T'(X_i) = X_i \lll j$

X_i	j	X_i	j	X_i	j
02020202	4, 12, 20, 28	5228B69C	6	A66BA66B	10, 26
06C206C2	1, 17	52505250	4, 20	AAA027D5	23
087B087B	4, 20	5522DB49	27	B0B0B0B0	3, 11, 19, 27
10B78569	2	58F758F7	8, 24	B69C5228	6
12121212	6, 14, 22, 30	5A5A5A5A	2, 10, 18, 26	B7856910	2
12161216	7, 23	61F161F1	14, 30	B8B8B8B8	2, 10, 18, 26
16121612	7, 23	64C164C1	9, 25	BAC74FDD	27
1B341B34	5, 21	6910B785	2	C164C164	9, 25
1D411D41	2, 18	6BA66BA6	10, 26	C206C206	1, 17
22DB4955	27	74747474	3, 11, 19, 27	C74FDDBA	27
25A498A2	1	7B087B08	4, 20	CBA1CBA1	14, 30
27D5AAA0	23	856910B7	2	D5AAA027	23
28B69C52	6	94949494	6, 14, 22, 30	D69AD69A	12, 28
32323232	3, 11, 19, 27	98A225A4	1	DB495522	27
341B341B	5, 21	9AD69AD6	12, 28	DDBAC74F	27
411D411D	2, 18	9C5228B6	6	DFDFDFDF	3, 11, 19, 27
495522DB	27	A027D5AA	23	E1E1E1E1	7, 15, 23, 31
4F4F4F4F	5, 13, 21, 29	A1CBA1CB	14, 30	F161F161	14, 30
4FDDBAC7	27	A225A498	1	F758F758	8, 24
50525052	4, 20	A498A225	1		

Algebraic Cryptanalysis of Curry and Flurry Using Correlated Messages

Jean-Charles Faugère and Ludovic Perret

SALSA Project
INRIA, Centre Paris-Rocquencourt
UPMC, Univ Paris 06, LIP6
CNRS, UMR 7606, LIP6
104, avenue du Président Kennedy
75016 Paris, France
`jean-charles.faugere@inria.fr, ludovic.perret@lip6.fr`

Abstract. In this paper, we present an algebraic attack against the **Flurry** and **Curry** block ciphers [12,13]. Usually, algebraic attacks against block ciphers only require *one* message/ciphertext pair to be mounted. In this paper, we investigate a different approach. Roughly, the idea is to generate an algebraic system from the knowledge of several well chosen correlated message/ciphertext pairs. **Flurry** and **Curry** are two families of ciphers which fully parametrizable and having a sound design strategy against the most common statistical attacks; i.e. linear and differential attacks. These ciphers are then targets of choices for algebraic attacks. It turns out that our new approach permits to go one step further in the (algebraic) cryptanalysis of difficult instances of **Flurry** and **Curry**. To explain the behavior of our attack, we have established an interesting connection between algebraic attacks and high order differential cryptanalysis [32]. From extensive experiments, we estimate that our approach – that we will call "algebraic-high order differential" cryptanalysis – is polynomial when the Sbox is a power function. As a proof of concept, we have been able to break **Flurry**/**Curry** – up to 8 rounds – in few hours. We have also investigated the more difficult (and interesting case) of the inverse function. For such function, we have not been able to bound precisely the theoretical complexity, but our experiments indicate that our approach permits to obtain a significant practical gain. We have attacked **Flurry**/**Curry** using the inverse Sbox up to 8 rounds.

1 Introduction

A fundamental problem in cryptography is to evaluate the security of widely used cryptosystems against the most powerful techniques. To this end, several *general* methods have been proposed : linear cryptanalysis [34], differential cryptanalysis [8,9,10], *etc* .... *Algebraic cryptanalysis* can be described as a general framework that permits to asses the security of a wide range of cryptographic schemes [4,15,16,17,26,27,28,29]. As pointed in [20] "*the recent proposal and development of algebraic cryptanalysis is now widely considered an important breakthrough in* *the analysis of cryptographic primitives.* It is a powerful technique that applies potentially to a wide range of cryptosystems, in particular block ciphers.

The basic principle of such cryptanalysis is to model a cryptographic primitive by a set of algebraic equations. The system of equations is constructed in such a way as to have a correspondence between the solutions of this system, and a secret information of the cryptographic primitive (for instance, the secret key of a block cipher). This line of research is somehow inspired by C.E. Shannon who stated that: "*Breaking a good cipher should require as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type."*(Commu–nication Theory of Secrecy Systems, 1949). Shannon relates then the security of a cryptosystem to the difficulty of solving a set of algebraic equations, and lays then the foundation of algebraic cryptanalysis.

In theory, any cryptosystem can be modeled by a set of algebraic equations over a finite field [30]. In fact, it is usual that the same cryptographic primitive can be described by several algebraic systems. However, it is an open research problem how to optimally model a cryptosystem so that it is easiest to solve. Thus, it is one of the most crucial aspects of algebraic cryptanalysis to derive the best system with respect to equations solving.

Algebraic techniques have been successfully applied against a number of multivariate schemes and in stream cipher cryptanalysis [4,15,16,17,26,27,28,29]. On the other hand, its feasibility against block ciphers remains the source of speculations [19,21,33,4,3]. The main problem is that the size of the corresponding algebraic system is so huge (thousand of variables and equations) that nobody is able to predict correctly the complexity of solving such polynomial systems.

However, it is worth to remark that the algebraic systems are huge but highly structured. The equations are very sparse and the round structure of the block ciphers implies a similar structure on the algebraic equations. Secondly, there is not a unique algebraic description of a cryptographic primitive. Although it is an open issue how to optimally model a cryptosystem, it is crucial to use this degree of freedom to derive the best system with respect to equations solving.

Typically, algebraic cryptanalysis against block ciphers only requires *one* message/ciphertext pair to be mounted. In this paper, we present a novel approach. The basic idea is to generate an algebraic system from the knowledge of several well chosen correlated message/ciphertext pairs. It turns our that this system is easier to solve in practice. To explain this behavior, we have established an interesting connection between algebraic attacks and high order differential cryptanalysis [31,32].Interesting enough, this is on the line of a new trend in algebraic cryptanalysis which is to combine statistical and algebraic techniques. Albrecht and Cid [1,2] recently proposed to mix differential and algebraic cryptanalysis to attack PRESENT.

As already explained, the algebraic cryptanalysis of a block cipher usually leads to huge systems of equations. For this reason, it makes sense to first experiment such attacks on "scalable" block ciphers. For this reason, Cid, Murphy, and Robshaw [18] described small scale variants of AES. In the same vain, Buchmann, Pyshkin and Weinmann [12,13] described two families of Feistel (**Flurry**) and SPN (**Curry**) block ciphers which are fully parametrizable. The main goal of **Curry** and **Flurry** was probably to be relevant families of ciphers for experimenting algebraic attacks. To do so, the encryption process of these ciphers can be easily described by a set of algebraic equations. On the other hand, these ciphers have a sound design strategy against linear [34] and differential [8,9,10] attacks. The aim was to mimic as much as possible the design criteria of widely used modern block ciphers. Therefore, any new successful algebraic cryptanalysis against **Curry**/**Flurry** can be a step toward more efficient algebraic cryptanalysis against industrial block ciphers such as AES.

1.1 Organization of the Paper: Main Results

After this introduction, the paper is organized as follows. In Section 2, we introduce the families of Feistel and SPN block ciphers **Flurry** and **Curry** respectively [12,13]. We briefly recall some security features of the ciphers.

In this last section (Section 3), we will present results that we have obtained when mounting two refined algebraic attack strategies. First, we show that the use of a sparse version of FGLM [23] permits to obtain a practical gain w.r.t. to the attack presented by Buchmann, Pyshkin and Weinmann [12]. However, this attack remains limited since its theoretical complexity is exponential in the number of rounds and the size of the plaintext space.

To overcome this limitation, we have investigated the possibility of using a small amount of suitably chosen message/ciphertext pairs to improve the efficiency of algebraic attacks. Precisely, we propose to use correlated messages. It appears that this approach permits to go one step further in the (algebraic) cryptanalysis of of difficult instances of **Flurry** and **Curry**. To explain the behavior of our attack, we have established an interesting connection between algebraic attacks and high order differential cryptanalysis [32]. From extensive experiments, we estimate that our approach – that we will call "algebraic-high order differential" cryptanalysis – is polynomial when the Sbox is a power function. As a proof of concept, we have been able to break **Flurry**/**Curry** – up to 8 rounds – in few hours. We have also thoroughly investigated the inverse function. Although we have not been able to bound precisely the theoretical complexity, our experiments indicate that our approach permits to obtain a significant practical gain. We have been able to break **Flurry**/**Curry** using the inverse Sbox up to 8 rounds.

2 The Flurry and Curry Block Ciphers

In this part, we describe the main concern of this paper, namely **Curry** and **Flurry** [12,13]. We will briefly recall the algebraic description of such ciphers, and highlight some security features of these ciphers.

In the rest of this paper $\mathbb{K} = \mathbb{F}_2(\theta)$ will denote a finite field of size $k = 2^n, n \in \{8, 16, 32, 64\}$. The number of rounds will be denoted by $r \in \mathbb{N}$. $D \in \mathcal{M}_{m\times m}(\mathbb{K})$ is a matrix describing the linear diffusion mapping of the round function. This matrix D is also used in the key scheduling. We refer to [12] for the exact description of these matrices, but we mention that these matrices have been chosen to have an optimal diffusion. Finally, f is a non-linear function describing the Sbox chosen as the power function $f(x) = f_p(x) : x \in \mathbb{K} \mapsto x^p \in \mathbb{K}$, with $p \in \{3, 5, 7\}$, or the inverse function $f(x) = f_{\text{inv}}(x) : x \in \mathbb{K} \mapsto x^{k-2} \in \mathbb{K}$.

2.1 The Feistel Case: Flurry

First we describe the family of Feistel ciphers **Flurry**(n, t, r, f, D), with $t \in \mathbb{N}$ being the size of a message block. To add to this confusion, we will also use the notation $m = \frac{t}{2}$ for the half-size of a block (t is assumed to be even).

We will denote by $L = (\ell_1, \ldots, \ell_m) \in \mathbb{K}^m$ $\big($resp. $R = (r_1, \ldots, r_m) \in \mathbb{K}^m$ $\big)$ the left (resp. right) part of the current state, and by $K = (k_1, \ldots, k_m) \in \mathbb{K}^m$ a key. The round function $T : \mathbb{K}^m \times \mathbb{K}^m \times \mathbb{K}^m \rightarrow \mathbb{K}^m \times \mathbb{K}^m$ is then defined as :

$$T(L, R, K) = \big(R, \big(f(r_1 + k_1), \ldots, f(r_m + k_m)\big) \cdot D + L\big)\,.$$

Let $K = (K_0, K_1) \in \mathbb{K}^m \times \mathbb{K}^m$ be the initial key. The subkey used at round $i, 2 \leq i \leq r+1$ is :

$$K_i = K_{i-1} \cdot D + K_{i-2} + v_i,$$

where $v_i = \big((\theta+1)^i, (\theta+1)^{i+1}, \ldots, (\theta+1)^{i+m-1}\big) \in \mathbb{K}^m$.

A message $m = (L_0, R_0) \in \mathbb{K}^m \times \mathbb{K}^m$ is encrypted into a ciphertext $c = (L_r, R_r) \in \mathbb{K}^m \times \mathbb{K}^m$ by iterating the round function T as follows :

$$(L_i, R_i) = T(L_{i-1}, R_{i-1}, K_{i-1}), \text{ for all } i, 1 \leq i \leq r-1,$$
$$c = (L_r, R_r) = T(L_{r-1}, R_{r-1}, K_{r-1}) + (K_r, K_{r+1}).$$

It is not difficult to describe the encryption process by a set of algebraic equations. Actually, this was a design policy [12,13]. To keep the degree as low a possible, we have to introduce new variables :
- $\{x_{i,j}\}_{1\leq i \leq (r-1)}^{1 \leq j \leq t}$ corresponding to the internal states of the cipher,
- and $\{k_{i,j}\}_{1 \leq i \leq r+1}^{1 \leq j \leq m}$ corresponding to the initial/expanded key.

We will denote by $\mathcal{R}_{\text{Flurry}}$ the polynomial ring $\mathbb{K}\big[\{x_{i,j}\}_{1\leq i \leq (r-1)}^{1 \leq j \leq t}, \{k_{i,j}\}_{1 \leq i \leq r+1}^{1 \leq j \leq m}\big]$. For a pair plaintext/ciphertext $(m, c) \in \mathbb{K}^t \times \mathbb{K}^t$, we will denote by : $\mathcal{P}_{\text{Flurry}}(m, c) \subset \mathcal{R}_{\text{Flurry}}$, the set of all algebraic equations describing a **Flurry** encryption process.

2.2 The SPN Case: Curry

Curry(n, m, r, f, D) is a family of SPN ciphers which is also fully parametrizable. The plaintext, ciphertext and secret key spaces are space dimension is $m \in \mathbb{N}$ are $\mathbb{K}^{m \times m}$. The round function $T : \mathbb{K}^{m\times m} \times \mathbb{K}^{m \times m} \rightarrow \mathbb{K}^{m \times m}$ of **Curry** is given by :

$$T(S, K) = G(S, K) \cdot D,$$

with $G : X = \{x_{i,j}\} \in \mathbb{K}^{m \times m} \rightarrow G(X) = \{f(x_{i,j})\} \in \mathbb{K}^{m\times m}$ being the parallel application of the SBox function f to the components of a matrix $X \in \mathbb{K}^{m \times m}$.

A plaintext $m = S_0 \in \mathbb{K}^{m\times m}$ is transformed into a ciphertext $c \in \mathbb{K}^{m\times m}$ by iterating the round function T exactly r times followed by a last key addition :

$$S_\ell = T(S_{\ell-1}, K_{\ell-1}),\ \text{for all}\ \ell, 1 \le \ell \le r-1,$$
$$c = S_r = T(S_{r-1}, K_{r-1}) + K_r.$$

The master key $K_0 \in K^{m\times m}$ is used at the first round; subsequent round keys $K_i, i \ge 1$ are computed using the formula :

$$K_i = K_{i-1} \cdot D + M_i,$$

with $M_i = \{\theta^{i+(j-1)m+k}\}_{1\le j,k\le m} \in \mathbb{K}^{m\times m}$ being a (constant) matrix depending of the round.

As for **Flurry**, there is no problem to construct a set of algebraic equations modeling the encryption process of **Curry**. We have to introduce new variables : $\{x_{i,j}^{\ell}\}_{1\le i,j\le m}^{1\le \ell\le (r-1)}$ corresponding to the internal states of the cipher, and $\{k_{i,j}^{\ell}\}_{1\le \ell\le r}^{1\le i,j\le m}$ corresponding to the initial/expanded key.

Using an obvious notation, $\mathcal{R}_{\text{Curry}}$ will denote $\mathbb{K}\left[\{x_{i,j}^{\ell}\}_{1\le i,j\le m}^{1\le \ell\le (r-1)}, \{k_{i,j}^{\ell}\}_{1\le \ell\le r}^{1\le i,j\le m}\right]$, and $\mathcal{P}_{\text{Curry}}(m,c) \subset \mathcal{R}_{\text{Curry}}$, the set of algebraic equations describing **Curry** $\big($for a plaintext/ciphertext $(m,c)\big)$.

3 Improved Algebraic Attacks against Curry and Flurry

This section is divided into two parts. First, we show that the Buchmann, Pyshkin and Weinmann (BPW) attack against **Curry** and **Flurry** [12] can be improved using a "fast version" of FGLM. The goal is to illustrate the gain that we can obtain using sparse algebra techniques. This will permit to have a tight complexity estimates of the BPW attack, and then a good basis to comparison with others attacks. In particular with respect to the practical behavior of the new attack presented in the second part of this section.

3.1 Practical Improvements of the Buchmann, Pyshkin and Weinmann Attack

We start by recalling a surprising result of to Buchmann, Pyshkin and Weinmann [12,13]. They proved that for a well chosen ordering $\prec^*$, the polynomials of $\mathcal{P}_{\text{Flurry}}$ and $\mathcal{P}_{\text{Curry}}$ already form a Gröbner basis [22,11]. It has to be noted that a similar result holds for AES-128 [14]. The key-recovery problem is then reduced to changing the order of a Gröbner basis. More precisely, solving $\mathcal{P}_{\text{Flurry}}$ (resp. $\mathcal{P}_{\text{Curry}}$) is equivalent to compute a Lex-Gröbner basis knowing a $\prec^*$–Gröbner basis. This can be done by using FGLM [23], and a precise complexity estimates can be then given [23,12,13].

Lemma 1. *Let $\mathcal{I}_{\text{Flurry}}$ (resp. $\mathcal{I}_{\text{Curry}}$) be the ideal generated by the polynomials of $\mathcal{P}_{\text{Flurry}}$ (resp. $\mathcal{P}_{\text{Curry}}$). It holds that :*

$$\dim_{\mathbb{K}}\left(\mathcal{R}_{\text{Flurry}}/\mathcal{I}_{\text{Flurry}}\right) = \deg(f)^{t\cdot r},\ \textit{for}\ \textbf{\textit{Flurry}}(n,t,r,f,D)$$
$$\dim_{\mathbb{K}}\left(\mathcal{R}_{\text{Curry}}/\mathcal{I}_{\text{Flurry}}\right) = \deg(f)^{m^2\cdot r},\textit{for}\ \textbf{\textit{Curry}}(n,m,r,f,D).$$

These results are not valid if f is the inverse function [12].

The complexity of FGLM is polynomial in the dimension of the quotient. We can use sparse linear algebra [35] techniques to decrease the exponent and obviously improve the efficiency of FGLM. To illustrate this fact, we will present experimental results. In the table below, we have quoted:
– the practical results obtained in [12,13] using FGLM. The authors have used the version available in Magma (version 2.11-8).
– the dimension $\dim_{\mathbb{K}}\big(\mathcal{R}/\mathcal{I}\big)$ of the quotient.
– The timings obtained with our sparse version of FGLM. This version of the algorithm has been implemented in C within the FGb software[1].

		[12]	This paper
Flurry(n,t,r,f,D)	$\dim_{\mathbb{K}}\big(\mathcal{R}/\mathcal{I}\big)$	F$_4$+FGLM (Magma)	"fast" FGLM (Fgb)
Flurry$(64,2,4,f_3,I_1)$	3^4	<0.1 s.	<0.1 s.
Flurry$(64,2,4,f_5,I_1)$	5^4	2.3 s.	<0.1 s.
Flurry$(64,2,4,f_7,I_1)$	7^4	82.62 s.	19.4 s.
Flurry$(64,2,6,f_3,I_1)$	3^6	145.08 s.	2.1 s.

Interpretation of the Results. We observe that there is a non-negligible practical gain when using a sparse version of FGLM. Anyway, this approach becomes quickly impractical due to huge dimension of $\mathcal{R}_{\text{Flurry}}/\mathcal{I}_{\text{Flurry}}$ (resp. $\mathcal{R}_{\text{Curry}}/\mathcal{I}_{\text{Curry}}$). This is mainly due to the fact that the field equations are not included in $\mathcal{P}_{\text{Flurry}}$ (resp. $\mathcal{P}_{\text{Curry}}$). Therefore, the variety associated with these systems will mostly contain spurious solutions (solutions over the algebraic closure of $\mathbb{K}$). However, this is to our knowledge the best (algebraic) attacks proposed so far against **Flurry** and **Curry**. We will now present an alternative approach for attacking these ciphers.

3.2 On the Use of Several Plaintext/Ciphertext Pairs

The key recovery systems $\mathcal{P}_{\text{Flurry}}$ and $\mathcal{P}_{\text{Curry}}$ are constructed from the knowledge of only one plaintext/ciphertext pair (Section 2). In this part, we investigate the possibility of using few pairs of plaintext/ciphertext. Namely, we select $N > 1$ messages $m_1,\ldots,m_N$ and request the corresponding ciphertexts $c_1,\ldots,c_N$. Instead of trying to solve each system $\mathcal{P}_{\text{Flurry}}(m_i,c_i)$ individually, the idea is to solve a new key recovery system: $\mathcal{P}_{\text{Flurry}}^N = \bigcup_{i=1}^N \mathcal{P}_{\text{Flurry}}(m_i,c_i)$. Obviously, the secret key will be also a solution of this larger system. The set of equations $\mathcal{P}_{\text{Curry}}^N$ is defined similarly.

Note that for each pair (m_i,c_i), we have to introduce new variables corresponding to the internal states of the cipher. On the other hand, the variables

[1] `http://fgbrs.lip6.fr/jcf/Software/FGb/index.html`

corresponding to the key will remain the same for each pair (m_i, c_i). Again, we emphasize that the field equations are not included in the systems. Remark that the result described previously (Sec. 3.1) for Flurry/Curry systems only holds when $N = 1$.

Using N random messages. Unfortunately, when we select *randomly* N messages $m_1, \dots, m_N$, this approach will not lead to any improvement w.r.t. to the use of a single plaintext/ciphertext pair. Even worse, we have observed that the new systems are harder to solve in practice. To illustrate this fact, we have quoted few results that obtained for **Flurry**$(8, 2, 2, D_2, f_{\text{inv}})$ with different values of N. The F_5 [25] and FGLM [23] algorithms have been implemented in C within the FGb software. We used this implementation for computing Gröbner bases. We have included the time T necessary for solving the systems and the total number of solutions Nb_{sol}.

N	1	2	3
T	0.43 s.	25.8 s.	16 m. 42 s.
Nb_{sol}	184	1	1

We have increased the number of equations/variables (corresponding to intermediate states), but the maximum degree reached during the Gröbner basis computation remains stable, i.e. the systems are not easier to solve. Interestingly enough, as soon as $N > 1$, the variety associated to $\mathcal{P}^N_{\text{Flurry}}$ (resp. $\mathcal{P}^N_{\text{Curry}}$) contains – most of the time – only one solution corresponding to the secret key; thus only one direct (DRL) Gröbner basis computation is needed.

3.3 Algebraic-High Order Differential Style Cryptanalysis

The difficulty is to find a suitable way to incorporate the additional knowledge of several message/ciphertext pairs. Or, stated differently, how to choose such pairs to improve the efficiency of the solving step. To do so, we have considered a classical cryptatanalytic tool, namely high order differentials [32,31], as a filter to select *correlated messages*. This will permit to decrease the complexity of the solving step. To explain the intuition behind our approach, we introduce few new definitions. The *derivative* (or *finite difference*) of a mapping $f : \mathbb{K}^n \mapsto \mathbb{K}^m$ at a point $r \in \mathbb{K}^n$ is defined as follows: $\Delta_r f(x) = f(x + r) - f(x)$. Remark that if the components of f are of an algebraic degree d, then the components of $\Delta_r f(x)$ will be of an algebraic degree $\leq d$. To further decrease this degree, we can consider the ith derivative of f at points $r_1, \dots, r_i$ which is recursively [32] defined as: $\Delta^{(i)}_{r_1,\dots,r_i} f(x) = \Delta_{r_i}\left(\Delta^{(i-1)}_{r_1,\dots,r_{i-1}} f(x)\right)$.

Now, let $L[r_1, \dots, r_i]$ be the set of all binary linear combinations of the $r_1, \dots, r_i$. It is well known that: $\Delta^{(i)}_{r_1,\dots,r_i} f(x) = \sum_{\delta \in L[r_1,\dots,r_i]} f(x + \delta)$. We can now explain how we have generated correlated the messages.

First, we will consider the most basic case, i.e. $N = 2$. We randomly select a message m_0, a difference r_1, and construct the key recovery system :

$$\mathcal{P}^2 = \mathcal{P}(m_0, c) \cup \mathcal{P}(m_0 + r_1, c'), \text{ with } \mathcal{P} \in \{\mathcal{P}_{\text{Flurry}}, \mathcal{P}_{\text{Curry}}\}.$$

Now, let $T(\mathbf{X_i}, K_i) = T_{K_i}(\mathbf{X_i})$ be the round function (for **Flurry** $X_i \in \mathbb{K}^t$, and for **Curry** $X_i \in \mathbb{K}^{m\times m}$) of **Flurry** or **Curry** (K_i is the subkey used at round i). For the first round, we have :

$$\mathbf{X_1}^{(0)} - T_{K_1}(m_0) = \mathbf{0} \in \mathcal{P}^2, \tag{1}$$

$$\mathbf{X_1}^{(1)} - T_{K_1}(m_0 + r_1) = \mathbf{0} \in \mathcal{P}^2. \tag{2}$$

$\mathbf{X_1}^{(0)}$ (resp. $\mathbf{X_1}^{(1)}$) being the first intermediate variables corresponding to m_0 (resp. $m_1 = m_0 + r_1$). From now on, a boldfaced letter will refer to a vector.

From (1) and (2), we deduce that the equations $\mathbf{X_1}^{(1)} - \mathbf{X_1}^{(0)} = \Delta_{r_1} T_{K_1}(m_0)$ are in the ideal generated by $\mathcal{P}^2$. Thus, by simply taking two pairs, we have created new equations relating the intermediates variables $\mathbf{X_1}^{(1)}, \mathbf{X_1}^{(0)}$, and the variables corresponding to K_1. The new equations are of degree strictly smaller than the initial equations of $\mathcal{P}^2$. We can iterate the process. Let $r_1, \ldots, r_N$ be a set of $N \geq 2$ linearly dependent vectors, and m_0 be a random message. We consider now the system :

$$\mathcal{P}^N = \bigcup_{r \in L[r_1,\ldots,r_N]} \mathcal{P}(m_0 + r, c_r), \text{ with } \mathcal{P} \in \{\mathcal{P}_{\text{Flurry}}, \mathcal{P}_{\text{Curry}}\}.$$

We will denote by $\mathbf{X_i}^{(j)}$ the intermediates variables used at the ith round and corresponding to the jth message[2], and c_r will be the encryption of $m_0 + r$. For the first round, we have that for all $k, 1 \leq k \leq \#L[r_1, \ldots, r_N]$:

$$\mathbf{X_1}^{(k)} - \mathbf{X_1}^{(0)} = \Delta_r T_{K_1}(m_0) \in \mathcal{P}^N, \text{ with } r \in L[r_1, \ldots, r_N].$$

As previously, we have created new low-degree equations corresponding to derivatives. But, we will also generate low-degree equations corresponding to high order derivatives. For instance, let $r_1, r_2 \in L[r_1, \ldots, r_N]$. It holds that:

$$\mathbf{X_1}^{(0)} - T_{K_1}(m_0) = \mathbf{0} \in \mathcal{P}^N, \qquad \mathbf{X_1}^{(1)} - T_{K_1}(m_0 + r_1) = \mathbf{0} \in \mathcal{P}^N,$$
$$\mathbf{X_1}^{(2)} - T_{K_1}(m_0 + r_2) = \mathbf{0} \in \mathcal{P}^N, \qquad \mathbf{X_1}^{(3)} - T_{K_1}(m_0 + r_1 + r_2) = \mathbf{0} \in \mathcal{P}^N.$$

Therefore, $\mathbf{X_1}^{(3)} - \sum_{k=0}^{2} \mathbf{X_1}^{(k)} = \Delta_{r_1,r_2} T_{K_1}(m_0) \in \mathcal{P}^N$. The ideal generated by $\mathcal{P}^N$ will now include linear relations between the intermediates variables $\mathbf{X_1}^{(j)}$. Moreover, such new linear equations will induce derivatives and high order derivatives in the subsequent rounds of the cipher. In our case, we know that $\mathbf{X_1}^{(3)} = \sum_{k=0}^{2} \mathbf{X_1}^{(k)}$. If we consider the second round $\mathbf{X_2}^{(0)} = T_{K_2}(\mathbf{X_1}^{(0)}) \in \mathcal{P}^N$, and $\mathbf{X_2}^{(3)} = T_{K_2}(\mathbf{X_1}^{(0)} + \mathbf{X_1}^{(1)} + \mathbf{X_1}^{(2)}) \in \mathcal{P}^N$. The equations $\mathbf{X_2}^{(3)} - \mathbf{X_2}^{(0)} = \Delta_{\mathbf{X_1}^{(1)} + \mathbf{X_1}^{(2)}} T_{K_2}(\mathbf{X_1}^{(0)})$ is then in the ideal generated by $\mathcal{P}^N$. In function of N, this phenomena will be propagated throughout the rounds of the cipher to generate high order differentials, and so new low-degree equations between the intermediates variables. This permits to establish an interesting

[2] We supose w.l.o.g. an explicit ordering on the elements of $L[r_1, \ldots, r_N]$.

connection between algebraic attacks and high order differential cryptanalysis [32,31]; that we can call "algebraic-high order differential" cryptanalysis.

Experimental results. To summarize, our attack works as follows. Let $N > 1$ be an integer. We fix $m_0 = (0, \ldots, 0)$, and $r_1 = (1, \ldots, 0)$. We construct differences r_i, for all $i, 2 \leq i \leq N$, using the relation $r_{i+1} = \theta \cdot r_i$. After that, we have to solve the system :

$$\mathcal{P}^N = \bigcup_{r \in U \subset L[r_1, \ldots, r_N]} \mathcal{P}(m_0 + r, c_r), \text{ with } \mathcal{P} \in \{\mathcal{P}_{\text{Flurry}}, \mathcal{P}_{\text{Curry}}\},$$

c_r being the encryption corresponding to $m_0 + r$. Note that in practice, we have not considered all the 2^N elements of $L[r_1, \ldots, r_N]$ but a smaller subset $U \subset L[r_1, \ldots, r_N]$ of size N. As explained previously, the ideal generated $\mathcal{P}^N$ includes many new low-degree equations corresponding to all the derivatives of order less than $\lfloor \ln(N) \rfloor$. We expect that these new equations will allow the ease the Gröbner basis computation. We will see that this is indeed the case in practice.

In next table, we have quoted the results we have obtained on **Flurry** and **Curry** with different values of N. Note that most of the parameters have been chosen for having a secret key of 128-bit. In this case, the ciphers considered are immune against differential/linear attacks when the number of rounds is ≥ 4 [12,13]. The experimental results have been obtained with a cluster of Xeon bi-processors 3.2 Ghz, with 64 Gb of Ram. It is well known that the efficiency of the Gröbner basis computation can vary in function of the order used. In our experiments, we have used the order proposed in [12,13]. In the table, we have included :
– T : total time of our attack; Nb_{op} : number of basic operations;
Mem : Maximum memory usage;
$D_{\max}$: the maximal degree reached during the Gröbner basis computation.
T_{BPW} : estimated complexity of the attack of [12]. Remark that this attack can not be mounted for f_{-1}.

Flurry(n,t,r,f,D)	T_{BPW}	N	$D_{\max}$	T	Nb_{op}	Mem
Flurry$(16,2,6,f_{-1},I_1)$		3	3	0.6 s.	2^{25}	1.8 Gb.
Flurry$(16,2,7,f_{-1},I_1)$		3	4	0.4 s.	2^{24}	1 Gb.
Flurry$(16,2,8,f_{-1},I_1)$		4	4	37.6 s.	2^{31}	1.4 Gb.
Flurry$(16,2,9,f_{-1},I_1)$		10	4	37296 s.	2^{41}	6.4 Gb.
Flurry$(16,4,5,f_{-1},D_2)$		2	4	0.5 s.	$2^{24.2}$	1.7 Gb.
Flurry$(16,4,6,f_{-1},D_2)$		4	4	810.3 s.	$2^{36.0}$	4.6 Gb.
Flurry$(16,8,5,f_{-1},D_4)$		3	4	3755.2 s.	$2^{37.5}$	5.4 Gb.
Flurry$(16,4,6,f_3,D_2)$	$\approx 2^{114}$	14	3	3.4 s.	$2^{27.4}$	1.3 Gb.
Flurry$(16,4,8,f_3,D_2)$	$\approx 2^{152}$	**90**	3	1952 s.	$2^{36.1}$	117 Gb.
	$\approx 2^{152}$	100	3	2058 s.	$2^{36.2}$	130 Gb.
Flurry$(16,8,6,f_3,D_4)$	$\approx 2^{228}$	20	3	35.8 s.	$2^{26.1}$	47 Gb.

Curry(n,m,r,f,D)	T_{BPW}	N	$D_{\max}$	T	Nb_{op}	Mem
Curry$(32,4,3,f_3,D_2)$	$\approx 2^{57}$	2	10	0.01 s.	$2^{12.7}$	2.4 Gb.
	$\approx 2^{57}$	17	6	0.01 s.	$2^{14.5}$	18.8 Gb.
	$\approx 2^{57}$	**20**	3	0.01 s.	$2^{11.5}$	2.1 Gb.

Interpretation of the Results. For power Sboxes, we can observe that our approach is significantly faster than BPW attack [12]. The most important is to observe that – in all cases – we have been able to find a number of pairs $N^* > 1$ such that the maximal degree reached during the Gröbner basis computation is equal to the degree d of the Sbox function. In practice, we have found this N^* by performing the following test incrementally on $N \geq 2$. We compute a DRL Gröbner basis of $\mathcal{P}^N_{\text{Flurry}}$ (resp. $\mathcal{P}^N_{\text{Curry}}$). If the maximal degree reached during this computation is greater than d then we stop the computation and set $N \leftarrow N+1$, otherwise $N \leftarrow N^*$. We can extrapolate the (experimental) complexity of our attack. Let b_{F} (resp. b_{C}) be the number of variables of the system $\mathcal{P}^{N^*}_{\text{Flurry}}$ (resp. $\mathcal{P}^{N^*}_{\text{Curry}}$). Let $\omega, 2 < \omega \leq 3$ be the linear algebra constant. The complexity of our attack is :

- $\mathcal{O}\left(b_{\text{F}}^{\deg(f)\cdot\omega}\right)$, for $\textbf{Flurry}(n,t,r,f,D)$, and
- $\mathcal{O}\left(b_{\text{C}}^{\deg(f)\cdot\omega}\right)$, for $\textbf{Curry}(n,m,r,f,D)$

We have then a polynomial time complexity for solving **Flurry** and **Curry** for (pure) power Sboxes. These results are no longer valid for the inverse function. For this Sbox, the maximum degree reached during the computation is more difficult to predict. On the other hand, we observed that our technique permits to have a significant gain of efficiency also in this case. The use of correlated messages has permitted to go one step further in the algebraic cryptanalysis of hard instances of **Flurry** instantiated with the inverse SBox. However, it is not clear that there exists an optimal number of correlated messages N^* such that the maximal degree reached during the Gröbner basis computation is bounded by a constant (for instance, 3 or 4). This deserves further investigations.

Conclusion. As explained in the introduction, a new trend in algebraic cryptanalysis is to combine statistical and algebraic techniques. Albrecht and Cid [1,2] recently proposed to mix differential and algebraic cryptanalysis. Note that there is a fundamental difference between our approach and the technique of Albrecht and Cid [1,2]. In this work, the equations derived from high order differentials are automatically (explicitly) generated during the Gröbner basis computation. In [1,2], linear equations derived from the knowledge of a differential are explicitly added to a key recovery system.

References

1. Albrecht, M., Cid, C.: Algebraic Techniques in Differential Cryptanalysis, `http://eprint.iacr.org/2007/137`
2. Albrecht, M., Cid, C.: Algebraic Techniques in Differential Cryptanalysis. In: Proceedings of the First International Conference on Symbolic Computation and Cryptography, SCC 2008, Beijing, China (April 2008)
3. Ars, G., Faugère, J.-C., Imai, H., Kawazoe, M., Sugita, M.: Comparison Between XL and Gröbner Basis Algorithms. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 338–353. Springer, Heidelberg (2004)

4. Ars, G.: Applications des Bases de Gröbner à la Cryptographie. Thèse de doctorat, Université de Rennes I (2004)
5. Bardet, M.: Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie. Thèse de doctorat, Université de Paris VI (2004)
6. Bardet, M., Faugère, J.-C., Salvy, B.: On the Complexity of Gröbner Basis Computation of Semi-Regular Overdetermined Algebraic Equations. In: Proc. of International Conference on Polynomial System Solving (ICPSS), pp. 71–75 (2004)
7. Bardet, M., Faugère, J.-C., Salvy, B., Yang, B.-Y.: Asymptotic Behaviour of the Degree of Regularity of Semi-Regular Polynomial Systems. In: Proc. of MEGA 2005, Eighth Inter. Symposium on Effective Methods in Algebraic Geometry (2005)
8. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991)
9. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology 4(1), 3–72 (1991)
10. Biham, E., Shamir, A.: Differential Cryptanalysis of of the Full 16-round DES. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 487–496. Springer, Heidelberg (1993)
11. Buchberger, B.: Ein algorithmisches Kriterium fur die Lösbarkeit eines algebraischen Gleichungssystems (An Algorithmical Criterion for the Solvability of Algebraic Systems of Equations). Aequationes mathematicae 4(3), 374–383 (1970); English translation in: Buchberger, B., Winkler, F. (eds.) Grobner Bases and Applications. In: Proceedings of the International Conference 33 Years of Gröbner Bases, RISC, Austria. Lecture Note Series, vol. 251, pp. 535–545. London Mathematical Society, Cambridge University Press (1998)
12. Buchmann, J., Pyshkin, A., Weinmann, R.-P.: Block Ciphers Sensitive to Gröbner Basis Attacks. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 313–331. Springer, Heidelberg (2006)
13. Buchmann, J., Pyshkin, A., Weinmann, R.-P.: Block Ciphers Sensitive to Gröbner Basis Attacks – Extended version, http://eprint.iacr.org/2005/200
14. Buchmann, J., Pyshkin, A., Weinmann, R.-P.: A Zero-Dimensional Gröbner Basis for AES-128. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 78–88. Springer, Heidelberg (2006)
15. Courtois, N., Pieprzyk, J.: Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)
16. Courtois, N., Meier, W.: Algebraic Attacks on Stream Ciphers with Linear Feedback. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 345–359. Springer, Heidelberg (2003)
17. Courtois, N.: Fast Algebraic Attacks on Stream Ciphers with Linear Feedback. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 176–194. Springer, Heidelberg (2003)
18. Cid, C., Murphy, S., Robshaw, M.J.B.: Small Scale Variants of the AES. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 145–162. Springer, Heidelberg (2005)
19. Cid, C., Murphy, S., Robshaw, M.J.B.: Algebraic Aspects of the Advanced Encryption Standard. Springer, Heidelberg (2006)

20. Cid, C., Albrecht, M., Augot, D., Canteaut, A., Weinmann, R.-P.: Algebraic Cryptanalysis of Symmetric Primitives. In: Deliverable STVL, ECRYPT - European Network of Excellence Collaborations (July 2008), http://hal.archives-ouvertes.fr/docs/00/32/86/26/PDF/D-STVL-7.pdf
21. Cid, C., Leurent, G.: An Analysis of the XSL Algorithm. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 333–352. Springer, Heidelberg (2005)
22. Cox, D.A., Little, J.B., O'Shea, D.: Ideals, Varieties, and algorithms: an Introduction to Computational Algebraic Geometry and Commutative algebra. In: Undergraduate Texts in Mathematics. Springer, New York (1992)
23. Faugère, J.C., Gianni, P., Lazard, D., Mora, T.: Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering. Journal of Symbolic Computation 16(4), 329–344 (1993)
24. Faugère, J.-C.: A New Efficient Algorithm for Computing Gröbner Basis: F_4. Journal of Pure and Applied Algebra 139, 61–68 (1999)
25. Faugère, J.-C.: A New Efficient Algorithm for Computing Gröbner Basis without Reduction to Zero: F_5. In: Proceedings of ISSAC, pp. 75–83. ACM Press, New York (July 2002)
26. Faugère, J.-C., Joux, A.: Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems using Gröbner bases. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 44–60. Springer, Heidelberg (2003)
27. Faugère, J.-C., Perret, L.: Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 30–47. Springer, Heidelberg (2006)
28. Faugère, J.-C., Perret, L.: Cryptanalysis of $2R^-$ Schemes. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 357–372. Springer, Heidelberg (2006)
29. Faugère, J.-C., Levy-dit-Vehel, F., Perret, L.: Cryptanalysis of MinRank. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 280–296. Springer, Heidelberg (2008)
30. Garey, M.R., Johnson, D.B.: Computers and Intractability. A Guide to the Theory of NP-Completeness. W.H. Freeman, New York (1979)
31. Knudsen, L.R.: Truncated and Higher Order Di fferentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
32. Lai, X.: Higher Order Derivatives and Differential Cryptanalysis. In: Communications and Cryptography, pp. 227–233. Kluwer Academic Publishers, Dordrecht (1994)
33. Lim, C.-W., Khoo, K.: An Analysis of XSL Applied to BES. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 242–253. Springer, Heidelberg (2007)
34. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
35. Wiedemann, D.H.: Solving sparse linear equations over finite fields. IEEE Trans. Information Theory IT-32, 54–62 (1986)

Weak Keys in RSA with Primes Sharing Least Significant Bits[⋆]

Xianmeng Meng[1] and Jingguo Bi[2]

[1] Dept. of Mathematics and Statistics
Shandong University of Finance
Jinan, 250014, P.R. China
mengxm@sdfi.edu.cn

[2] Lab of Cryptographic Technology and Information Security
Shandong University
Jinan, 250100, P.R. China
jguobi@mail.sdu.edu.cn

Abstract. Let $N = pq$ be an LSBS-RSA modulus where primes p and q have the same bit-length and share the m least significant bits, and $(p-1, q-1) = 2$. Given (N, e) with $e \in \mathbb{Z}^*_{\frac{\phi(N)}{4}}$ that satisfies $ew + z \cdot 2^{2(m-1)} = 0 \pmod{\phi(N)/4}$ with $0 < w \leq \frac{1}{9}\sqrt{\frac{\phi(N)}{e}} N^{\frac{1}{4}+\theta}$ and $|z| \leq c\frac{ew}{\phi(N)} N^{\frac{1}{4}-\theta}$, we can find p and q in polynomial time. We show that the number of these weak keys e is at least $N^{\frac{3}{4}+\theta-\varepsilon}$, where $\theta = m/\log_2 N$, and there exists a probabilistic algorithm that can factor N in time $O(N^{\frac{1}{4}-\theta+\varepsilon})$.

Keywords: RSA, Coppersmith's theorem.

1 Introduction

The RSA cryptosystem was proposed by Rivest, Shamir and Adleman [9] in 1978, in which the public and private exponents are chosen to be inverses of each other modulo $\varphi(N) = (p-1)(q-1)$. Usually, in the original RSA $\gcd(p-1, q-1) = 2$.

It is well-known that small prime difference makes RSA insecure. In [13], Weger showed that if $p - q$ is small, RSA system with small exponent is much more vulnerable. In [2], Blömer and May proved that the number of weak keys in their approach increases as $p-q$ decreases. But only requiring the primes to have large difference will not ensure the security of RSA with small private-exponent. See the LSBS-RSA.

The LSBS-RSA is an RSA system with modulus primes sharing a large number of least significant bits. This RSA variant was suggested to improve the computational efficiency of server-aided signature generation in [1][10]. The security of LSBS-RSA under the partial key exposure attacks has been analyzed by

[⋆] This research is partially supported by Project 973 (no: 2007CB807902) and the natural science foundation in Shandong province (no: Y2008A22) in China.

Steinfeld and Zheng [10] and Sun et. al. [12]. The cryptanalysis of short exponent LSBS-RSA was given by Zhao and Qi [15] and further by Sun et. al. [11].

In this paper we give an attack to LSBS-RSA. We are able to show that if $N = pq$ and primes p and q share the m least significant bits, there are at least $N^{\frac{3}{4}+\theta-\varepsilon}$ weak keys (N, e), where $\theta = m/\log_2 N$. Thus the number of weak keys increases as primes p and q sharing more least significant bits. As θ approaches $\frac{1}{4}$, almost all keys are weak. Notice that if p and q sharing the m least significant bits, then the m least significant bits are known. Thus our attack has an interpolation property towards Coppersmith's theorem [3], which states that if primes p and q are balanced and the $\frac{1}{4}\log_2 N$ least significant bits of p are known, then $N = pq$ can be factored in polynomial time. In case $m = 1$, our result is consistent with that of [2].

The remainder of this paper is organized as follows. In section 2, we review the LSBS-RSA and the lattice attack. In section 3, we show our attack on the LSBS-RSA. In section 4, we show the number of weak LSBS-RSA keys under our attack. Conclusions are finally drawn in section 5.

2 Preliminary

2.1 Primes Sharing the Least Significant Bits

In this paper, we denote an LSBS-RSA modulus $N = pq$ where primes p and q are balanced and share the m least significant bits, and

$$p - q \geq cN^{\frac{1}{2}} \tag{1}$$

with $0 < c \leq 1$ being a constant. Furthermore, in this paper, as in the original RSA, we assume

$$\gcd(p-1, q-1) = 2. \tag{2}$$

Since primes p and q share the m least significant bits, we may write $p = p_1 \cdot 2^m + v_0$, $q = q_1 \cdot 2^m + v_0$. The following lemma is a reformation of Lemma 3.1 in [10] and Lemma 1 in [11].

Lemma 1. *Let $N = pq$ be an LSBS-RSA modulus, where p and q share the m least significant bits. Then in time polynomial in $\log N$, we can compute u_0 and v_0 such that*

$$p = p_1 \cdot 2^m + v_0, \quad q = q_1 \cdot 2^m + v_0, \quad p + q = u_1 \cdot 2^{2m} + u_0,$$

where p_1, q_1 and u_1 are unknown.

Proof. Since p and q share the m least significant bits, we write $p = p_1 \cdot 2^m + v_0$ and $q = q_1 \cdot 2^m + v_0$. By $pq = N$, we have

$$(p_1 \cdot 2^m + v_0)(q_1 \cdot 2^m + v_0) = N,$$

and then $v_0^2 \equiv N \pmod{2^m}$. By Lemma 2.1 in [10], we can obtain v_0 in time $O(m^2)$ bit-operations.

Since $(p-q)^2 = (p+q)^2 - 4N$, we have

$$((p_1+q_1)\cdot 2^m + 2v_0)^2 - 4N = (p_1-q_1)^2 \cdot 2^{2m},$$

$$(p_1+q_1)^2 \cdot 2^{2m} + 4v_0(p_1+q_1)\cdot 2^m - 4N + 4v_0^2 = (p_1-q_1)^2 \cdot 2^{2m}.$$

Since $2|p_1+q_1$, we have

$$4v_0(p_1+q_1)\cdot 2^m \equiv 4N - 4v_0^2 \pmod{2^{2(m+1)}}.$$

Let $v_0^{-1}\cdot v_0 \equiv 1 \pmod{2^{2(m+1)}}$. We have

$$4(p_1+q_1)\cdot 2^m \equiv 4(N-v_0^2)v_0^{-1} \pmod{2^{2(m+1)}}.$$

Let $u_0 \equiv (N-v_0^2)v_0^{-1} + 2v_0 \pmod{2^{2m}}$. By the above, we have that $(p_1+q_1)\cdot 2^m + 2v_0 = u_0 + v_1 \cdot 2^{2m}$. Thus $p+q = (p_1+q_1)\cdot 2^m + 2v_0 = u_0 + u_1 \cdot 2^{2m}$, where u_1 is unknown.

2.2 Lattice Attack

Let $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_\omega \in \mathbb{Z}^n$ be ω linearly independent vectors over $\mathbb{Z}$, and

$$L = \left\{ \sum_{i=1}^{\omega} a_i \mathbf{u}_i | \text{ where } a_i \in \mathbb{Z} \text{ and } \mathbf{u}_i \in \mathbb{Z}^n \text{ for } i = 1, \cdots, \omega \right\}.$$

The vectors $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_\omega$ are also called basis of lattice L. We define the determinant of the lattice as the square root of the Gramian determinant $\det_{1\le i,j\le \omega} < u_i, u_j >$.

The following is a well-known theorem [8] on a short vector in lattice L.

Theorem 1. *(Minkowski) Every n-dimensional lattice L contains a non-zero vector $\mathbf{v}$ with*

$$||\mathbf{v}|| \le \sqrt{n} \cdot \det(L)^{\frac{1}{n}}.$$

LLL algorithm outputs a short vector in lattice L. This algorithm works in deterministic polynomial time. One can see [5] for details. Coppersmith [3] took the advantage of LLL algorithm to find the small roots of a modular equation. Define the norm of a polynomial $h(x,y) = \sum_{i,j} a_{i,j} x^i y^j$ as $||h(x,y)||^2 = \sum_{i,j} a_{i,j}^2$. Using the following lemma, if the coefficients of $h(x,y)$ are sufficiently small, we have that $h(x_0,y_0) = 0$ holds over the integers. The proof can be found in earlier citations, see [3] for example. The original lemma can be found in [4].

Lemma 2. *(Howgrave-Graham) Let $h(x,y) \in \mathbb{Z}[x,y]$ be a bivariate polynomial which is a sum of at most ω monomials. Further, let m be a positive integer. Suppose that*

1. *$h(x_0,y_0) = 0 \pmod{H^m}$, where $|x_0| < X$ and $|y_0| < Y$;*
2. *$||h(xX, yY)|| < H^m/\sqrt{\omega}$.*

Then $h(x_0, y_0) = 0$ holds over the integers.

Here is the result of Coppersmith [3] in the reformulation of May [7].

Theorem 2. *(Coppersmith) Let N be an integer of unknown factorization with a divisor $d \geq N^{\beta}$. Let $f_d(x)$ be a univariate, monic polynomial of degree δ. Furthermore, let c_N be a function that is upper-bounded by a polynomial in $\log N$. Then we can find all solutions x_0 for the equation $f_d(x) = 0 \pmod{d}$ with*

$$|x_0| \leq c_N N^{\frac{\beta^2}{\delta}}$$

in polynomial time in $(\log N, \delta)$.

From Theorem 2, we obtain the following lemma which will be applied later.

Lemma 3. *Let $N = pq$ be an LSBS-RSA modulus, where primes p and q share the m least significant bits such that*

$$p = p_1 \cdot 2^m + v_0, \quad q = q_1 \cdot 2^m + v_0.$$

Given an approximation of p_1 with an error at most $c_1 N^{\frac{1}{4}}$, then one can recover p and q in polynomial time.

Proof. By Lemma 1, the above v_0 is known. Let $\bar{p_1}$ be an approximation of p_1 with error x such that $|x| \leq c_1 N^{\frac{1}{4}}$. Then we have $p_1 = \bar{p_1} + x$ and $p = \bar{p_1} \cdot 2^m + x \cdot 2^m + v_0$.

Let $b \cdot 2^m \equiv 1 \pmod{N}$. Then by the above and p is a divisor of N, we have

$$bp = \bar{p_1} + x + bv_0 = 0 \pmod{p}, \text{ with } |x| \leq c_1 N^{\frac{1}{4}}.$$

By Theorem 2, p and q can be recovered in time polynomial in $\log N$. This concludes the lemma.

3 Main Result

Now we state our main theorem and give a proof in this section. We will apply the similar technique that is used in Chapter 4 in [6], which is a generalization of Wiener's attack [14].

Theorem 3. *Let $0 < c \leq 1$ and let (N, e) be an RSA public key tuple with $N = pq$ where primes p and q satisfy (1) and share the m least significant bits such that $p - q = 2^m \cdot u$. Let $\theta = m/\log_2 N$. Suppose that $e \in \mathbb{Z}^*_{\frac{\phi(N)}{4}}$ satisfies an equation*

$$ew + z \cdot 2^{2(m-1)} = k\phi(N)/4 \tag{3}$$

with $w, z, k \in \mathbb{Z}$ and

$$0 < w \leq \frac{1}{9}\sqrt{\frac{\phi(N)}{e}} N^{\frac{1}{4}+\theta} =: X, \quad |z| \leq c \frac{ew}{\phi(N)} N^{\frac{1}{4}-\theta} =: Z. \tag{4}$$

Then N can be factored in time polynomial in $\log N$.

Here is the roadmap for the proof of Theorem 3.

1. We construct a lattice such that we can use the lattice attack to find w and k;
2. From w and k, we compute an approximation of $p_1 + q_1$. And from an approximation of $p_1 + q_1$, we compute an approximation of $p_1 - q_1$;
3. Combining both approximations gives us an approximation of p_1, which leads to the factorization of N by using Coppersmith's theorem.

Proof. Let us begin with equation (3). Using $\phi(N) = N+1-(p+q)$, we rewrite (3) as

$$ew + z \cdot 2^{2(m-1)} = k(N+1-(p+q))/4.$$

By Lemma 1, we have

$$ew + z \cdot 2^{2(m-1)} = k(N+1-(u_1 \cdot 2^{2m} + u_0))/4,$$

and

$$ew + (ku_1 + z)2^{2(m-1)} = k(N+1-u_0)/4.$$

Let $M = (N+1-u_0)/4$. We have

$$ew + (ku_1 + z) \cdot 2^{2(m-1)} = kM.$$

By the assumption in (2), we have $\gcd(M, 2) = 1$. Compute a such that

$$a \cdot 2^{2(m-1)} \equiv 1 \pmod{M} \tag{5}$$

and compute h_1 such that $a \cdot 2^{2(m-1)} = 1 + h_1 M$. By this, we have that

$$aew + (ku_1 + z) = hM, \tag{6}$$

where

$$h = ak + h_1(ku_1 + z). \tag{7}$$

Let $f_M(x, y) = aex + y$. This gives us a bivariate polynomial $f_M(x, y)$ with the root $(x_0, y_0) = (w, ku_1 + z)$ modulo $M = (N+1-u_0)/4$. And (6) can be reformed as

$$aex + y = hM. \tag{8}$$

Without loss of generality, we assume

$$\gcd(x, h) = 1. \tag{9}$$

Otherwise, every integer that divides both x and h must also divide y by (8), thus we can divide (8) by $\gcd(x, h)$ which gives us an equation $eax' + y' = 0 \pmod{M}$ with even smaller parameters x' and y'.

Now we apply Coppersmith's method by Lemma 2 to transform the polynomial $f_M(x, y)$ into a polynomial $f(x, y)$ satisfying $f(x_0, y_0) = 0$ over integers (and not just modulo M), and then compute (x_0, y_0). By definition, we have that $0 < x \le X$ and $0 < x_0 \le X$.

Since

$$|z \cdot 2^{2(m-1)}| \leq \frac{c}{4}\frac{ew}{\phi(N)}N^{\frac{1}{4}+\theta} < \frac{1}{2}ew$$

and $k = \left(ew + z \cdot 2^{2(m-1)}\right)/\phi(N)$, we have

$$\frac{1}{2}\frac{ew}{\phi(N)} \leq k \leq \frac{3}{2}\frac{ew}{\phi(N)} \leq \frac{3}{2}\frac{e}{\phi(N)}X. \tag{10}$$

We have

$$y_0 = ku_1 + z \leq \frac{3}{2}\frac{e}{\phi(N)}Xu_1 + z < 5\frac{e}{\phi(N)}XN^{\frac{1}{2}-2\theta} =: Y.$$

Let $g(x,y) = c_0Mx + c_1f_M(x,y)$. The coefficient vectors of $g(xX, yY)$ form a lattice L in $\mathbb{Z}^2$, where L is spanned by the row vectors of the (2×2)-lattice basis

$$B = \begin{bmatrix} MX & 0 \\ aeX & Y \end{bmatrix}.$$

By Theorem 1, we know that L contains a vector v with $||v|| \leq \sqrt{2\det(L)}$. Hence, whenever the condition

$$\sqrt{2\mathrm{det}(L)} < \frac{1}{\sqrt{2}}M$$

is satisfied, then $||v|| \leq \frac{1}{\sqrt{2}}M$. The condition is equivalent to $\det(L) < \frac{1}{4}M^2$. Since $M = (N+1-u_0)/4$, we have that

$$\det(L) = MXY < \frac{5}{81}NM \leq \frac{1}{16}(N+1-u_0)^2$$

holds for $N > 884$. This means that $\sqrt{2\mathrm{det}(L)} < \frac{1}{\sqrt{2}}M$ is satisfied.

By Theorem 1, L contains a vector $v = (c_0, c_1)\cdot B$ with $||v|| \leq \frac{1}{\sqrt{2}}M$. We interpret v as the coefficient vector of a polynomial $g(xX, yY)$ and apply Lemma 2, then we know that $g(x_0, y_0) = c_0Mx_0 + c_1f_M(x_0, y_0) = 0$ over $\mathbb{Z}$. Using (8), we conclude that $f_M(x_0, y_0) = hM$. Reordering terms leads to

$$\frac{h}{x_0} = -\frac{c_0}{c_1}.$$

By construction, we know that the parameters h and x_0 are positive. Our goal is to show that the fractions on both sides of the equation are in their lowest terms, since then $h = |c_0|$ and $x_0 = |c_1|$. But $\gcd(h, x_0) = \gcd(h, w) = 1$ as we assumed in (9). And we have $\gcd(c_0, c_1) = 1$, since v is a shortest vector in L.

Therefore, the coefficients c_0, c_1 of a shortest vector in L (in terms of the basis B) give us the secret parameters h and x_0. Using (8), we can compute

$$y_0 = hM - aex_0 = |c_0|M - |c_1|ae.$$

Notice that exactly one of the coefficients c_0 and c_1 is negative. Therefore $|y_0| = |c_0 M + c_1 ae|$, but we know that $y_0 > 0$ which proves the claim. Since $y_0 = ku_1 + z$ is known and h is known, we can compute k by (7).

Now we obtain the secret parameters (x_0, y_0) and k. Since $y_0 = ku_1 + z$, we have

$$\frac{y_0}{k} = u_1 + \frac{z}{k} = \frac{p+q-u_0}{2^{2m}} + \frac{z}{k} = \frac{p_1+q_1}{2^m} + \frac{2v_0 - u_0}{2^{2m}} + \frac{z}{k}, \tag{11}$$

then

$$2^m \cdot \left(\frac{y_0}{k} - \frac{2v_0 - u_0}{2^{2m}}\right) = p_1 + q_1 + \frac{z}{k} \cdot 2^m.$$

Let $s = 2^m \cdot (\frac{y_0}{k} - \frac{2v_0-u_0}{2^{2m}})$. Then s is an approximation of $p_1 + q_1$ up to some additive error $\frac{z}{k} \cdot 2^m$. By (10) and (4), we have

$$\left|\frac{z}{k} \cdot 2^m\right| \le 2cN^{\frac{1}{4}}. \tag{12}$$

By (12) and $0 < \theta < \frac{1}{4}$, we have that $\frac{z}{k} \cdot 2^m < 2cN^{\frac{1}{4}} \le 2N^{\frac{1}{4}} < \frac{4}{3}(p_1 + q_1)$. Thus

$$s = 2^m \cdot \left(\frac{y_0}{k} - \frac{2v_0 - u_0}{2^{2m}}\right) = p_1 + q_1 + \frac{z}{k} \cdot 2^m \le \frac{7}{3}(p_1 + q_1). \tag{13}$$

Also by (12), we have

$$s - (p_1 + q_1) = \frac{z}{k} \cdot 2^m \le 2cN^{\frac{1}{4}},$$

and thus

$$2^m s - (p+q) \le 2cN^{\frac{1}{4}} \cdot 2^m. \tag{14}$$

Let $t = \sqrt{s^2 - 4N \cdot 2^{-2m}}$. By (13) and (14), and the condition $p - q > cN^{\frac{1}{2}}$, we have

$$\begin{aligned} t - (p_1 - q_1) &= \frac{2^m t - (p-q)}{2^m} \\ &= \frac{(2^m s - (p+q))(2^m s + (p+q))}{2^m(2^m t + (p-q))} \\ &\le \frac{2cN^{\frac{1}{4}} \cdot \frac{10}{3}(p+q)}{p-q} < 14N^{\frac{1}{4}}. \end{aligned} \tag{15}$$

Then by (3) and (15) we have

$$\begin{aligned} &\left|\frac{s+t}{2} - p_1\right| = \frac{1}{2}\left|s - (p_1+q_1) + t - (p_1 - q_1)\right| \\ &\le \frac{1}{2}\left|s - (p_1+q_1)\right| + \frac{1}{2}\left|t - (p_1 - q_1)\right| \\ &< cN^{\frac{1}{4}} + 7N^{\frac{1}{4}} \le 8N^{\frac{1}{4}}. \end{aligned}$$

Thus $\frac{s+t}{2}$ is an approximation to p_1 with an error $8N^{\frac{1}{4}}$. Then one can recover p by Lemma 3.

In conclusion, we summarize the whole algorithm which leads to the factorization of the modulus.

```
Algorithm
Input: $(N, e)$, where $N = pq$ and $ew + z \cdot 2^{2(m-1)} = k\phi(N)/4$ for some unknown
$0 < w \leq X$ and $|z| \leq Z$.

1. Let $aex + y = hM$ with $x \leq X$ and $y \leq Y$, where $a$ and $h$ are defined in
   (5) and (7) respectively;
2. Construct the lattice $L$ with base $B$, where $B = \begin{bmatrix} MX & 0 \\ aeX & Y \end{bmatrix}$;
3. Find a short vector $v = (c_0, c_1) \cdot B \in L$ using Gauss reduction;
4. Compute $(x_0, y_0) = (|c_1|, |c_0 N + c_1 e|)$ and $h = c_0$;
   (a) Compute $s = 2^m \cdot (\frac{y_0}{k} - \frac{2v_0 - u_0}{2^{2m}})$, $t = \sqrt{s^2 - 4N \cdot 2^{-2m}}$ and $\bar{p_1} = [(s + t)/2]$;
   (b) Apply Coppersmith's algorithm (Theorem 2) to the candidate $\bar{p_1}$. If
       Coppersmith's algorithm output the factorization of $N$, then stop.

Output: $p$, $q$.
```

Every step in Algorithm can be done in polynomial time. This concludes the proof of Theorem 3.

4 Weak Keys

If a public key (N, e) yields the factorization of N in polynomial time, then (N, e) is called a *weak key*. In this section, our purpose is to prove

Theorem 4. *Let* (N, e) *be the public key tuples defined in Theorem 3 with* $0 < \theta < \frac{1}{4}$. *Then the number of such weak keys is at least* $\Omega(\frac{N^{\frac{3}{4}+\theta}}{(\log\log N)^2})$.

The public key (N, e) satisfying Theorem 3 can also be written as $e \equiv -\frac{z \cdot 2^{2(m-1)}}{w}$ (mod $\phi(N)/4$), with $e \in \mathbb{Z}^*_{\frac{\phi(N)}{4}}$ where w and z and $\phi(N)/4$ are pairwise relatively prime, and w and z satisfy (4). Let E be the class of all these (N, e). We call E a weak class, since there is an algorithm that on (N, e) outputs the factorization of N in polynomial time. Let $Size_E(N)$ be the cardinality of E. To prove Theorem 4, we only have to show $Size_E(N) = \Omega\left(\frac{N^{\frac{3}{4}+\theta}}{(\log\log N)^2}\right)$. Since $e \equiv -\frac{z \cdot 2^{2(m-1)}}{w}$ (mod $\phi(N)/4$), we have that every (w, z) satisfying $0 < w \leq X$ and $|z| \leq Z$ defines a public key e with $ew + z \cdot 2^{2(m-1)} = k\phi(N)/4$. Now we show that different (w, z) defines different public key e.

Lemma 4. *Let* w, z, w' *and* z' *be integers such that* $\gcd(w, z\phi(N)) = \gcd(zz', \phi(N)) = \gcd(w', z'\phi(N)) = 1$. *Let*

$$e \equiv -\frac{z \cdot 2^{2(m-1)}}{w} \pmod{\phi(N)/4} \qquad (16)$$

and

$$e' \equiv -\frac{z' \cdot 2^{2(m-1)}}{w'} \pmod{\phi(N)/4}, \tag{17}$$

with $e, e' \in \mathbb{Z}^*_{\frac{\phi(N)}{4}}$. *If* $(w,z) \neq (w',z')$, *then* $e \neq e'$.

Proof. Suppose that $(w,z) \neq (w',z')$, and for contradiction, that $e = e'$ where e and e' satisfy (16) and (17). Since $\gcd(w,z) = \gcd(w',z') = 1$, then by the above $e = e'$ implies

$$z \cdot 2^{2(m-1)} w' \equiv z' \cdot 2^{2(m-1)} w \pmod{\phi(N)/4}$$

and

$$(zw' - z'w) \cdot 2^{2(m-1)} = 0 \pmod{\phi(N)/4}.$$

By the assumption in (2), we have $\gcd(2, \phi(N)/4) = 1$, and thus by the above

$$zw' - z'w = 0 \pmod{\phi(N)/4}.$$

For $0 < |zw'|, |z'w| \le ZX < \phi(N)/4$, we have $zw' = z'w$, and $\frac{z}{w} = \frac{z'}{w'}$. Since $\gcd(w,z) = \gcd(w',z') = 1$, we have $(w,z) = (w',z')$, which is a contradiction. Therefore $e = e'$ and the claim follows.

By the above, we have that every (w,z) with $w \le X$ and $z \le Z$ defines a public key $e \in \mathbb{Z}^*_{\frac{\phi(N)}{4}}$ satisfying $ew + z \cdot 2^{2(m-1)} = k\phi(N)/4$, and different (w,z) defines different public key e. Thus we have

$$Size_E(N) = \sum_{\substack{0<w\le X \\ \gcd(w,\phi(N)/4)=1}} \sum_{\substack{|z|\le Z \\ \gcd(z,w\phi(N)/4)=1}} 1. \tag{18}$$

By Corollary 14 in [2], we have

$$\begin{aligned} Size_E(N) &\ge \sum_{\substack{0<w\le X \\ \gcd(w,\phi(N)/4)=1}} \left(\frac{Z}{2\log\log N^2} - O(N^{\varepsilon}) \right) \\ &\ge \frac{N^{\frac{3}{4}+\theta}}{4(\log\log N)^2} - O(N^{\frac{1}{4}+\theta+\varepsilon}). \end{aligned}$$

This completes the proof of Theorem 4.

5 Conclusion

In this paper, we use the lattice attack to the LSBS-RSA. Let $N = pq$ be an LSBS-RSA modulus where primes p and q have the same bit-length and share the m least significant bits, and $(p-1, q-1) = 2$. We investigate an attack on the LSBS-RSA keys e satisfying $ew + z \cdot 2^{2(m-1)} = k\phi(N)/4$ with a small solution $0 < w \le X$ and $|z| \le Z$. We compute the number of keys satisfying $e \in \mathbb{Z}^*_{\frac{\phi(N)}{4}}$ and $ew + z \cdot 2^{2(m-1)} = k\phi(N)/4$ with $0 < w \le X$ and $|z| \le Z$, and we are able to give a lower-bound of the number of such keys e by $\Omega\left(N^{\frac{3}{4}+\theta-\varepsilon}\right)$, where $\theta = m/\log_2 N$.

References

1. Bellare, M., Rogaway, P.: The exact security of digital signatures: How to sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
2. Blömer, J., May, A.: A generalized wiener attack on RSA. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 1–13. Springer, Heidelberg (2004)
3. Coppersmith, D.: Small solutions to polynomial equations and low exponent RSA vulnerabilities. Journal of Cryptology 10(4), 223–260 (1997)
4. Howgrave-Graham, N.: Finding small roots of univariate modular equations revisited. In: Darnell, M.J. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 131–142. Springer, Heidelberg (1997)
5. Lenstra, H.W., Lenstra, A.K., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261, 513–534 (1982)
6. May, A.: New RSA Vulnerabilities Using Lattice Reduction Methods. PhD thesis, University of Paderborn (2003)
7. May, A.: Using LLL-reduction for solving RSA and factorization problems: a survey. In: LLL+25 Conference in Honour of the 25th Birthday of the LLL Algorithm (2007)
8. Minkowski, H.: Geometrie der Zahlen. Teubner Verlag (1912)
9. Shamir, A., Rivest, R.L., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. of the ACM 21, 120–126 (1978)
10. Steinfeld, R., Zheng, Y.: On the security of RSA with primes sharing least- significant bits. Appl. Algebra Eng. Commun. Comput. 15(3-4), 179–200 (2004)
11. Sun, H.-M., Wu, M.-E., Steinfeld, R., Guo, J., Wang, H.: Cryptanalysis of short exponent RSA with primes sharing significant bits. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 49–63. Springer, Heidelberg (2008)
12. Sun, H.-M., Wu, M.-E., Wang, H., Guo, J.: On the improvement of the BDF attack on LSBS-RSA. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 84–97. Springer, Heidelberg (2008)
13. de Weger, B.: Cryptanalysis of RSA with small prime difference. Applicable Algebra in Engineering 13, 17–28 (2002)
14. Wiener, M.: Cryptanalysis of short RSA secret exponents. IEEE Transactions on Information Theory 36, 553–558 (1998)
15. Zhao, Y.-D., Qi, W.-F.: Small private-exponent attack on RSA with primes sharing bits. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 221–229. Springer, Heidelberg (2007)

Hybrid Proxy Re-encryption Scheme for Attribute-Based Encryption

Takeo Mizuno[1,2] and Hiroshi Doi[2]

[1] NTT DATA CORPORATION, 3-3-3 Toyosu, Koutou-ku, Tokyo, 135-6033 Japan
[2] INSTITUTE of INFORMATION SECURITY, 2-14-1 Tsuruya-cho, Kanagawa-ku, Yokohama-shi, Kanagawa, 221-0835 Japan
mizunotko@nttdata.co.jp, doi@iisec.ac.jp

Abstract. In ciphertext policy *attribute based encryption* (ABE) schemes the sender selects an access structure and generates a ciphertext, which decryptors can get plaintext if he has certain set of secret key associate with his attributes which satisfies the access structure. On the other hand, many organisations already introduced standard *identity based encryption* (IBE) or *public key encryption* (PKE) where only a single recipient is specified at the time of encryption. To utilize the above schemes and to simplify the management of user's key, it is valuable to develop a proxy re-encryption schemes between ABE schemes and IBE schemes. In this paper we propose the first proxy re-encryption scheme, which can convert an ABE ciphertext to a ciphertext which is encrypted by IBE scheme. Using new proxy re-encryption scheme, some useful applications can be constructed. Furthermore, we prove the security in the standard model based on decisional bilinear Diffie-Hellman assumption.

Keywords: attribute-based encryption, proxy re-encryption, identity-based encryption, bilinear maps.

1 Introduction

In ciphertext policy *attribute based encryption* (ABE) schemes the sender selects an access structure and generates a ciphertext, which decryptors can get plaintext if he has certain set of secret key associate with his attributes which satisfies the access structure. Using the ABE scheme, the sender does not specify all the recipients at the time of encryption, and enforces access policies, defined on attributes within the encryption procedure.

Suppose some organisation plan to apply an ABE scheme to share the secret data among members securely. They might have been applied *identity based encryption* (IBE) or *public key encryption* (PKE) system, where only a single recipient is specified at the time of encryption, already. If they plan to distribute the ABE secret keys to the members, the ABE key authority should generates ABE secret keys to each members, and the member should manage the ABE secret key in addition.

On the other hand, in proxy re-encryption schemes, a semi-trusted entity called proxy can convert a ciphertext encrypted for Alice into a new ciphertext, which another user Bob can decrypt with his own secret information without revealing the underlying plaintext. Because the proxy is not fully trusted, it is required that the proxy cannot reveal Alice's and Bob's secret key, and cannot learn the plaintext during the conversion. If the proxy which can convert the ABE ciphertext to the IBE ciphertext exists, the IBE user can decrypt the ABE ciphertext using his own IBE secret key only. In this paper, we propose the first proxy re-encryption scheme which can convert the ABE ciphertext to the IBE ciphertext securely.

If the above mentioned organisation builds the gateway which converts ABE ciphertexts to IBE ciphertext with our new proxy re-encryption scheme, the member of the organization can access ABE ciphertexts only using this gateway, and stores IBE secret key only. This gateway only re-encrypts ABE ciphertext to IBE ciphertext without revealing underling plaintext.

Furthermore, the member of the organisation does not need to consider about decryption operation of the ABE scheme. The proxy removes the effect of the ABE scheme.

1.1 Attribute-Based Encryption Schemes

The ABE schemes were first introduced by Sahai and Waters as an application of their fuzzy IBE scheme [2], which have single threshold access structure.

Two variants of ABE were subsequently proposed. The key policy attribute-based encryption (KP-ABE) was proposed by Goyal, Pandey, Sahai and Waters in [22]. In [22], every ciphertext are associated with a set of attributes, and every user's secret key is associated with a monotone access structure on attributes. Decryption is enabled iff the ciphertext attribute set satisfies the access structure on the user's secret key. The first ciphertext policy attribute-based encryption (CP-ABE) was proposed by Bethencourt, Sahai and Waters in [11]. In [11], the situation is reversed: attributes are associated with user's secret keys and monotone access structures with ciphertexts. However, in [11], the security of scheme was proved in generic bilinear group model only. Cheung and Newport proposed a simple and provably secure CP-ABE scheme in standard model, where the access policy is defined by AND gates, in [12]. Waters [5] recently proposed the first fully expressive CP-ABE in the standard model.

1.2 Proxy Re-encryption Schemes

Several proxy re-encryption schemes have been proposed in the context of *public key encryption* (PKE), e.g., ElGamal or RSA. Other schemes have been proposed in the context of *identity based encryption* (IBE) which the sender encrypts a plaintext using arbitral strings that represents the recipient's identity as the public key.

Matsuo proposed a hybrid proxy re-encryption scheme which can convert a PKE ciphertext to an IBE chiphertext in [20]. He also classifies proxy re-encryption schemes as follows:

[PKE-PKE] type Proxy converts PKE ciphertexts to PKE ciphertexts. [17], [14], [16], [23], [13], [10], [18], [9] and [3] have been proposed as this type.

[IBE-IBE] type Proxy converts IBE ciphertexts to IBE ciphertexts. [23], [20], [15], and [6] have been proposed as this type.

[PKE-IBE] type Proxy converts PKE ciphertexts to IBE ciphertexts. [20] has been proposed as this type.

[IBE-PKE] type Proxy converts IBE ciphertexts to PKE ciphertexts. [15] [21] have been proposed as this type.

1.3 Our Contribution

We propose [ABE-IBE] type proxy re-encryption scheme, which can convert a ciphertext encrypted by ABE scheme to an IBE ciphertext, without revealing the underlying plaintext. Our scheme holds the following advantages simultaneously.

- Our scheme achieve proxy invisibility, which means delegatee does not require additional algorithm and does not require additional secret information while decrypting a re-encrypted ciphertext.
- In our scheme the size of a re-encrypted ciphertext is same as an original ABE ciphertext, while in some scheme ([15]) requires additional elements of ciphertext only used for re-encryption.
- Our scheme are secure in the standard model against chosen plaintext attack. We prove the security, combining two different scheme (ABE and IBE) all together.

1.4 Organisation

The rest of paper consists of 4 sections. In section 2 we give some definitions and preliminaries. In section 3 we define security of [ABE-IBE] type proxy re-encryption. In section 6 we give an extension of our scheme. In section 4 we present the [ABE-IBE] type proxy re-encryption scheme, in section 5 we prove the security, and finally conclude this study in section 7.

2 Preliminaries

2.1 Bilinear Groups

Let $\mathbb{G}$, $\mathbb{G}_T$ be the two multiplicative cyclic groups of prime order p, and g be a generator of $\mathbb{G}$. We say that $\mathbb{G}_T$ has an admissible bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ if the following conditions hold.

1. $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$ for all a, b
2. $\hat{e}(g, g) \neq 1$

We say that $\mathbb{G}$ is a bilinear group if the group action in $\mathbb{G}$ can be computed efficiently and there exists a group $\mathbb{G}_T$ and an efficiently computable bilinear map $\hat{e}$ as above.

2.2 Decisional Bilinear Diffie-Hellman (DBDH) Assumption

The Decisional BDH problem [1], [19], [7] in $\mathbb{G}$ is defined as follows:

The challenger chooses $a, b, c \in \mathbb{Z}_p$ at random and then flips a fair binary coin β. If $\beta = 1$ it outputs the tuple $\left\langle g, g^a, g^b, g^c, \hat{e}(g,g)^{abc} \right\rangle \in \mathbb{G}^4 \times \mathbb{G}_T$. Otherwise, if $\beta = 0$, the challenger choose $\Gamma_T \in_R \mathbb{G}_T$ at random and outputs the tuple $\left\langle g, g^a, g^b, g^c, \Gamma_T \right\rangle \in \mathbb{G}^4 \times \mathbb{G}_T$. The adversary must then output a guess β' of β. An adversary, $\mathcal{B}$, has at least an ϵ advantage in solving the decisional DBDH problem if $\left|\Pr[\beta = \beta'] - \frac{1}{2}\right| \geq \epsilon$ where the probability is taken over the random choice of the generator g, the random choice of a, b, c in $\mathbb{Z}_p$ and Γ_T in $\mathbb{G}_T$, and the random bits consumed by $\mathcal{B}$.

Definition 1. *The Decisional (κ, t, ϵ)-BDH assumption holds in $\mathbb{G}$ if no t-time adversary has at least ϵ advantage in solving the Decisional BDH problem in $\mathbb{G}$ under a security parameter κ.*

2.3 Ciphertext Policy ABE

The access structure on attribute is a rule W that returns either 0 or 1 given a set S of attributes. We say that S satisfies W(written $S \models W$) if and only if W answers 1 on S. In ABE schemes, access structures may be Boolean expressions, threshold trees, etc. A ciphertext policy attribute based encryption (ABE) consists of the following algorithms.

$SetUp_A(1^\kappa)$ Given a security parameter 1^κ as input, outputs a public key PK_A and master secret key MK_A.

$KeyGen_A(MK_A, S)$ Let $\mathcal{N}$ be the set of all attributes in the system. For input of a master secret key MK_A, and a set $S \in \mathcal{N}$ of attributes, outputs a secret key SK_A associated with S.

$Encrypt_A(PK_A, M, W)$ For input of a public key PK_A, a message M and an access structure W, outputs a ciphertext C_A with the property that a user with a secret key generated from attribute set S can decrypt C_A iff $S \models W$.

$Decrypt_A(C_A, SK_A)$ For input of a ciphertext C_A and a secret key SK_A, outputs the message M if $S \models W$, where S is the attribute set used to generate SK_A.

2.4 Identity Based Encryption

Identity Based Encryption (IBE) consists of the following algorithm.

$SetUp_I(1^\kappa)$ Given a security parameter 1^κ as input, a trusted entity called Private Key Generator (PKG) generates a master key MK_I and public parameters π, and outputs MK_I and π.

$KeyGen_I(MK_I, \pi, ID)$ For inputs of a master key MK_I, public parameters π, and an identity ID, the PKG outputs an IBE secret key SK_I corresponding to the identity.

$Enc_I(ID, \pi, M)$ For inputs of an identity ID, public parameters π, and a plaintext M, outputs an IBE ciphertext C_I.

$Dec_I(SK_I, \pi, C_I)$ For inputs of a IBE secret key SK_I, public parameters π, and an IBE ciphertext C_I, decrypts a plaintext M.

2.5 [ABE-IBE] Proxy Re-encryption

The [ABE-IBE] type proxy re-encryption consists of the following algorithms.

$KenGen_{A \to I}(S, ID, s_{ID}, SK_A, \pi)$ For inputs of a set of attributes S, an IBE identity ID, IBE public parameter π, an IBE additional secret information to generate re-encryption key s_{ID}, and an ABE secret key SK_A, outputs a re-encrypt key $RK_{A \to I}$ to the proxy.

$ReEncrypt_{A \to I}(RK_{A \to I}, C_A)$ For inputs of a re-encrypt key $RK_{A \to I}$ and an ABE ciphertext C_A, the proxy re-encrypts and outputs a IBE ciphertext C_I to the delegatee.

3 Chosen Plaintext Security for [ABE-IBE] Type Proxy Re-encryption

We define chosen plaintext security for [ABE-IBE] type proxy re-encryption scheme according to the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

We design the following game on the basis of Cheung and Newport's CPA Security Game for ABE in [12], Boneh and Boyen's selective ID game in [8] and Green and Ateniese's proxy re-encryption game [15]. We show even if an adversary obtains additional informations related to proxy re-encryption, such as re-encryption keys, they does not affect the security of underlying ABE and IBE scheme. In the challenge phase, the adversary can adaptively select which scheme to attack (ABE or IBE), this implies that these two schemes which are combined by our scheme, secure against chosen plaintext attacks.

In the following game, the adversary is allowed to adaptively conduct ABE secret key queries, IBE secret key queries and re-encryption key queries. Following the claim of Green and Ateniese, the adversary must not be restricted to obtain re-encryption keys which can convert the target ciphertext to a ciphertext if the adversary cannot decrypt it, in [15]. In other words, the adversary was only restricted to obtain the set of secret keys which can decrypt the target ciphertext.

Hence, in our security definition, the adversary is restricted to obtain re-encryption keys which can convert a target ABE ciphertext to an identity whose IBE secret key is already queried (and answered). In this case, the adversary can convert the target ABE ciphertext to a ciphertext which can be decrypted by the IBE key which is the adversary already obtains. The adversary is also restricted to obtain an IBE secret key, if the adversary already obtain re-encryption key which can convert the target ABE ciphertext to that identity.

Definition 2. *(Security of [ABE-IBE] type proxy re-encryption) The security against [ABE-IBE] type proxy re-encryption scheme is defined according to the following game between an attacker $\mathcal{A}$ and a challenger $\mathcal{C}$.*

Init : $\mathcal{A}$ chooses the following and sends them to $\mathcal{C}$.
- The target access structure W.
- The target IBE identity ID^*.

SetUp : $\mathcal{C}$ runs the $Setup_A(1^\kappa)$ and $Setup_I(1^\kappa)$. $\mathcal{C}$ gives ABE public parameters and IBE public parameters to the $\mathcal{A}$.

Phase 1 :

$Extract_A(S)$: $\mathcal{A}$ can adaptively request an ABE secret key for a set S where $S \not\models W$. $\mathcal{A}$ can repeat this multiple times.

$Extract_I(ID, params)$: $\mathcal{A}$ can adaptively request an IBE secret key corresponding to an identity ID of his choice. $\mathcal{A}$ can repeat this multiple times for different IBE identities.

$Extract_{A\to I}(S, ID)$: $\mathcal{A}$ can adaptively request re-encryption key which can transform ABE ciphertexts encrypted for set S to IBE ciphertexts corresponding to an identity ID. $\mathcal{A}$ can repeat this multiple times for different sets and identities.

Challenge : $\mathcal{A}$ submits two equal length messages M_0 and M_1 and selects which scheme to attack(ABE or IBE). $\mathcal{C}$ flips a coin $\mu \in \{0,1\}$ and returns the encrypted result of M_μ encrypted by the selected scheme.

Phase 2 : Same as Phase 1.

Solve : $\mathcal{A}$ submits a guess $\mu' \in \{0,1\}$ for μ. The adversary $\mathcal{A}$ wins if $\mu' = \mu$.

During Phase 1 and 2, $\mathcal{A}$ is restricted the following queries which $\mathcal{A}$ can decrypt a challenge ciphertext only using $\mathcal{C}$'s answers

- $Extract_A(S^*)$, where $S^* \models W$.
- $Extract_I(ID^*)$.
- The set of queries $Extract_{A\to I}(S^*, ID)$ and $Extract_I(ID, param)$, where $S^* \models W$ and ID is an identity of IBE user.

Definition 3. *Let $\mathcal{A}$ be an adversary against our scheme. We define the IND-sAttr-CPA advantage of $\mathcal{A}$ is $Adv_{\mathcal{A}}(\kappa) = 2(\Pr[\mu' = \mu] - 1/2)$.*

We say that the our scheme is (κ, t, q, ϵ) adaptive chosen plaintext secure if for any t-time adversary $\mathcal{A}$ that makes at most q chosen queries under a security parameter κ, we have that $Adv_{\mathcal{A}}(\kappa) < \epsilon$.

4 Construction

We construct an [ABE-IBE] type proxy re-encryption scheme which achieves CPA-security without random oracle.

Our scheme enables conversion of an ABE ciphertext to an IBE ciphertext. Our construction is based on Basic Construction of Cheung and Newport ABE scheme proposed in [12], because [12] is proved DBDH assumption and we combine the security of ABE and IBE schemes which bridged by our re-encryption method under single assumption. Furthermore, we use [8] as IBE scheme which achieves selective ID security under DBDH assumption.

4.1 CN-ABE Scheme

Let the set of attributes $\mathcal{N} = \{1, \cdots, n\}$ for some natural number n. We refer to attributes i and their negations $\neg i$ as *literals*. In [12], access structure consists of a single *AND* gate whose input are literals and denoted as $\bigwedge_{i \in I} \underline{i}$, where $I \subseteq \mathcal{N}$ and every $\underline{i}$ is a literal(i.e. i or $\neg i$).

$SetUp_A(1^\kappa)$ Let $\mathbb{G}, \mathbb{G}_T$ be a bilinear group of prime order p. Let $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be the bilinear map. Choose a random generator $g \in \mathbb{G}$ and $t_0, t_1, \cdots, t_{3n} \in_R \mathbb{Z}_p$. Let $T_0 = \hat{e}(g, g)^{t_0}$ and $T_j = g^{t_j}$ for each $j \in \{1, \cdots, 3n\}$. The public key is $PK_A = \langle \hat{e}, g, T_0, T_1, \cdots T_{3n} \rangle$. The master secret key is $MK_A = \langle t_0, t_1, \cdots, t_{3n} \rangle$.
The public key element T_i, T_{n+i} and T_{2n+i} correspond to the three types of occurrences of i: positive, $n + i$: negative and $2n + i$: *don't care* for each $i \in \{1, \cdots, n\}$.

$Encrypt_A(M, W)$ Given a message $M \in \mathbb{G}_T$ and an *AND* gate $W = \bigwedge_{i \in I} \underline{i}$ as input, select a random element $s \in \mathbb{Z}_p$ and sets $\widetilde{C} = M \cdot T_0^s$ and $\widehat{C} = g^s$. For each $i \in I$, set C_i as follows: If $i \in I$ and $\underline{i} = i$, set $C_i = T_i^s$. If $i \in I$ and $\underline{i} = \neg i$, set $C_i = T_{n+i}^s$. For each $i \in \mathcal{N} \setminus I$, set $C_i = T_{2n+i}^s$. The ciphertext is $C_A = \left\langle W, \widetilde{C}, \widehat{C}, \{C_i | i \in \mathcal{N}\} \right\rangle$.

$KeyGen_A(S, MK_A)$ Given the attribute set S and ABE master secret key MK_A as input, output a secret key $SK_A = \left\langle \widehat{D}, \{\langle D_i, F_i \rangle \,| i \in \mathcal{N}\} \right\rangle$ as follows:
Select $r_i \in_R \mathbb{Z}_p$ for every $i \in \mathcal{N}$ and set $r = \sum_{i=1}^n r_i$.

1. Set $\widehat{D} = g^{t_0 - r}$.
2. For each $i \in \mathcal{N}$, set D_i as follows:
   If $i \in S$, set $D_i = g^{\frac{r_i}{t_i}}$. If $i \notin S$, set $D_i = g^{\frac{r_i}{t_{n+i}}}$. Note that every $i \in S$ represents a positive attribute and $i \notin S$ represents a negative attribute.
3. For every $i \in \mathcal{N}$, set $F_i = g^{\frac{r_i}{t_{2n+i}}}$.

$Decrypt_A(SK_A, C_A)$ Given an ABE secret key SK_A and an ABE ciphertext C_A as input, if an ABE secret key SK_A can satisfy *AND* gate W in the ABE ciphertext C_A, output a plaintext as follows:

1. For each $i \in \mathcal{N}$, compute C'_i as follows:

$$\begin{cases} \text{If } i \in I \wedge \underline{i} = i \wedge i \in S: & C'_i = \hat{e}(C_i, D_i) = \hat{e}(g^{t_i s}, g^{\frac{r_i}{t_i}}) = \hat{e}(g,g)^{r_i \cdot s} \\ \text{If } i \in I \wedge \underline{i} = \neg i \wedge i \notin S: & C'_i = \hat{e}(C_i, D_i) = \hat{e}(g^{t_{n+i} s}, g^{\frac{r_i}{t_{n+i}}}) = \hat{e}(g,g)^{r_i \cdot s} \\ \text{If } i \notin I: & C'_i = \hat{e}(C_i, F_i) = \hat{e}(g^{t_{2n+i} s}, g^{\frac{r_i}{t_{2n+i}}}) = \hat{e}(g,g)^{r_i \cdot s} \end{cases}$$

2. Output a plaintext

$$\frac{\widetilde{C}}{\hat{e}\left(\widehat{C}, \widehat{D}\right) \cdot \prod_{i=1}^n C'_i} = \frac{M \cdot \hat{e}(g,g)^{t_0 \cdot s}}{\hat{e}(g^s, g^{t_0 - r}) \cdot \hat{e}(g,g)^{r \cdot s}} = M.$$

4.2 BB-IBE Scheme

We show BB-IBE [8] construction as follows:

SetUp$_I(1^\kappa)$ Let $\mathbb{G}, \mathbb{G}_T$ be a bilinear group of prime order p, and $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be the bilinear map.Given a security parameter 1^κ as input, select a random generator $g \in \mathbb{G}$ and $h, g_2 \in_R \mathbb{G}$. Pick $\alpha \in_R \mathbb{Z}_p^*$ and set $g_1 = g^\alpha$,$MK_I = \alpha$ and set $\pi = \langle g, g_1, g_2, h\rangle$.
Let MK_I be a master secret key, and π be the public parameters.

KeyGen$_I(MK_I, \pi, ID)$ Given master secret key $MK_I = \alpha$, public parameters π and an identity ID as input, the PKG picks $u \in_R \mathbb{Z}_p^*$ and output an IBE secret key as $SK_I = \langle sk_1^I, sk_2^I\rangle = \langle g_2^\alpha \left(g_1^{ID} h\right)^u, g^u\rangle$.

Encrypt$_I(ID, \pi, M)$ Given an identity ID, public parameter π and plaintext $M \in \mathbb{G}_T$ as input, select $w \in_R \mathbb{Z}_p^*$ and output an IBE ciphertext C_I.

$$C_I = \langle C_1, C_2, C_3\rangle = \left\langle g^w, \left(g_1^{ID} h\right)^w, M\hat{e}(g_1, g_2)^w \right\rangle.$$

Decrypt$_I(SK_I, \pi, C_I)$ Given an IBE secret key SK_I, public parameters π and an IBE ciphertext C_I as input, output a plaintext M.

$$M = \frac{C_3 \hat{e}(sk_2^I, C_2)}{\hat{e}(sk_1^I, C_1)}.$$

4.3 [ABE-IBE] Type Proxy Re-encryption

KenGen$_{A\to I}(S, ID, SK_A, \pi, sk_2^I)$ Given the attribute set S, a delegatee's IBE identity ID, a delegator's ABE secret key SK_A, IBE public parameter π and an IBE user's 2nd component of secret key[1] sk_2^I as input, output a re-encryption key $RK_{A\to I} = \left\langle \widehat{R}_a, \widehat{R}_b, \widehat{R}_c, \widehat{R}_d,\ \ \{\langle R_i, Q_i\rangle \,|i \in \mathcal{N}\}\right\rangle$ as follows:

1. Set $\widehat{R}_a = \widehat{D} \cdot sk_2^I = g^{t_0 - r} g^u$.
2. Select $\tau \in_R \mathbb{Z}_p$ and set $\widehat{R}_b = \left(g_1^{ID} h\right)^\tau$, $\widehat{R}_c = g^\tau$, $\widehat{R}_d = \hat{e}\left(g_1, g_2\right)^\tau$.
3. Select $\delta_i \in_R \mathbb{Z}_p$ for every $i \in \mathcal{N}$, set R_i as follows:

$$\begin{cases} \text{If } i \in S: & R_i = \langle r_{i,1}, r_{i,2}\rangle = \left\langle D_i \cdot g^{\delta_i}, T_i^{\delta_i} \right\rangle \\ \text{If } i \notin S: & R_i = \langle r_{i,1}, r_{i,2}\rangle = \left\langle D_i \cdot g^{\delta_i}, T_{n+i}^{\delta_i} \right\rangle \end{cases}$$

4. For every $i \in \mathcal{N}$, set Q_i as follows:

$$Q_i = \langle q_{i,1}, q_{i,2}\rangle = \left\langle F_i \cdot g^{\delta_i}, T_{2n+i}^{\delta_i} \right\rangle.$$

ReEncrypt$_{A\to I}(RK_{A\to I}, C_A)$ Given a re-encryption key $RK_{A\to I}$ and an ABE ciphertext $C_A = \left\langle W, \widetilde{C}, \widehat{C}, \{C_i | i \in \mathcal{N}\} \right\rangle$ as input, output an IBE ciphertext C_I as follows:

[1] In our [ABE-IBE] type proxy re-encryption scheme, an IBE user must pass own second component of secret key sk_2^I to the ABE user, however this property does not affect security of [8]. This property proved in Lemma 1 of [20].

1. For each $i \in \mathcal{N}$ compute $\overline{C}_i$ as follows:

$$\begin{cases} \text{If } i \in I \wedge \underline{i} = i \wedge i \in S: & \overline{C}_i = \frac{\hat{e}(C_i, r_{i,1})}{\hat{e}\left(r_{i,2}, \widehat{C}\right)} = \hat{e}(g,g)^{r_i \cdot s} \\ \text{If } i \in I \wedge \underline{i} = \neg i \wedge i \notin S: & \overline{C}_i = \frac{\hat{e}(C_i, r_{i,1})}{\hat{e}\left(r_{i,2}, \widehat{C}\right)} = \hat{e}(g,g)^{r_i \cdot s} \\ \text{If } i \notin I: & \overline{C}_i = \frac{\hat{e}(C_i, q_{i,1})}{\hat{e}\left(q_{i,2}, \widehat{C}\right)} = \hat{e}(g,g)^{r_i \cdot s} \end{cases}$$

2. Select $y \in_R \mathbb{Z}_p$ and output $C_I = \langle C_1, C_2, C_3 \rangle$ as follows:

$$\langle C_1, C_2, C_3 \rangle = \left\langle \left(\widehat{R}_c\right)^y, \left(\widehat{R}_b\right)^y \widehat{C}, \frac{\widetilde{C} \cdot \left(\widehat{R}_d\right)^y}{\hat{e}\left(\widehat{C}, \widehat{R}_a\right) \cdot \left(\prod_{i=1}^{n} \overline{C}_i\right)} \right\rangle.$$

Note that, the delegatee can decrypt this re-encrypted result C_I using his own secret key SK_I with same IBE decryption algorithm as follows:

$$\frac{C_3 \hat{e}(sk_2^I, C_2)}{\hat{e}(sk_1^I, C_1)} = \frac{\widetilde{C} \cdot \left(\widehat{R}_d\right)^y \hat{e}\left(sk_2^I, \left(\widehat{R}_b\right)^y \widehat{C}\right)}{\hat{e}\left(\widehat{C}, \widehat{R}_a\right) \cdot \prod_{i=1}^{n} \overline{C}_i \cdot \hat{e}\left(sk_1^I, \left(\widehat{R}_c\right)^y\right)}$$

$$= \frac{M \cdot \left(e\left(g,g\right)^{t_0}\right)^s \hat{e}\left(g_1, g_2\right)^{y\tau} \hat{e}\left(g^u, \left(g_1^{ID} h\right)^{y\tau} g^s\right)}{\hat{e}\left(g^s, g^{t_0 - r} g^u\right) \cdot \prod_{i=1}^{n} \hat{e}\left(g,g\right)^{r_i \cdot s} \cdot \hat{e}\left(g_2^{\alpha} \left(g_1^{ID} h\right)^u, g^{y\tau}\right)} = M$$

5 Security

We next show that ABE-PRE is IND-sAttr-CPA, if the Decisional BDH problem holds in $\mathbb{G}, \mathbb{G}_T$.

Theorem 1. *Suppose that the $(\kappa, t, \epsilon) - DBDH$ assumption holds in $(\mathbb{G}, \mathbb{G}_T)$. Then, the ABE-PRE is $(\kappa, t', q, \epsilon)$-IND-sAttr-CPA secure against an adversary for any (q, κ, ϵ) and $t' < t - \Theta(\tau q)$, where τ denotes a maximum time for exponentiation in $\mathbb{G}, \mathbb{G}_T$.*

Proof. Let $\mathcal{A}$ be a (t, q, ϵ) adversary against the [ABE-IBE] type proxy re-encryption scheme(ABE-IBE-PRE). We construct an adversary $\mathcal{B}$ which can solve the DBDH problem in $\mathbb{G}$ by using $\mathcal{A}$. The $\mathcal{B}$ is given an input $\langle g, \varGamma_a, \varGamma_b, \varGamma_c, \varGamma_T \rangle = \left\langle g, g^a, g^b, g^c, \varGamma_T \right\rangle$, and distinguishes $\varGamma_T$ is $\hat{e}(g,g)^{abc}$ or $\varGamma_T \in_R \mathbb{G}_T$. $\mathcal{B}$ works as follows:

Init $\mathcal{A}$ chooses the following and sends it to $\mathcal{B}$.
- The challenge access structure $W = \bigwedge_{i \in I} \underline{i}$
- The challenge IBE identity ID^*

Setup $\mathcal{B}$ setup simulation as follows:

List SetUp $\mathcal{B}$ generates three blank lists to store a query and answer pairs for every queries, and setup ABE, IBE as follows:

Table 1. ABE Public Key in CPA simulation

	$i \in I$		$i \notin I$
	$\underline{i} = i$	$\underline{i} = \neg i$	
T_i	g^{α_i}	$\Gamma_b^{\alpha_i}$	$\Gamma_b^{\alpha_i}$
T_{n+i}	$\Gamma_b^{\beta_i}$	g^{β_i}	$\Gamma_b^{\beta_i}$
T_{2n+i}	$\Gamma_b^{\gamma_i}$	$\Gamma_b^{\gamma_i}$	g^{γ_i}

ISKL (IBE Secret Key List): Record the tuple $\langle ID, SK_I \rangle$, where ID is an identity of IBE user, SK_I are IBE secret key corresponding to ID.

ASKL (ABE Secret Key List): Record the tuple $\langle S, \ SK_A \rangle$, where S is a set of attributes and SK_A are ABE secret key corresponding to set S.

REKL (Re-Encryption Key List for IBE): Record the tuple $\langle S, \ ID, \ RK_{A \to I} \rangle$, where S is a set of attributes, ID is an identity of IBE user, $RK_{A \to I}$ is a re-encryption key.

ABE SetUp $\mathcal{B}$ sets $T_0 = \hat{e}(\Gamma_a, \Gamma_b) = \hat{e}(g, g)^{ab}$ and chooses $\alpha_i, \beta_i, \gamma_i \in_R \mathbb{Z}_p$ for each $i \in \mathcal{N}$. Then set ABE public key components T_i, T_{n+i} and T_{2n+i} as Table 1.

Under this condition, the first component of ABE master secret key is ab which $\mathcal{B}$ cannot compute.

IBE SetUp Then $\mathcal{B}$ generates random numbers $z_1, z_2, z_3 \in_R \mathbb{Z}_p^*$ and sets $g_1 = \Gamma_a^{z_1}$, $g_2 = \Gamma_b^{z_2}$, $h = g_1^{-ID^*} g^{z_3}$. $\mathcal{B}$ provides public parameters $\pi = \langle g, g_1, g_2, h \rangle$ to $\mathcal{A}$. Under this condition, the master secret key is $MK_I = az_1$ which $\mathcal{B}$ cannot compute.

Phase 1 $\mathcal{A}$ adaptively queries $\mathcal{B}$, and $\mathcal{B}$ responds as follows:

Extract$_A(S)$ $\mathcal{A}$ queries the ABE secret key SK_A, $\mathcal{B}$ as follows:

$\mathcal{A}$ queries the ABE secret key SK_A with a set $S \subseteq \mathcal{N}$ where $S \not\models W$. There must exist $j \in I$ such that, either $j \in S \wedge \underline{j} = \neg j$ or $j \notin S \wedge \underline{j} = j$. $\mathcal{B}$ chooses such j. Without loss of generality, we can assume that $j \notin S \wedge \underline{j} = j$.

For every $i \in \mathcal{N}$, $\mathcal{B}$ chooses $r_i' \in_R \mathbb{Z}_p$. Then sets $r_j = ab + r_j' b$ and for every $i \neq j, i \in \mathcal{N}$,$r_i = r_i' b$. Finally $\mathcal{B}$ sets $r = \sum_{i=1}^n r_i = ab + \sum_{i=1}^n r_i' b$. The $\widehat{D}$ component of the secret key can be computed as

$$\prod\nolimits_{i=1}^n \frac{1}{\Gamma_b^{r_i'}} = g^{-\sum_{i=1}^n r_i' b} = g^{ab-r}.$$

Recall that $j \in I \setminus S \wedge \underline{j} = j$, then $D_j = \Gamma_a^{\frac{1}{\beta_j}} g^{\frac{r_j'}{\beta_j}} = g^{\frac{ab + r_j' b}{b\beta_j}} = g^{\frac{r_j}{b\beta_j}}$.

For each $i \in I \wedge i \neq j$, ABE secret key components D_i can be computed as follows:

$$\begin{cases} \text{If } i \in S \wedge i \in I \wedge \underline{i} = i: & D_i = \Gamma_b^{\frac{r_i'}{\alpha_i}} = g^{\frac{r_i}{\alpha_i}}. \\ \text{If } i \in S \wedge ((i \in I \wedge \underline{i} = \neg i) \vee i \notin I): & D_i = g^{\frac{r_i'}{\alpha_i}} = g^{\frac{r_i}{b\alpha_i}}. \\ \text{If } i \notin S \wedge ((i \in I \wedge \underline{i} = i) \vee i \notin I): & D_i = g^{\frac{r_i'}{\beta_i}} = g^{\frac{r_i}{b\beta_i}}. \\ \text{If } i \notin S \wedge i \in I \wedge \underline{i} = \neg i: & D_i = \Gamma_b^{\frac{r_i'}{\beta_i}} = g^{\frac{r_i}{\beta_i}}. \end{cases}$$

The ABE secret key components F_i (for *don't care* attribute) can be computed as follows:

1. If $i = j$

$$F_j = \Gamma_a^{\frac{1}{\gamma_j}} g^{\frac{r'_j}{\gamma_j}} = g^{\frac{ab+r'_j b}{b\gamma_j}} = g^{\frac{r_j}{b\gamma_j}}.$$

2. Otherwise $(i \neq j)$

$$\begin{cases} \text{If } i \in I: & F_i = g^{\frac{r'_i}{\gamma_i}} = g^{\frac{r_i}{b\gamma_i}} \\ \text{If } i \notin I: & F_i = \Gamma_b^{\frac{r'_i}{\gamma_i}} = g^{\frac{r_i}{\gamma_i}} \end{cases}$$

$\mathcal{B}$ answers SK_A and writes down to the $ASKL$.

$Extract_I(ID)$ $\mathcal{A}$ queries the IBE user's secret key SK_I with an identity ID.

1. If the $ID = ID^*$, $\mathcal{B}$ rejects.
2. If the $ID \neq ID^*$,$\mathcal{B}$ checks the $REKL$, and if already answers re-encryption key to the ID and $S \models W$, $\mathcal{B}$ rejects.
3. Otherwise $\mathcal{B}$ answers $SK_I = \langle sk_1^I, sk_2^I \rangle$ as follows. $\mathcal{B}_I$ generates a random number $u \in_R \mathbb{Z}_p^*$ and computes $SK_I = \langle sk_1^I, sk_2^I \rangle$ as follows:

$$sk_1^I = \Gamma_b^{\frac{-z_2 z_3}{(ID-ID^*)}} \left(\Gamma_a^{z_1(ID-ID^*)} g^{z_3} \right)^u, sk_2^I = \Gamma_b^{\frac{-z_2}{(ID-ID^*)}} g^u$$

$\mathcal{B}$ answers SK_I and writes down to the $ISKL$.

$Extract_{A\to I}(S, ID)$ $\mathcal{A}$ queries the re-encryption key which can transform ABE ciphertext corresponding to the set S, $\mathcal{B}$ answers a re-encrypt key $RK_{A\to I} = \left\langle \widehat{R}_a, \widehat{R}_b, \widehat{R}_c, \widehat{R}_d, \{\langle R_i, Q_i \rangle | i \in \mathcal{N}\} \right\rangle$ as follows:

1. If $S \not\models W$,
   $\mathcal{B}$ runs $Extract_A(S)$ and obtain an ABE secret key $SK_A = \left\langle \widehat{D}, \{\langle D_i, F_i \rangle | i \in \mathcal{N}\} \right\rangle$, then chooses $\tau, u \in_R \mathbb{Z}_p$ for every $i \in \mathcal{N}$.
   (a) If $ID \neq ID^*$, $\mathcal{B}$ sets re-encryption key components $\widehat{R}_a$, $\widehat{R}_b$, $\widehat{R}_c$, $\widehat{R}_d$ as follows:

$$\widehat{R}_a = \widehat{D}\Gamma_b^{\frac{-z_2}{ID-ID^*}} g^u, \widehat{R}_b = \left(\Gamma_a^{z_1(ID-ID^*)} g^{z_3} \right)^\tau,$$
$$\widehat{R}_c = g^\tau, \widehat{R}_d = \hat{e}\left(\Gamma_a^{z_1}, \Gamma_b^{z_2}\right)^\tau$$

   Note that, let $u' = \frac{-bz_2}{ID-ID^*}$, simulated $\widehat{R}_a$ and $\widehat{R}_b$ can be transform as follows:

$$\widehat{R}_a = \widehat{D}\Gamma_b^{\frac{-z_2}{ID-ID^*}} g^u = g^{u'},$$
$$\widehat{R}_b = \left(\Gamma_a^{z_1(ID-ID^*)} g^{z_3} \right)^\tau = \left(\Gamma_a^{z_1 ID} h \right)^\tau = \left(g_1^{ID} h \right)^\tau$$

   (b) If $ID = ID^*$, $\mathcal{B}$ sets re-encryption key components $\widehat{R}_a$, $\widehat{R}_b$, $\widehat{R}_c$, $\widehat{R}_d$ as follows:

$$\widehat{R}_a = \widehat{D}g^u, \widehat{R}_b = \left(g^{z_3}\right)^\tau, \widehat{R}_c = g^\tau, \widehat{R}_d = \hat{e}\left(\Gamma_a^{z_1}, \Gamma_b^{z_2}\right)^\tau$$

Note that, simulated $\widehat{R}_b$ can be transform as follows:

$$\widehat{R}_b = (g^{z_3})^{\tau} = \left(g_1^{ID^*} g_1^{-ID^*} g^{z_3}\right)^{\tau} = \left(g_1^{ID^*} h\right)^{\tau}$$

After the success of above simulation, $\mathcal{B}$ chooses $\delta_i \in_R \mathbb{Z}_p$ for every $i \in \mathcal{N}$and $\phi \in_R \mathbb{Z}_p$ and sets re-encryption key components R_i, Q_i as follows:

$$R_i = \begin{cases} i \in S: & \left\langle D_i \cdot g^{\delta_i}, T_i^{\delta_i} \right\rangle \\ i \neq\in S: & \left\langle D_i \cdot g^{\delta_i}, T_{n+i}^{\delta_i} \right\rangle \end{cases}$$
$$Q_i = \left\langle F_i \cdot g^{\delta_i}, T_{2n+i}^{\delta_i} \right\rangle$$

2. Otherwise ($S \models W$),
   $\mathcal{B}$ chooses $\rho_i \in_R \mathbb{Z}_p$ for every $i \in \mathbb{N}$ and sets $\rho = \sum_{i=1}^{n} \rho_i \bmod p$. $\mathcal{B}$ chooses $\tau \in_R \mathbb{Z}_p$ for every $i \in \mathcal{N}$.
   (a) $\mathcal{B}$ checks, and if already answers IBE secret key for ID, $\mathcal{B}$ rejects.
   (b) If $ID \neq ID^*$, $\mathcal{B}$ sets re-encryption key components $\widehat{R}_a$, $\widehat{R}_b$, $\widehat{R}_c$, $\widehat{R}_d$ as follows:

$$\widehat{R}_a = \Gamma_b^{-\rho} \Gamma_b^{\frac{-z_2 u}{ID-ID^*}}, \widehat{R}_b = \left(\Gamma_a^{z_1(ID-ID^*)} g^{z_3}\right)^{\tau},$$
$$\widehat{R}_c = g^{\tau}, \widehat{R}_d = \hat{e}\left(\Gamma_a^{z_1}, \Gamma_b^{z_2}\right)^{\tau}$$

   Note that, let $u' = -ab - \frac{buz_2}{ID-ID^*}$ and $r' = b\rho$, simulated $\widehat{R}_a$,$\widehat{R}_b$ can be transform as follows:

$$\widehat{R}_a = \Gamma_b^{-\rho} \Gamma_b^{\frac{-z_2 u}{ID-ID^*}} = g^{ab-b\rho} g^{-ab} g^{\frac{-buz_2}{ID-ID^*}} = g^{ab-r'} g^{u'},$$
$$\widehat{R}_b = \left(\Gamma_a^{z_1(ID-ID^*)} g^{z_3}\right)^{\tau} = \left(\Gamma_a^{z_1 ID} h\right)^{\tau} = \left(g_1^{ID} h\right)^{\tau}$$

   Under this condition, $\mathcal{B}$ cannot compute an IBE secret key for ID, however $\mathcal{B}$ can reject the IBE secret key query for the ID.
   (c) If $ID = ID^*$, $\mathcal{B}$ sets re-encryption key components $\widehat{R}_a$, $\widehat{R}_b$, $\widehat{R}_c$, $\widehat{R}_d$ as follows:

$$\widehat{R}_a = g^{-\rho}, \widehat{R}_b = (g^{z_3})^{\tau}, \widehat{R}_c = g^{\tau}, \widehat{R}_d = \hat{e}\left(\Gamma_a^{z_1}, \Gamma_b^{z_2}\right)^{\tau}$$

   Note that, let $u' - r = -ab - \rho$, simulated $\widehat{R}_a$ can be transform as follows:

$$\widehat{R}_a = g^{-\rho} = g^{ab-\rho} g^{-ab} = g^{ab-r} g^{u'},$$
$$\widehat{R}_b = (g^{z_3})^{\tau} = \left(g_1^{ID^*} g_1^{-ID^*} g^{z_3}\right)^{\tau} = \left(g_1^{ID^*} h\right)^{\tau}$$

   Under this condition, $\mathcal{B}$ cannot compute an IBE secret key for ID^*, however $\mathcal{B}$ can reject the IBE secret key for the ID^*.

After the success of above simulation, $\mathcal{B}$ chooses $\delta_i \in_R \mathbb{Z}_p$ for every $i \in \mathcal{N}$and $\phi \in_R \mathbb{Z}_p$ and sets re-encryption key components R_i, Q_i as follows:

$$R_i = \begin{cases} i \in S: & \left\langle g^{\frac{\rho_i}{\alpha_i}} \cdot g^{\delta_i}, T_i^{\delta_i} \right\rangle \\ i \notin S: & \left\langle g^{\frac{\rho_i}{\beta_i}} \cdot g^{\delta_i}, T_{n+i}^{\delta_i} \right\rangle \end{cases}$$

$$Q_i = \left\langle g^{\frac{\rho_i}{\gamma_i}} \cdot g^{\delta_i}, T_{2n+i}^{\delta_i} \right\rangle$$

Challenge. $\mathcal{A}$ submits two equal length plaintexts $M_0, M_1 \in \mathbb{G}_T$ and selects which scheme to attack. $\mathcal{B}$ chooses $\mu \in \{0,1\}$ and outputs a challenge ciphertext as follows:

- If $\mathcal{A}$ selects ABE scheme to attack, $\mathcal{B}$ outputs an ABE ciphertext for a challenge access structure $C_A^* = \left\langle \widetilde{C}^*, \widehat{C}^*, \{C_i^* | i \in I\} \right\rangle$ as follows:

$$\widetilde{C}^* = M_\mu \varGamma_T, \widehat{C}^* = \varGamma_c,$$
$$C_i^* = \left\{ \{\varGamma_c^{\alpha_i} | i \in I \wedge \underline{i} = i\}, \{\varGamma_c^{\beta_i} | i \in I \wedge \underline{i} = \neg i\}, \{\varGamma_c^{\gamma_i} | i \notin I\} \right\}$$

- If $\mathcal{A}$ selects IBE scheme to attack, $\mathcal{B}$ outputs a IBE ciphertext $C_I^* = \langle C_1, C_2, C_3 \rangle$ corresponding to a target identity ID^* as follows:

$$C_1 = \varGamma_c, C_2 = (\varGamma_c)^{z_3}, C_3 = M_\mu (\varGamma_T)^{z_1 z_2}$$

Phase 2. $\mathcal{B}$ answers $\mathcal{A}$'s queries in same manner of Phase 1.

Solve. Finally, $\mathcal{A}$ outputs a guess result $\mu' \in \{0,1\}$. If $\mu' = \mu$, then $\mathcal{B}$ judges $\varGamma_T = \hat{e}(g,g)^{abc}$ and outputs 1. Otherwise, $\mathcal{B}$ judges $\varGamma_T \in_R \mathbb{G}_T$ and outputs 0.

We claim that in the above simulation answers of $\mathcal{B}$ are correctly distributed, and $\mathcal{A}$ cannot distinguish our simulation from the real-world interaction. Furthermore, $Adv_{\mathcal{A}}^{DBDH} = Adv_{\mathcal{A}}^{\mathcal{S}}$, because $\mathcal{B}$ does not abort during the above simulation.

In the above simulation, maximum computation cost of the queries is at most polynomial time exponentiation, hence $t' < t - \varTheta(\tau q)$. Therefore, the ABE-IBE-PRE is $(\kappa, t', q, \epsilon)$-IND-sAttr-CPA secure against an adversary.

6 Extension

In our scheme, the delegator can delegate a part of his decryption rights. The delegator can pass subset of the ABE secret key SK'_A and subset of S' to generate a re-encryption key for substitute of SK_A. The subset of the ABE secret key SK'_A at least have a *positive* or *negative* or *don't care* component for each attribute. On the other hand, the full set of the ABE secret key SK_A have a *positive* or *negative* component and *don't care* component for each attribute.

For example, if the delegator passes *positive* component and does not passes *don't care* component for some attribute, then the proxy cannot convert ciphertexts which have *don't care* policy for the attribute.

7 Conclusion

In this paper, we propose new proxy re-encryption scheme which can convert an ABE ciphertext to an IBE ciphertext. Our scheme achieves proxy invisible which means delegatee does not aware of existence of the proxy. We define the security notation and prove security based on DBDH assumption in the standard model against chosen plaintext attack. To achieve the CCA security and adaptive-ID security is further study. However, it should be possible to change [8] to [4] to achieve adaptive ID security.

Furthermore, [ABE-PKE] type proxy, and conbination of types such as [IBE-ABE] and [PKE-ABE], [ABE-ABE] might be useful, but it is also further study.

Acknowledgement

We thank the anonymous reviewers for many useful comments.

References

1. Joux, A.: A one round protocol for tripartite diffie-hellman. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 385–394. Springer, Heidelberg (2000)
2. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
3. 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)
4. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
5. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: Cryptology ePrint Archive, Report 2008/290 (2008), `http://eprint.iacr.org/2008/290.pdf`
6. Chu, C., Tzeng, W.: Identity-based proxy re-encryption without random oracles. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 189–202. Springer, Heidelberg (2007)
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. Boneh, D., Boyen, X.: Efficient selectiveid secure identity based encryption without random oracle. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
9. Ateniese, G., Benson, K., Hohenberger, S.: Key-private proxy re-encryption. In: Cryptology ePrint Archive, Report 2008/463 (2008), `http://eprint.iacr.org/2008/463.pdf`
10. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: Proceedings of the 12th Annual Network and Distributed System Security Symposium - NDSS 2005, pp. 83–107 (2005)

11. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: Proc. IEEE Symposium on Security and Privacy, pp. 321–334. IEEE, Los Alamitos (2007)
12. Cheung, L., Newport, C.: Provably secure ciphertext policy abe. In: CCS 2007, pp. 456–465 (2007)
13. Zbou, L., Marsh, M.A., Schneider, F.B., Redz, A.: Distributed blinding for elgamal reencryption. Technical Report 2004-1924. Cornell Computer Science Department (2004)
14. 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)
15. Green, M., Ateniese, G.: Indentity-based proxy re-encryption. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 288–306. Springer, Heidelberg (2007)
16. Jakobsson, M.: On quorum controlled asymmetric proxy re-encryption. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 112–121. Springer, Heidelberg (1999)
17. Mambo, M., Okamoto, E.: Proxy cryptosystems: @delegation of the power to decrypt ciphertexts. IEICE Trans. Fund. Electronics Communications and Computer Science, IEICE E80-A/1, 54–63 (1997)
18. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: CCS 2007, pp. 185–194. ACM, New York (2007)
19. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairings. In: Proceedings of the Symposium on Cryptography and Information Security, SCIS 2000 (2000)
20. Matsuo, T.: Proxy re-encryption systems for identity-based encryption. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 247–267. Springer, Heidelberg (2007)
21. Mizuno, T., Doi, H.: Efficient ibe-pke proxy re-encryption. In: International Conference on Security and Cryptography (SECRYPT 2008), pp. 285–293. Insticc Press (2008)
22. 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 (2006)
23. Dodis, Y., Ivan, A.: Proxy cryptography revisited. In: Proceedings of the 10th Annual Network and Distributed System Security Symposium- NDSS 2003 (2003)

Constructing Better KEMs with Partial Message Recovery

Rui Zhang and Hideki Imai

Research Center for Information Security (RCIS)
National Institute of Advanced Industrial Science and Technology (AIST)
{r-zhang,h-imai}@aist.go.jp

Abstract. In this paper, we consider the problem of building efficient key encapsulation mechanism (KEM) with partial message recovery, in brief, PKEM, which aims at providing better bandwidth for standard KEM. We demonstrate several practical issues that were not considered by the previous research, e.g., the additional security loss due to loose reduction of OAEP, and the ciphertext overhead caused by the corresponding data encapsulation mechanism (DEM). We give solutions to these problems, furthermore, we consider the multi-challenge model for PKEMs, where an adversary can obtain up to multiple challenge ciphertexts. Apparently, this is a more severe and more realistic model for PKEM. We then show two generic constructions of PKEMs and prove their security in the multi-challenge model. Our constructions are natural and simple. Finally, we give some instantiations of our generic constructions, and compare their efficiency. Our results demonstrate that there are strong ties between PKEM and public key encryption.

1 Introduction

Background. With public key cryptosystems applied more and more widely in the mobile environment, bandwidth of public key encryption (PKE) schemes has become a prominent problem, since data transmissions usually consume a large proportion of power, even more than extra off-line computations [23]. Let us give some numerical explanations. Suppose one wants to encrypt a long message using the RSA key encapsulation mechanism (RSA-KEM), one of the ISO/IEC standards[1] for public key encryption [22] to have κ-bit security. Let the length of the plaintext m be $|m|$ bits, and that of the ciphertext c be $|c|$ bits. To have 128-bit security (as considered necessary nowadays), the ciphertext overhead (defined by $|c| - |m|$) is at least 3074 bits, according to recent NIST recommendation on parameter size [17] (Fig. 1). In the near future, when 256-bit security is a must, this overhead will soar to more than 15k bits!

One may alternatively consider to use elliptic curve cryptography (ECC) based schemes, e.g, ECIES-KEM [22], which archives about 2κ-bit overhead[2] with κ-bit

[1] We note that there is another key encapsulation mechanism (KEM) in the ISO standard, namely RSAES (a.k.a. RSA-OAEP) [6], however, it does not support long plaintexts. So we do not take it into account.

[2] The original DHIES [1] actually didn't achieve this, since the DEM part uses message authentication code (MAC). However it is easy to remove the MAC while maintaining the chosen ciphertext security by using a strong DEM [14].

Bits of Security (κ)	Size of RSA Modulus (R_κ)	Size of Elliptic Curve (E_κ)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Fig. 1. NIST recommendation of key sizes (in bits) for comparable security [17]

security. But adoption of ECC-based schemes often means rebuilding the current encryption module, which can be expensive and time-consuming for a practical system. Especially, considering the fact that RSA-based cryptographic modules are widely deployed in many systems, which are mature and with support from various industrial standards, ECC-based schemes does not always seem to be the best solution.

Recently, the problem of improving overhead of RSA-based KEMs was formally considered by Bjørstad, Dent and Smart [7]. Their key idea was to construct key encapsulation mechanisms with partial message recovery (PKEMs). They gave a formal security model based on the KEM/DEM framework [20] as well as the tag-KEM/DEM [2] framework, with two constructions based on RSA: one from OAEP, the other from OAEP+ [21] with feedbacks. As a result, the overhead of both constructions no longer depend on R_κ, but linear in κ.

Given the problem of constructing a PKEM, one may naturally consider to utilize secure public key encryption. Actually, this works! One just needs to encrypt $dk||m_0$ in the KEM ciphertext, where dk is the session key and m_0 is the partial message to be recovered. If the public key encryption is secure, it is possible to show this scheme achieves both message confidentiality and session key privacy. Note that by choosing a proper PKE scheme, one can achieve quite good efficiency, ever smaller ciphertext overhead than the best from [7], which use more intricate tag-KEM structure [2]. Throughout this paper, we don't consider the tag-KEM variant of PKEMs, since it requires feedbacks, thus is slower than the standard KEM/DEM construction [20].

Missing Points for the Previous Schemes. The OAEP PKEM was previously considered a practical PKEM [7], since it is compatible with OAEP. However, we couldn't find any rigorous security argument why the OAEP PKEM can achieve the redundancy of about 5κ bits in [7]. Because OAEP is only known to suffer from a quadratic security loss, due to the reduction from partial domain onewayness to full domain onewayness [9,10,15], it is quite unlikely this optimistic estimation can hold. Also in [7], combining with passively secure DEM with ciphertext aunthenticity, a secure PKEM yields a secure hybrid encryption. But such DEM will require additionally κ bits ciphertext overhead due to the best known construction of such DEMs. We emphasize that the PKEM is not tag-based in the construction, since otherwise this can be easily avoided. We then give a new composition theorem, which requires only a weaker DEM. The direct merit is this additional κ bits can be removed from the DEM.

The Multi-Challenge Model. In practice, an adversary can observe multiple ciphertexts, possibly chosen by itself, as discussed by [4,3]. It was then shown in [3] that for public key encryption, the security reduction will degrade as the number challenge queries increase, in general a factor of q_e, where q_e is the number of the challenge queries. Thus proving the security in the single-challenge model becomes sufficient. For practical applications, if q_e is estimated as 2^{40}, this actually implies that the analysis given in [7] is not valid in this stronger, however, more realistic scenario. As a result, one has to enlarge the public key size and the randomness size of the ciphertext to compensate this security loss, e.g., at least 40-bit additional randomness and a 3072-bit RSA modulus (for only 80-bit security) should be used, for the security result of the RSA-OAEP PKEM given in [7].

Additionally, we remark that even in the single challenge model, the analysis of OAEP-PKEM in [7] may be a little too optimistic, because κ has to be much longer than what was mentioned, due to the fact that the best known reduction of RSA-OAEP [10,15] suffers from a quadratic security loss. I.e., for κ-bit security, one has to adopt $R_{2\kappa}$-bit RSA modulus. In fact, it was already pointed out in [18], RSA-OEAP with a 1024-bit modulus only provides a security level of 2^{40}. Considering this security loss, one has to enlarge R_κ, which makes the scheme very inefficient.

To summarize, the results in [7] become not meaningful in the multi-challenge model, furthermore, as we will argue shortly after, even in the single challenge model, known PKEM schemes are not efficient enough, because there are a few important issues unconsidered in the previous research. So we naturally consider the following question, how to efficient build PKEM in the multi-challenge model?

1.1 Related Work

We note the idea of using the KEM to transport partial plaintext is not new: Shoup has mentioned a "long message mode" for RSA-OAEP+ KEM [20], which aimed at shortening the overhead of OAEP+ KEM when dealing with a long plaintext, but no formal security definition or analysis was given there. It seemed that the long message mode did not draw much attention, probably because the OAEP+ KEM didn't show up in the final ISO/IEC standard [22].

The standard security definition for public key encryption is indistinguishability against adaptive chosen ciphertext attack IND-CCA [11,16,19]. In such a security model, the adversary can query the challenge oracle only once, acquiring only one target challenge ciphertext, thus it is called the single-challenge model.

The multi-challenge (single receiver) model was first considered in [3], where the adversary can query the challenge oracle multiple (up to a number bounded by a polynomial of the security parameter) times, thus it is called the multi-challenge model. Apparently, the multi-challenge model is stronger. But in [3], a general reduction to from the multi-challenge to the single-challenge security was given, yet with noticeable security loss. It is then worth emphasizing that a loose security reduction means only less security is guaranteed under the same key size. So if a multi-challenge adversary is considered, one has to enlarge the key size to compensate the security loss due to the loose reduction.

Recently, Boldyreva [8] proved an RSA-based scheme (OAEP++), oringinally proposed by Kobara and Imai [13], achieves tight security in the multi-challenge model. We remark that only Diffie-Hellman based schemes were known to achieve tight security reduction in the multi-challenge model previously [3].

1.2 Our Contributions

Since most techniques used in this paper were well-known in the related literature, we view our primary contributions as confirming previous research on PKEMs and pointing out some practical issues, and giving alternative solutions. Though some RSA based schemes are in the random oracles, our generic constructions are without random oracles.

Especially, regarding security notions, we extend the single challenge model in [7], and define a multi-challenge model for PKEMs. We also give some constructions of PKEMs based on secure PKEs in the multi-challenge model. Such constructions are natural and simple. We note similar constructions were considered in [7] to build PKEMs in the single-challenge model. The short security arguments in [7] claimed that if the PKE scheme is IND-CCA secure [11,16,19] regarding m (the plaintext) and oneway against chosen plaintext attack (OW-CPA), the above PKEM scheme is secure. Their confidence comes from the proof of RSA-KEM [20], where $H(m||r)$ happens to be the session key for RSA-KEM. In this way, they have implicitly assmed that $m||r$ can be encoded as a plaintext of the underlying (deterministic) PKE (RSA encryption in this case), which is exactly the proof strategy for RSA-KEM [20]. We note that for general PKE with randomness recovery, e.g., McEliece Encryption, this may *not* necessarily be true. It is thus concluded that the claim of [7] requires further justification. In this paper, we investigate this point and give an affirmative answer. As an independent interest, we also improve the PKEM/DEM composition theorem of [7], in both the single challenge model and the multi-challenge model.

Finally, we show some instantiations of our generic constructions and compare their performances. We are thus confident that our generic constructions are efficient.

2 Preliminary

Notations. If x is a string, let $|x|$ denotes its length, while if S is a set then $|S|$ denotes its size. If S is a set then $s \leftarrow S$ denotes the operation of picking an element s of S uniformly at random. We write $z \leftarrow \mathcal{A}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $(x, y, \ldots)$ and an output z. Denote $x||y$ as the string concatenation of x and y. If $k \in \mathbb{N}$, a function $f(k)$ is negligible if $\exists\, k_0 \in \mathbb{N}$, $\forall\, k > k_0$, $f(k) < 1/k^c$, where $c > 0$ is a constant.

2.1 Public Key Encryption

A public key encryption scheme consists of three algorithms $\mathcal{PKE}$ = (PKE.Kg, PKE.Enc, Dec).

PKE.Kg: a randomized algorithm, taking a security parameter κ as input, generates a public key pk and a corresponding secret key sk, denoted as $(pk, sk) \leftarrow$ PKE.Kg(κ).

PKE.Enc: a possibly randomized algorithm, taking a public key pk, and a plaintext m taken from the message space as input, with internal coin flipping r, outputs a ciphertext c, denoted as $c \leftarrow \mathsf{PKE.Enc}(pk, m; r)$, in brief $c \leftarrow \mathsf{PKE.Enc}(pk, m)$.

PKE.Dec: a deterministic algorithm, taking a secret key sk and a ciphertext c as input, outputs the corresponding m, or "$\perp$" (indicating invalid ciphertext), denoted as $m \leftarrow \mathsf{PKE.Dec}(sk, c)$.

We require a PKE scheme should satisfy the standard correctness requirement, namely for all $(pk, sk) \leftarrow \mathsf{PKE.Kg}(1^\kappa)$ and all m, $\mathsf{PKE.Dec}(sk, \mathsf{PKE.Enc}(pk, m)) = m$. We give the definition of IND-CCA security below.

Definition 1 (PKE Security). *We say a public key encryption scheme $\mathcal{E}$ is (ϵ, T)-*IND-CCA *secure, if the advantage of any adversary $\mathcal{A}$ is at most ϵ within time T in the following experiment.*

$$\mathrm{Adv}^{\text{ind-cca}}_{\mathcal{E},\mathcal{A}}(\kappa) = |\Pr[(pk, sk) \leftarrow \mathsf{PKE.Kg}(1^\kappa); b \leftarrow \{0, 1\}; \\ b' \leftarrow \mathcal{A}^{\mathcal{EO}(pk,\cdot,\cdot,b),\mathcal{DO}(sk,\cdot)}(pk) : b' = b] - 1/2|$$

where $\mathcal{DO}$ is a decryption oracle, which returns the corresponding decryption result on a decryption query on a ciphertext c (except the challenge ciphertexts output by $\mathcal{EO}$). The adversary queries $\mathcal{EO}$ and $\mathcal{DO}$ at most q_e and q_d times, respectively. We say a PKE scheme is IND-CCA-MC *secure, if for polynomially bounded q_e, q_d and T, ϵ is negligible.*

Moreover, if we call the PKE scheme IND-CCA secure if $q_e = 1$.

2.2 KEMs with Partial Message Recovery (PKEMs)

Informally, a KEM with partial message recovery is a public key encryption that not only encrypts the session key but also encrypts part of the plaintext. Towards an easy comparison, we take the syntax of [7]. A KEM with partial message recovery (PKEM) consists three algorithms $\mathcal{PKEM} = (\mathsf{PKEM.Kg}, \mathsf{PKEM.Enc}, \mathsf{PKEM.Dec})$:

PKEM.Kg: a randomized algorithm, taking a security parameter κ as input, generates a public key pk and a corresponding secret key sk, denoted as $(pk, sk) \leftarrow \mathsf{PKEM.Kg}(\kappa)$. As part of the public key, there are two parameters *PKEM.msglen* and *PKEM.keylen*, denoting the length of message data that may be recovered from the KEM.

PKEM.Enc: a possibly randomized algorithm, taking a public key pk, and a partial plaintext m of length at most *msglen* taken from the message space as input, with internal coin flipping, outputs a session key dk of length *keylen* and a ciphertext c, denoted as $(dk, c) \leftarrow \mathsf{PKEM.Enc}(pk, m)$.

PKEM.Dec: a deterministic algorithm, taking a secret key sk and a ciphertext c as input, outputs the corresponding m, or "$\perp$" (indicating invalid ciphertext), denoted as $(m, dk) \leftarrow \mathsf{PKEM.Dec}(sk, c)$.

Additionally, we require the correctness property, namely, $\forall$ $(pk, sk) \leftarrow \mathsf{PKEM.Kg}(\kappa)$, $\forall$ m and $(c, dk) \leftarrow \mathsf{PKEM.Enc}(pk, m)$, there is $\mathsf{PKEM.Dec}(sk, c) = (m, dk)$. For simplicity, we shall omit the descriptions of *PKEM.msglen* and *PKEM.keylen* in clear contexts.

Security Definition. PKEMs provide better efficiency over plain KEMs, however, the more complicated structure makes the security definitions more intricate. As pointed out by [7], both session key confidentiality and data privacy of plaintext should be considered. Following [7], we call the resulting security notion IND-RoR-CCA, where data privacy, namely indistinguishability (IND), and session key confidentiality, namely real-or-random (RoR), are considered simultaneously against chosen ciphertext attack (CCA). Note that we allow the adversary to ask multiple challenge queries. Since the data confidentiality of the PKEMs is related to that of the PKEs, for easy understanding, we also use the notation IND-CCA-MC.

Definition 2 (PKEM Data Privacy). *We say a PKEM scheme is* (ϵ, q_e, q_d, t)-IND-CCA-MC *secure, if an adversary* $\mathcal{A}$ *with running time* t *has at most advantage* ϵ *in the following experiment, assuming it queries* $\mathcal{EO}$ *at most* q_e *times and* $\mathcal{DO}$ *at most* q_d *times, respectively.*

$$\mathrm{Adv}^{\mathsf{ror\text{-}cca}}_{\mathcal{E},\mathcal{A}}(\kappa) = |\Pr[(pk, sk) \leftarrow \mathsf{PKEM.Kg}(\kappa); b \leftarrow \{0,1\};$$
$$b' \leftarrow \mathcal{A}^{\mathcal{EO}(pk,\cdot,\cdot,b),\mathcal{DO}(sk,\cdot)}(pk) : b = b'] - 1/2|$$

$\mathcal{A}$ is given access to two oracles: An encryption oracle $\mathcal{EO}(pk,\cdot,\cdot,b)$, on an input pair of messages (m_0, m_1) of equal length, returns c_b and dk_b, where $(dk_b, c_b) \leftarrow$ PKEM.Enc(pk, m_b). A decryption oracle $\mathcal{DO}$, on an input ciphertext c, returns the corresponding session key dk and partial plaintext m or a special symbol "$\perp$" if c is invalid, where $(dk, m) \leftarrow$ PKEM.Dec(sk, c). The only limitation is that $\mathcal{A}$ cannot query to $\mathcal{DO}$ any ciphertext previously output by $\mathcal{EO}$. We say a PKEM is secure, if for any probabilistic polynomial time (PPT) bounded $\mathcal{A}$, ϵ is negligible.

Definition 3 (PKEM Session Key Privacy). *We say a PKEM scheme is* (ϵ, q_e, q_d, t)-RoR-CCA-MC *secure, if an adversary* $\mathcal{A}$ *with running time* t *has at most advantage* ϵ *in the following experiment, assuming it queries* $\mathcal{EO}$ *at most* q_e *times and* $\mathcal{DO}$ *at most* q_d *times, respectively.*

$$\mathrm{Adv}^{\mathsf{ror\text{-}cca}}_{\mathcal{E},\mathcal{A}}(\kappa) = |\Pr[(pk, sk) \leftarrow \mathsf{PKEM.Kg}(\kappa); b \leftarrow \{0,1\};$$
$$b' \leftarrow \mathcal{A}^{\mathcal{RR}(pk,\cdot,b),\mathcal{DO}(sk,\cdot)}(pk) : b = b'] - 1/2|$$

$\mathcal{A}$ is given access to two oracles: An Real-or-Random oracle $\mathcal{RR}(pk,\cdot,b)$, on an input message m, returns dk_b and c, where $(dk_1, c) \leftarrow$ PKEM.Enc(pk, m) and $dk_0 \leftarrow \mathcal{K}$ is a random session key chosen from the key space $\mathcal{K}$. A decryption oracle $\mathcal{DO}$, on an input ciphertext c, returns the corresponding session key dk and partial plaintext m or a special symbol "$\perp$" if c is invalid, where $(dk, m) \leftarrow$ PKEM.Dec(sk, c). The only limitation is that $\mathcal{A}$ cannot query to $\mathcal{DO}$ any ciphertext previously output by $\mathcal{EO}$. We say a PKEM is secure, if for any probabilistic polynomial time (PPT) $\mathcal{A}$, ϵ is negligible.

It is worth emphasizing that in the above two definitions, we allow $\mathcal{A}$ to have multiple challenge queries, i.e., access $\mathcal{EO}$ or $\mathcal{RR}$ multiple times. When $q_e = 1$, the multi-challenge model degenerates to the single challenge model.

3 A Modified Composition Theorem for PKEM

[7] has shown how to use PKEM with data encapsulation mechanism (DEM) to design hybrid encryption. The method is quite classical. Compared with normal KEM, PKEM has to additionally carry a part of the plaintext. We remark that one can use the same construction as mentioned in Def. 7 of [7], which is just a classic KEM/DEM construction of hybrid encryption with the KEM part carry a partial plaintext. We have the following modified composition theorem for the construction:

Theorem 1 (PKEM/DEM Security). *The PKEM/DEM construction of PKE is* $(\epsilon_1 + \epsilon_2 + \epsilon_3, t_1 + t_2 + t_3)$-IND-CCA-MC *secure, assuming the PKEM is* (ϵ_1, t_1)-IND-CCA-MC *and* (ϵ_2, t_2)-RoR-CCA-MC *secure and the DEM is* (ϵ_3, t_3)-IND-CCA-MC *secure.*

For limitation of space, we omit the definition of IND-CCA-MC security for DEMs and leave the complete proof the full version of this paper [24]. We discuss two issues. First, in [7], the DEM part was required to provide both passive security (a.k.a. semantic security for symmetric key encryption) and ciphertext authenticity (a.k.a. integrity of ciphertext). Actually, as shown in [14], this requirement can be relaxed. Recall the result of [5], a passively secure symmetric key encryption with ciphertext authenticity implies IND-CCA-MC secure symmetric key encryption [4,5,12], which may be weaker.

4 Generic Constructions from Secure PKE

In this section, we give two generic constructions: one from PKEs, and the other from PKEs with randomness recovery.

4.1 From PKEs

Construction 1. *Denote* $\mathcal{PKE} = \{\mathsf{PKE.Kg}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec}\}$ *as a public key encryption scheme with plaintext space* $\{0,1\}^{\ell}$*, then a secure PKEM can be constructed with plaintext space* $\{0,1\}^{\ell-k}$ *and the session key space* $\{0,1\}^{k}$ *as follows:*

PKEM.Kg: *taking as input a security parameter* κ*, outputs* $(pk, sk) \leftarrow \mathsf{PKE.Kg}(\kappa)$*, where pk is the public key and sk is the secret key.*

PKEM.Enc: *taking as input a public key pk and a message m, sampling* $dk \leftarrow \{0,1\}^{k}$*, outputs the ciphertext* $c \leftarrow \mathsf{PKE.Enc}(pk, m \| dk)$ *and the encapsulated session key dk.*

PKEM.Dec: *taking as input a secret key sk and a ciphertext c, computing* $(m \| dk) \leftarrow \mathsf{PKE.Dec}(sk, c)$*, outputs the partial plaintext m and the session dk.*

The correctness of this construction follows that of the underlying PKE, which is easily verifiable from the construction. We furthermore have Theorem 2 guarantee its security.

Theorem 2. *The scheme of Construction 1 is a secure PKEM scheme in the multi-challenge model. In particular, the PKEM scheme is* $(\epsilon, t+O(q_e\kappa))$-IND-CCA-MC *secure and* $(\epsilon, t+O(q_e\kappa))$-RoR-CCA-MC *secure, assuming the PKE scheme is* (ϵ, t)-IND-CCA-MC *secure.*

4.2 From PKEs with Randomness Recovery

With the warming-up in Section 4.1, it is natural to come up with the following idea: in Construction 1, one has used (independent) random coins for PKE. It will be more efficient, if the session key is also derived from the same random coins. Certainly, for the correctness condition, it is necessary that the decryption algorithm of the underlying PKE also recovers these random coins.

In the standard model, this approach may be problematic since the same random coins are used both to encrypt the partial message and generate the session key, and this is hard for a proof in general. Fortunately, our main applications are RSA-based primitives, and usually random oracles are inherent in the schemes. By assuming random oracles, we can actually achieve pretty good efficiency.

Informally speaking, a PKE with randomness recovery means that there is an efficient randomness recovery algorithm $\mathsf{PKE.Dec}'$ related to the decryption algorithm, with input a secret key sk and a valid ciphertext, recovers the original message m and the randomness r used in the encryption algorithm. We require that for all $(pk, sk) \leftarrow \mathsf{PKE.Kg}(\kappa)$, $(c, dk) \leftarrow \mathsf{PKE.Enc}(pk, m)$, we have $(m, r) \leftarrow \mathsf{PKE.Dec}'(sk, c)$. Naturally $|r| \geq \kappa$.

Construction 2. *Denote* $\mathcal{PKE} = (\mathsf{PKE.Kg}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *as a public key encryption scheme. Denote* $\mathsf{PKE.Dec}'$ *as the associated randomness recovery algorithm defined above. A secure PKEM can be constructed with plaintext space* $\{0,1\}^\ell$ *as follows:*

$\mathsf{PKEM.Kg}$: *taking as input a security parameter* κ, *outputs* $(pk, sk) \leftarrow \mathsf{PKE.Kg}(\kappa)$. *Additionally pick a cryptographic hash function* $T : \{0,1\}^{2\kappa} \to \{0,1\}^\kappa$. *The public key is* (pk, T) *and the secret key is* (sk, T).

$\mathsf{PKEM.Enc}$: *taking as input a public key pk and a message m, outputs the ciphertext* $c \leftarrow \mathsf{PKE.Enc}(pk, m; r)$ *and the encapsulated session key is* $dk \leftarrow T(pk, m, r)$.

$\mathsf{PKEM.Dec}$: *taking as input a secret key sk and a ciphertext c, computing* $(m, r) \leftarrow \mathsf{PKE.Dec}'(sk, c)$, *outputs the partial plaintext m and the session* $dk \leftarrow T(pk, m, r)$.

In the above construction, the randomness recovery algorithm $\mathsf{PKE.Dec}'$ is used instead of the decryption algorithm $\mathsf{PKE.Dec}$. The construction is similar to, but with essential improvement over [7]. In [7], the session key dk is set as $dk \leftarrow T(m, r)$, while in our scheme, $dk = T(pk, m, r)$. Our scheme has an obvious advantage: Including pk into the session key derivation eliminates the so-called rogue key malleability of the encapsulated session key. This attack is outside of the model of chosen ciphertext security, but may have significant impact in practice.

Theorem 3. *Construction 2 is a* $(\epsilon + q_T 2^{-k_2}, t + O(q_T \kappa))$-$\mathsf{IND\text{-}CCA\text{-}MC}$ *and* $(\epsilon, t + O(q_T \kappa))$-$\mathsf{RoR\text{-}CCA\text{-}MC}$ *secure, assuming* $\mathcal{PKE}$ *is* (ϵ, t)-$\mathsf{IND\text{-}CCA\text{-}MC}$ *secure with randomness recovery and T is a random oracle.*

The proofs of Theorems 2, 3 are left to the full version of this paper [24].

5 Concrete Schemes and Comparisons

In this section, we instantiate our generic constructions, and analyze their security and efficiency. The table only listed multi-challenge, but the result is similar wit single challenge model. It is then seen that compared with many other candidates, OAEP PKEM is quite inefficient.

Table 1. Comparisons of PKEMs in the multi-challenge model

Schemes	Reduction Cost	Ciphertext Size	Ciphertext Overhead
OAEP PKEM	$q_e\epsilon^2 + O(q_e q_G q_H 2^{-k_1} + q_e q_d q_G 2^{-k_2})$	$R_{2\kappa} + k_2$	$k_1 + k_2 \approx 7\kappa$
OAEP+ PKEM	$q_e\epsilon + O((q_e q_G 2^{-k_1} + q_e q_d 2^{-k_2})\kappa)$	R_κ	$k_1 + k_2 \approx 6\kappa$
OAEP++ PKEM	$\epsilon + O(q_d 2^{-k_1} + (q_e(q_G + q_d) + q_d q_G)2^{-k_2})$	$R_\kappa + k_2$	$k_1 + k_2 \approx 5\kappa$
OAEP-3R PKEM	$\epsilon + O(q_e(q_d^2 + q_G q_H + q_d q_f)2^{-k_2})$	R_κ	$k_2 \approx 4\kappa$

† For simplicity, we only list a few PKEM schemes using Construction 2 in the table. k_1 is the length of a constant vector or the output length of a hash function. k_2 is the bit length of the random coin. It remains possibility for the OAEP PKEM or the OAEP+ PKEM to have a better proof, since the above bound is obtained by directly using [3,21,9]. For simplicity, we assume $R_\kappa - k_2 = \ell \gg k_2$ for OAEP-3R.

6 Conclusion

In this paper, we investigate several issues regarding the KEM with partial message recovery (PKEM). We first point out some practical issues not considered by previous research, then go to to extend the security model in the multi-challenge setting. We then show two generic constructions in the extended model and compare some instantiations from well-known transforms for RSA.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: DHIES: An Encryption Scheme Based on the Diffie-Hellman Problem. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)
2. Abe, M., Gennaro, R., Kurosawa, K.: Tag-KEM/DEM: A New Framework for Hybrid Encryption. Cryptology ePrint Archive, Preliminary version appeared in Eurocrypt 2005 (2005), `http://eprint.iacr.org/2005/027/`
3. Bellare, M., Boldyreva, A., Micali, S.: Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (2000)
4. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES modes of operation. In: FOCS 1997. IEEE, Los Alamitos (1997)
5. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)

6. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption – How to encrypt with RSA. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
7. Bjørstad, T.E., Dent, A.W., Smart, N.P.: Efficient KEMs with Partial Message Recovery. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007. LNCS, vol. 4887, pp. 233–256. Springer, Heidelberg (2007)
8. Boldyreva, A.: Strengthening Security of RSA-OAEP. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 399–413. Springer, Heidelberg (2009)
9. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP Is Secure under the RSA Assumption. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 260–274. Springer, Heidelberg (2001)
10. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP Is Secure under the RSA Assumption. Journal of Cryptology 17(2), 81–104 (2004); Full version of [9]
11. Goldwasser, S., Micali, S.: Probabilistic Encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
12. Halevi, S., Rogaway, P.: A Tweakable Enciphering Mode. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 482–499. Springer, Heidelberg (2003)
13. Kobara, K., Imai, H.: Semantically Secure McEliece Public-Key Cryptosystems-Conversions for McEliece PKC. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 19–35. Springer, Heidelberg (2001)
14. 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)
15. Kurosawa, K., Schmidt-Samoa, K., Takagi, T.: A Complete and Explicit Security Reduction Algorithm for RSA-Based Cryptosystems. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 474–491. Springer, Heidelberg (2003)
16. Naor, M., Yung, M.: Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In: STOC 1990, pp. 427–437. ACM, New York (1990)
17. National Institute of Standards and Technology. Recommendation for Key Management - Part 1: General (Revised). NIST Special Publication 800-57 (2007), http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf
18. Pointcheval, D.: How to Encrypt Properly with RSA. RSA Laboratories' CryptoBytes 5(1), 9–19 (Winter/Spring 2002)
19. Rackoff, C., Simon, D.R.: Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
20. Shoup, V.: ISO 18033-2: An Emerging Standard for Public-Key Encryption (committee draft) (June 2001), http://shoup.net/iso/
21. Shoup, V.: OAEP Reconsidered. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 239–259. Springer, Heidelberg (2001)
22. Shoup, V.: ISO/IEC FCD 18033-2 - Information technology - Security techniques - Encryption algorithms - Part 2: Asymmetic ciphers. Technical report, International Organization for Standardization (2004), http://www.shoup.net/iso/std6.pdf
23. Wander, A.S., Gura, N., Eberle, H., Gupta, V., Shanz, S.C.: Energy Analysis of Public-Key Cryptgraphy for wireless Sensor Network. In: 3rd IEEE Internatinal Conference on Pervasive Computing and Communications (PerCom), pp. 324–328. IEEE, Los Alamitos (2005)
24. Zhang, R., Imai, H.: Constructing Better KEMs with Partial Message Recovery (full version). In: Extended abstract appeared in Inscrypt 2009 (2009), http://staff.aist.go.jp/r-zhang/papers/PKEMs.pdf

A Novel Contagion-Like Patch Dissemination Mechanism against Peer-to-Peer File-Sharing Worms

Xiaofeng Nie, Jiwu Jing, and Yuewu Wang

State Key Laboratory of Information Security
Graduate University of Chinese Academy of Sciences, Beijing 100049, China
{xfnie,jing,ywwang}@is.ac.cn

Abstract. Peer-to-peer (P2P) file-sharing worms are becoming a deadly security threat to P2P systems. The defense that just relies on the improvement of users' security awareness and their individual recoveries is not adequate. Existing automated patching systems such as Microsoft Windows Update and Symantec Security Update are also not necessarily the best fits in combat with P2P file-sharing worms, because of the inconsistency between the jurisdiction of these patching systems and the propagation community of P2P file-sharing worms. In this paper, with a deep understanding of the propagation characteristic of P2P file-sharing worms and the inspiration of more rapid contagion worms, we propose a complementary contagion-like patch dissemination mechanism which utilizes the existing file-sharing infrastructure to timely disseminate security patches between the participating peers of the file downloading. In addition, the digital signature scheme is introduced to prevent malicious peers tampering with patches in the dissemination process. Through the epidemiological model and extensive packet-level simulations we demonstrate the effectiveness of the proposed patch dissemination mechanism.

Keywords: Peer-to-peer file-sharing worm, Contagion, Workload.

1 Introduction

In recent years, more and more network applications adopt the peer-to-peer (P2P) architecture to achieve high scalability and reliability, such as file-sharing, instant messaging, cooperative computing, and live media streaming. Among the numerous existing P2P applications, file-sharing is the most widely used application. It has been illustrated that P2P file-sharing traffic accounted for more than 44% of North-American Internet traffic in the traffic statistics report on North-American broadband networks released by Sandvine in June 2008 [1].

The growing popularity of P2P networks provides an ideal venue for worm propagation. Recently, many worms exploiting the file-sharing service of P2P systems as their major infection vector have been reported, e.g., Benjamin, Surnova and Duload [2], which are referred to as P2P file-sharing worms. These worms usually disguise themselves as popular files in P2P file-sharing systems to attract users' attention. When downloaded and opened by the user without being aware of the threat, they infect the victim host and add a number of copies of themselves with different attracting names to the shared folder of the new infected host to accelerate their propagation. Since P2P file-sharing worms passively wait the download of the victims instead of active scanning and also require human's operation to propagate, they are also referred to as P2P passive worms or P2P virus. Two measurements about malware prevalence in the KaZaA file-sharing network in February and May 2006 suggested that: over 12% of KaZaA hosts were infected by over 40 different malwares, and the malwares accounted for more than 22% and 15% of the total crawled data respectively [3].

Due to the increasing trend in worms of exploiting file-sharing application as their major infection vector, it is necessary to carry out in-depth investigations on the design space of defense strategies against P2P file-sharing worms. Being the currently most popular vehicle for eliminating and containing Internet worms, automated patching systems are widely employed by vendors (e.g., Microsoft and Symantec) to automatically push the latest security patches to Internet hosts. Since the existing patching systems are not exclusively designed for P2P systems, there exist some problems when combating with P2P file-sharing worms. First, the jurisdiction of these patching systems and the propagation community of P2P file-sharing worms are are often not a good match. In most cases, hosts running P2P client programs may not install automated patching tools, and vice versa. On one hand, the P2P users out of the jurisdiction of patching systems cannot get the latest P2P patches; on the other hand, whether the security server of patching tools blindly push P2P patches to all Internet hosts or scan for existence of P2P client programs within each Internet host before sending a patch to it, not only system resources and network bandwidth could be greatly wasted, but unnecessary latency would be introduced. Moreover, most existing patching systems adopt the traditional client/server architecture, which is essentially different from the distributed P2P architecture with high scalability and reliability. The introduce of the centralized patching server may cause the whole system to have a single point of failure and suffer from lack of scalability.

Utilizing the existing file-sharing infrastructure, Xie et al. provided two internal patch dissemination mechanisms exclusively for P2P file-sharing systems, a download-based approach and a search-based approach [4]. In the download-based approach, the security patches are first disseminated to a small set of popular key peers which provide a large fraction of shared files, and then the other participating peers could receive the security patches from these popular peers when they actively download files from these popular peers. In the search-based approach, once the key peer detects worm infection in a file just downloaded, it immediately re-performs a search to locate a set of suspicious targets, and actively push security patches to some of them.

Contributions: In this paper, with a deep understanding of the propagation characteristic of P2P file-sharing worms and the inspiration of more rapid contagion worms, we propose a novel contagion-like patch dissemination mechanism against P2P file-sharing worms. Being a proprietary patch dissemination mechanism specially designed for P2P systems, the proposed patch dissemination mechanism utilizes the existing file-sharing infrastructure to internally disseminate patches between the participating peers of the file downloading. Security patches cannot only be received from the file provider as a piggyback of the requested file, but also can be received from the file requestor which proactively pushes the patch to the file provider as an acknowledgement of the requested file. Just as contagion worms, security patches continue their dissemination cycle between the participating peers with the workload of P2P file-sharing systems, while P2P file-sharing worms can only propagate from the file provider to the file requestor. In addition, the digital signature scheme is introduced to prevent malicious peers from tampering with patches in the process of patch dissemination. Our proposed scheme is not a substitution of the existing automated patching systems but rather a nice complement to them.

In the download-based approach, security patches can only be received from the fixed initial patch dissemination peers; in the search-based approach, the precise identification of suspicious targets is difficult. In our proposed contagion-like patch dissemination mechanism, all the participating peers can be the candidates for security patch providers and receivers. After being patched, these participating peers cannot only provide the received security patch to the incoming file requestors, but also proactively push the received security patch back to the file providers when they download files from the file providers. Thus, security patches can be disseminated more effectively in a simple way. Through the deterministic epidemiological model and extensive packet-level simulations we demonstrate the effectiveness of the proposed patch dissemination mechanism.

Organization: The rest of the paper is organized as follows. Section 2 gives an overview of the propagation characteristics of P2P file-sharing worms. Section 3 provides a deep dive into the details of the proposed contagion-like patch dissemination mechanism against P2P file-sharing worms, and discusses the effectiveness of the patching scheme with a deterministic epidemiological model. With the constructed P2P file-sharing worm simulator, we further evaluate the proposed patch dissemination mechanism through the packet-level simulations and summarize the experimental results in Section 4. Finally, we conclude this paper in Section 5 with a brief summary and an outline of future work.

2 Preliminaries

P2P file-sharing worms usually disguise themselves as popular files in P2P systems to attract users' attention. When downloaded and opened by the user without being aware of the threat, P2P file-sharing worms infect the victim host. To avoid suspicion, these worm usually not directly add a number of copies of themselves with different attracting names to the existing shared folder of the new infected host to accelerate their propagation. Instead they usually first create a hidden folder in the system directory of the new infected host, and then set the hidden folder as an extra shared folder of the P2P application to which a number of malicious copies with different attracting names are appended [2]. In this way, P2P file-sharing worms continue their spread cycle. It is important to note that, the propagation of P2P file-sharing worms is closely related to the workload of P2P file-sharing systems, and P2P file-sharing worms can only spread from the file provider to the file requestor with the file downloading. Although the spread of P2P file-sharing worms is slower than that of random scanning worms, due to their better performance in stealth, it is recommended as a major infection vector especially in the preparing stage for a full fledged attack.

Files in P2P file-sharing systems are divided into normal files and malicious files that are virtually the P2P file-sharing worms in this paper. According to the existence of malicious files and their damage, all peers in the P2P file-sharing system can be classified into one of following categories:

Susceptible - Peers that are not sharing any malicious files, but at risk of downloading malicious files and being infected.

Exposed - Peers that have downloaded one or more malicious files, but have not executed them. The exposed category is included on account of the delay between the download of malicious files and execution.

Infected - Peers that have executed the malicious file. Upon execution, the host become compromised and some more copies of malicious files reside in the peer's shared folder.

Recovered - Peers that have installed security patches. Once the patch has been applied, the peer's shared folder will be immediately scanned and cleaned, and the peer become immunized to the worm.

The number of peers in each category at time t is denoted as $S(t)$, $E(t)$, $I(t)$, and $R(t)$, respectively. N is the total number of peers in the P2P system. At all times, every one of the N peers making up the system falls into one of the four categories. Thus, for all value of t, $N = S(t) + E(t) + I(t) + R(t)$.

3 The Contagion-Like Patch Dissemination Mechanism

To effectively eliminate and contain P2P file-sharing worms, we cannot merely rely on the improvement of users' security awareness and their individual recoveries; rather, we should disseminate the security patches to all participating peers of P2P systems in an automated and systematic approach [5]. However, as described in previous sections, existing automated patching systems are not necessarily the best fits in combat with P2P file-sharing worms. Besides the inconsistency between the jurisdiction of these patching systems and the propagation community of P2P file-sharing worms, the essential difference in the architecture is another important reason. Because the patch is one special kind of files, the existing file-sharing infrastructure should be considered rather than rebuilding. Moreover, Since the spread of P2P file-sharing worms is much slower than that of random scanning worms, a relative faster patch dissemination mechanism would throttle the spread of P2P file-sharing worms well, not as the traditional patch dissemination mechanism against random scanning worms that should be as soon as possible. Based on the above considerations, we propose a contagion-like patch dissemination mechanism against file-sharing worms.

3.1 Scheme Description

Staniford et al. first presented the theoretical threat of *contagion worms* that spread surreptitiously within the usual communication patterns to escape detection [6]. The core idea of the propagation measure of contagion worms can be expressed as follows. Suppose an attacker has attained a exploit W which subvert the clients of a P2P system. The attacker releases the worm on a peer, and then simply waits. When other peers request download from the subverted peer, the exploit W is sent with the download response; when the subverted peer requests download from other peers, the exploit W is also sent with the download request. The file requestor or file provider receiving the exploit are subverted by the exploit W if they are vulnerable to it. In this fashion, the infection spreads in the P2P system, much as a contagious disease spread based on the incidental traffic patterns of its hosts, so this surreptitious worm is notably referred to as the contagion worm.

If the exploit of contagion worms was substituted by the security patch, a new fast patch dissemination mechanism would be available. The rate of patch dissemination under the new mechanism will be faster than the spread of P2P file-sharing worms, and little influence will be caused to the traffic patterns of P2P systems. The patches in our proposed mechanism are not like traditional worms which exploit the system vulnerability to self-propagate, but with slight modification in the file download protocol to make the security patches disseminated in P2P systems with the similar behavior of contagion worms.

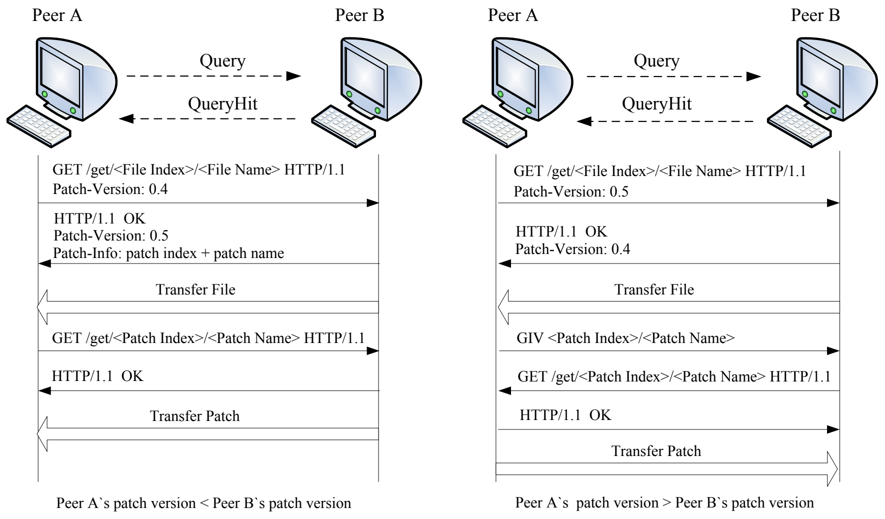

Peer A
Peer B
Query
QueryHit
GET /get/<File Index>/<File Name> HTTP/1.1
Patch-Version: 0.4
HTTP/1.1 OK
Patch-Version: 0.5
Patch-Info: patch index + patch name
Transfer File
GET /get/<Patch Index>/<Patch Name> HTTP/1.1
HTTP/1.1 OK
Transfer Patch
Peer A`s patch version < Peer B`s patch version
Peer A
Peer B
Query
QueryHit
GET /get/<File Index>/<File Name> HTTP/1.1
Patch-Version: 0.5
HTTP/1.1 OK
Patch-Version: 0.4
Transfer File
GIV <Patch Index>/<Patch Name>
GET /get/<Patch Index>/<Patch Name> HTTP/1.1
HTTP/1.1 OK
Transfer Patch
Peer A`s patch version > Peer B`s patch version

Fig. 1. Illustration of contagion-like patch dissemination with the workload

The interaction between peers in the process of contagion-like patch dissemination is illustrated in Fig. 1. Suppose peer A determines to download a file from peer B based on the search results. Peer A first initiates a HTTP GET request to peer B with the current version number of the patch repository in peer A. After receiving the download request, peer B sends a HTTP response with the current version number of the patch repository in peer B. If the version number of the patch repository in peer A is smaller than that of peer B, the information about the latest patch (e.g. the index, name and size of the patch) in peer B also should be provided with the HTTP response. After the transfer of the requested file as the normal Gnutella paradigm, if the version number of the patch repository in peer A is smaller than that of peer B and the information about the latest patch in peer B is provided, peer A sends a HTTP GET request to download the latest patch from peer B. If the version number of the patch repository in peer A is larger than that of peer B, peer A sends a GIV message with the information about the latest patch in it as a signal to proactively push the patch back. After receiving the GIV, peer B initials a reverse HTTP GET request to download the latest patch from peer A.

3.2 Security Enhancement

In the above described scheme, all the participating peers can be the candidates of patch providers, while in the traditional centralized patching systems the security patches can only be got from the central server. For the openness nature of P2P systems, malicious peers could easily replace the patch with other files to interfere with the normal patch dissemination process or even inject malicious codes to compromise the whole system, so it is essential to consider the authentication and integrity of security patches. The digital signature scheme based on public key cryptography is introduced to our proposed scheme to ensure that security patches originate from the legitimate security vendors and are not tampered with in the dissemination process.

The security patches in contagion-like patch dissemination mechanism consist of three parts: peer information, patch information and signature. The peer information part contains the peer identifier and the version number of the patch repository of the patch provider, and the two counterparts of the patch receipt. The patch information part contains the description of the security patch (e.g., the release time and the publisher of the security patch, and the name, severity level and propagation vector of the worm) and the security patch itself, which is encoded in binary delta compression format to reduce the patch size.

Once a new security patch is generated, the security vendor immediately uses its private key to sign the concatenation of the patch description and the patch content, then publishes patches to initial patch dissemination peers. In the process of patch dissemination, the fields of patch description, patch content and digital signature should be kept unchangeable. Our proposed mechanism does not require every receiver to authenticate the intermediate patch distributor but the original publisher of security patches.

When receiving a security patch, the peer validates the signature with the built-in certificate of the security vendor. If the signature is valid, the peer installs the security patch; otherwise, the security patch is dropped.

3.3 Modeling and Performance Analysis

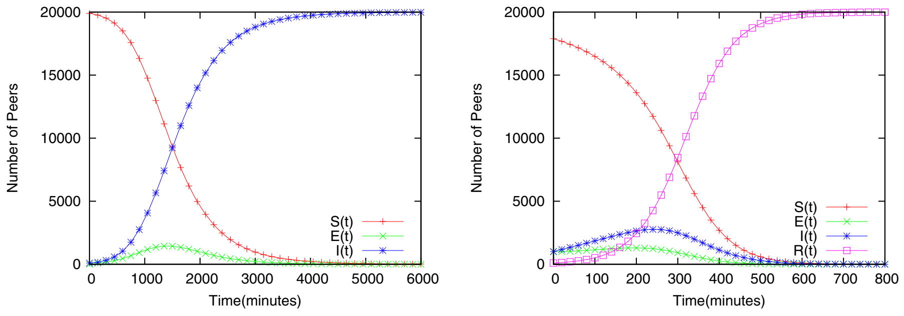
In this section we derive a deterministic epidemiological model and analyze the performance of the contagion-like patch dissemination mechanism theoretically.

Besides the fundamental notations, $S(t), E(t), I(t), R(t)$ and N, we adopt the notation λ to denote the average rate of file download per peer, η to denote the average rate of file execution per peer, and c to denote the number of malicious files generated when a file-sharing worm is executed. The total number of malicious files and normal files at time t are denoted by $K(t)$ and $J(t)$, respectively. When a user downloads a file, suppose the probability of choosing a malicious file, $h(t)$, depends on the prevalence of malicious files in the network, that is, $h(t) = \alpha\frac{K(t)}{K(t)+J(t)}$, where the constant α is a regulatory factor [7].

Based on the peer state transition described in Section 2, we derive the following group of differential equations that govern the system evolution.

$$\begin{cases} dS(t)/dt = -\lambda S(t)h(t) - 2\lambda S(t)R(t)/N \\ dE(t)/dt = \lambda S(t)h(t) - \eta E(t) - 2\lambda E(t)R(t)/N \\ dI(t)/dt = \eta E(t) - 2\lambda I(t)R(t)/N \\ dR(t)/dt = 2\lambda[S(t) + E(t) + I(t)]R(t)/N \\ dK(t)/dt = \lambda S(t)h(t) + \eta E(t)(c-1) - 2\lambda E(t)R(t)/N - 2c\lambda I(t)R(t)/N \\ dJ(t)/dt = \lambda N[1 - h(t)] \end{cases}$$

The simulation parameters are set as follows: $N = 20000$, $J(0) = 200,000$, $c = 20$, $\lambda = \eta = 0.5/hour$, and $\alpha = 0.6$. The initial proportion of all categories of peers are set as follows: in the case of no defense deployment, $S(0) = 99.5\%$, $E(0) = 0$, and $I(0) = 0.5\%$; while in the case with contagion-like patch dissemination mechanism, $S(0) = 89.5\%$, $I(0) = E(0) = 5\%$, and $R(0) = 0.5\%$.



(a) Without patching (b) With contagion-like patching

Fig. 2. System evolution status on the epidemiological model

Figure 2 illustrates the system evolution status under the above two scenarios. The results indicate that the contagion-like dissemination of security patches is significantly faster than the propagation of file-sharing worms, and the contagion-like patch dissemination mechanism can effectively contain file-sharing worms.

The maximum proportion of infected peers is about 13%, slightly larger than the sum of exposed peers and infected peers before disseminating security patches.

4 Experimental Evaluation

4.1 Simulation Setup

Based on the Gnutella 0.6 protocol and the latest measurement results, we implement a packet-level P2P file-sharing worm simulator. In this simulator, several important factors affecting the spread of P2P file-sharing worms are taken into consideration. The main modules of the simulator are constructed as follows.

Initial replicas distribution: The measurement study on the characteristic of available files in the modern Gnutella networks demonstrated that the number of shared files and contributed storage space by individual peers both follow a power-law distribution, and the replicas popularity of individual files follow a Zipf distribution [8]. Zipf's law is a particular power-law function with coefficient close to 1. Based on the Generalized Linear Preference (GLP) model for one-dimensional power-law distribution, we construct the initial replicas distribution of P2P systems as a two-dimensional power-law distribution. The coefficients of the power-law distribution and the Zipf distribution are respectively set to be 2.5 and 0.8 according to the measurement results. With the continuous downloads, the replicas distribution of P2P systems keeps changing.

Overlay topology: Previous measurements reported that both the top-level overlay topology and the entire overlay topology of the Gnutella 0.6 network with two-tier hierarchy exhibit small world properties [9]. According to this discovery, we construct the overlay topology for simulation. We first construct the top-level overlay topology composed of ultrapeers based on the Watts and Strogatz's one-dimensional small world model. Then for each leafpeer, we choose U ultrapeers to connect to uniformly at random with an assumption that the number of ultrapeers that the leafpeer connected to follows a normal distribution $U \sim N(2, 1)$. If the chosen ultrapeer already has η leafpeers, re-choose another ultrapeer to connect to, where η denotes the maximum number of leafpeers that a ultrapeer can serve. The ratio of ultrapeers to leafpeers in the overlay topology is set to 1:4 according to the measurement results.

Workload engine: The immutability of shared files causes the workload of P2P file-sharing systems two distinguished characteristics. First, the file-sharing clients rarely request the same file twice ("fetch-at-most-once"), unlike web clients which fetch the same Web page many times. Second, the introduction of new files and the addition of new users are the driven force which prevents the fetch-at-most-once behavior driving the system to stagnation [10]. In our model, we hypothesize that the underlying file request popularity follows a Zipf distribution, and the popularity ranks are set according to the initial replicas distribution. Due to the fetch-at-most-once behavior, subsequent requests from the same client obey the distribution obtained by removing already fetched files from the candidate file set and re-scaling so the total probability is 1.0. When a new file is introduced to the file-sharing system, its popularity rank is determined by selecting randomly from Zipf(1) distribution.

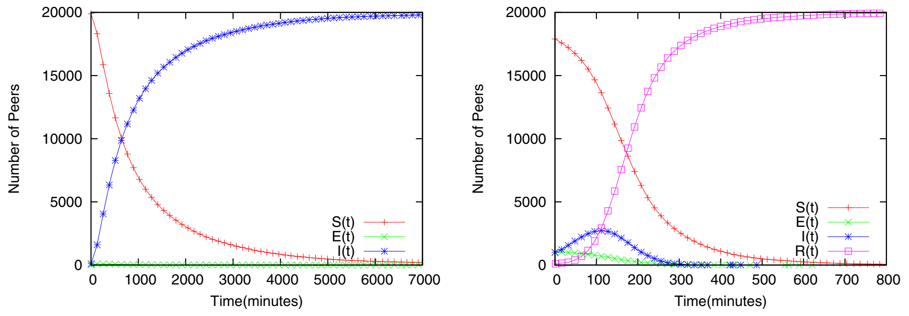
The simulation parameters are set as follows: the total number of peers $N = 20000$, the initial number of unique files $M = 30000$, the initial number of replicas $J = 200,000$. The 20 popular files disguised by file-sharing worms are selected uniformly at random from the top 100 most popular files. Suppose the number of generated queries of each peer in an hour follows an exponential distribution, $\lambda_Q \sim Exponential(0.5)$. Suppose all downloaded files will be executed, and the interval between the file download and the execution follows another exponential distribution, $t \sim Exponential(5min)$. For simplicity we assume that the file is downloaded from a single peer, and the initial patch dissemination peers are selected uniformly at random. The initial proportions of all categories of peers are set as those in the epidemiological model. All the following results are the average of 100 experiments.

4.2 Performance Analysis

The system evolution status on the simulator is illustrated in Fig. 3. In the beginning the propagation of P2P file-sharing worms on the simulator is significantly faster than that on the analytical model, but after about 75% of the participating peers are infected, the propagation on the simulator greatly declines. The total time needed for P2P file-sharing worms to infect the whole system on the simulator is longer than that on the analytical model. This phenomenon can be explained as follows. In the beginning, as most peers request the popular files which file-sharing worms disguise, the probability of choosing a malicious file is considerably higher than the proportion of malicious files. With the ongoing infections, the remaining uninfected peers are mostly the peers which share some or all the files which file-sharing worms disguise. Because of the fetch-at-most-once behavior, these peers will rarely request the shared files any more, so they have a lower probability of choosing a malicious file. While on the analytical model the probability of choosing a malicious file is just related to the proportion of malicious files, which keeps increasing with the ongoing infections.

The dissemination of security patches on the simulator is slightly faster than that on the analytical model. The reason is that when the precise replicas distribution is taken into consideration, peers are more likely to download files from the peers sharing more files, not as on the analytical model where all peers have the equal probability to be downloaded from. This preference accelerates the dissemination of security patches. In spite of the difference between the dissemination curve on the analytical model and that on the simulator, the maximum proportions of infected peers are approximately equivalent.

With the same parameters, we investigate the performance of the download-based patch dissemination mechanism, where security patches can only be received from a fixed set of popular peers which share the most files in the P2P system. Figure 4 shows the system evolution status with the download-based patch mechanism. Compared with the download-based patch dissemination mechanism, our proposed contagion-like patch dissemination mechanism can disseminate



(a) Without patching (b) With contagion-like patching

Fig. 3. System evolution status on the file-sharing worm simulator

security patches to all participating peers in significantly less time, and throttle the maximum number of infected peers to a rather lower level. Besides we investigate the impact of the number of initial dissemination peers and the reaction time on the maximum number of infected peers. We adopt *patching threshold* to control when the patching procedure starts, which represents the time delay since the worm starts propagating till it is detected and a patch is generated. In this model, the patching threshold is measured as the percentage of a sum of exposed peers and infected peers in the system. Figure 5 illustrates that both a higher number of initial dissemination peers and a lower patching threshold result in a less severe infection.

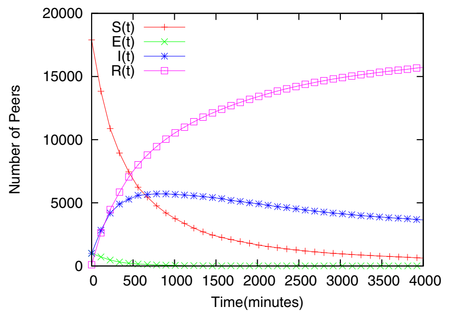


Fig. 4. System evolution status under the download-based patching approach

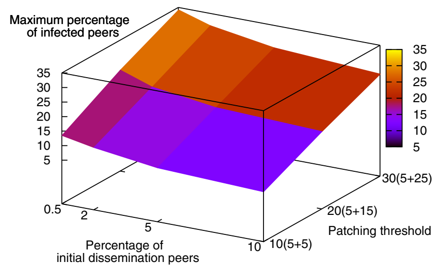


Fig. 5. Maximum percentage of infected peers vs. percentage of initial dissemination peers and patching threshold

5 Conclusions and Future Work

In this paper, we propose a contagion-like patch dissemination mechanism as a complement to existing centralized patching mode to combat with P2P file-sharing worms. The contagion-like patch dissemination mechanism utilizes the existing file-sharing infrastructure to internally disseminate security patches between the participating peers of the file downloading. In addition, the digital signature scheme is introduced to prevent malicious peers tampering with patches in the process of patch dissemination. Through the epidemiological model and extensive simulations we demonstrate the effectiveness of the proposed patch dissemination mechanism. In future work we plan to investigate effective patch dissemination mechanisms for P2P topological worms.

Acknowledgments

This research was supported in part by the National Natural Science Foundation of China under Grant Nos.60573015, 70890084/G021102, and the China Postdoctoral Science Foundation under Grant No.20080440072.

References

1. Sandvine Incorporated ULC. 2008 Analysis of Traffic Demographics in North-American Broadband Networks (June 2008), `http://www.sandvine.com/general/documents/Traffic_Demographics_NA_Broadband_Networks.pdf`
2. `http://www.viruslist.com/en/virusesdescribed?chapter=153311928`
3. Shin, S., Jung, J., Balakrishnan, H.: Malware Prevalence in the KaZaA File-Sharing Network. In: 6th ACM SIGCOMM Internet Measurement Conference, pp. 333–338. ACM Press, New York (2006)
4. Xie, L., Song, H., Zhu, S.: On the Effectiveness of Internal Patching Against File-Sharing Worms. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 1–20. Springer, Heidelberg (2008)
5. Brumley, D., Liu, L., Poosankam, P., Song, D.: Design Space and Analysis of Worm Defense Strategies. In: ACM Symposium on Information, Computer and Communication Security, pp. 125–137. ACM Press, New York (2006)
6. Staniford, S., Paxson, V., Weaver, N.: How to Own the Internet in Your Spare Time. In: 11th USENIX Security Symposium, pp. 149–167. USENIX Association, Berkeley (2002)
7. Thommes, R., Coates, M.: Epidemiological Modeling of Peer-to-Peer Viruses and Pollution. In: IEEE INFOCOM 2006, pp. 1–12. IEEE Press, Piscataway (2006)
8. Stutzbach, D., Zhao, S., Rejaie, R.: Characterizing Files in the Modern Gnutella Network: A Measurement Study. Multimedia Systems 13, 35–50 (2007)
9. Stutzbach, D., Rejaie, R., Sen, S.: Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems. IEEE/ACM Transactions on Networking 16, 267–280 (2008)
10. Gummadi, K.P., Dunn, R.J., Saroiu, S., Gribble, S.D., Levy, H.M., Zahorjan, J.: Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload. In: 19th ACM Symposium on Operating System Principles, pp. 314–329. ACM Press, New York (2003)

Remodeling Vulnerability Information

Feng Cheng, Sebastian Roschke, Robert Schuppenies, and Christoph Meinel

Hasso Plattner Institute (HPI), University of Potsdam,
P.O.Box 900460, 14440, Potsdam, Germany
{feng.cheng, sebastian.roschke, robert.schuppenies,
christoph.meinel}@hpi.uni-potsdam.de

Abstract. This paper addresses the challenges to formally specify the vulnerability information and unify text-based vulnerability descriptions, which might be available in various commercial, governmental, or open source vulnerability databases, into a generic information model. Our motivation is to utilize the remodeled vulnerability data for automating the construction of attack graph, which has been recognized as an effective method for visualizing, analyzing, and measuring the security of complicated computer systems or networks. A formal data structure is proposed based on a comprehensive conceptual analysis on normal computer infrastructure and related vulnerabilities. The newly proposed vulnerability representation, which contains most of meaningful properties extracted from textual descriptions of actual vulnerability items, can be directly fed into the reasoning engine of attack graph tools. A lightweight information extraction mechanism is designed to automatically transform textual vulnerability descriptions into the proposed data structure. Several *Reader* and *Writer* plugins are implemented to enable the communication with known vulnerability repositories.

1 Introduction

In recent years, a large number of vulnerabilities, which may result from weak passwords, software bugs, computer virus, malware, script code, SQL injection, misconfiguration, etc., have been revealed and archived by various communities, either commercial, governmental, or open source [1]. The provided vulnerability capabilities (repositories, tools, and services) are widely used by security auditing, risk measurement, system adminstration, network management, penetration testing, IDS/IPS, as well as software design, implementation, and evaluation, etc.

However, due to huge varieties on goals and requirements, the vulnerability information is represented and organized in different ways. The lack of similarities and common criteria for vulnerability description has been recognized as a major issue for sharing data across separate vulnerability databases. Several standards or techniques, e.g., CVE [2], OVAL [3], CVSS [4], IDMEF [5], etc., have been proposed to address this challenge ([6], [7]). However, the listed techniques are not fully supported by most vulnerability repositories used in practice. Moreover, most of standards are based on the textual descriptions, which makes it difficult to directly use the vulnerability information for automatic reasoning or further processing efforts based on other formal methods, e.g., attack graph ([7], [8], and [9]), which is an effective method to model, analyze, and evaluate the security of complicated computer systems or networks.

In this paper, we propose a generic data structure which can model the vulnerability information in a formal way. Our work is originally motivated by finding a way for automatic construction of attack graphs. To construct an attack graph, the runtime information about the target system or network environment should be monitored, gathered, and later evaluated with existing descriptions of known vulnerabilities. The heterogeneous vulnerability repositories maintained by different vulnerability vendors result in many practical problems on compatibility and inconsistency for the attack graph tools. Unifying vulnerability description is expected but unfortunately has not been researched. On the other hand, some vulnerability information, which is originally reported and even later archived in textual descriptions, have been neglected by some emerged attack graph constructors. The reason is that these descriptions do not follow a machine understandable format and therefore can not be directly processed by computer programs. Thus, current attack graph tools mostly rely on hand-generated input or are restricted to only a few attributes, which are manually extracted from those public vulnerability databases. The problems we expect to solve in this paper include: a) how to design a data structure for fully containing the properties of a textual vulnerability information. b) how to automatically extract meaningful information from various existing vulnerability databases and transform them into the proposed unified format.

The reminder of this paper is organized as follows. Some background information and analysis, concerning on basic concepts of computer vulnerability and existing works on modeling vulnerability information, are provided in Section 2. Besides, some recognized efforts towards unifying vulnerability information are presented. In Section 3, a new data structure is proposed which can be used to represent vulnerability information as well as some important properties of systems under attacks. Section 4 shortly presents how to extract useful information from existing vulnerability databases and then transform them into the proposed data structure. Then, we conclude the paper in Section 5.

2 Towards Modeling Vulnerability Information

2.1 Vulnerability Representation: Description and Condition

Computer system vulnerabilities describe the potential harm threatening an IT system. The cause of this harm can be assigned to one of the four categories: software layer, physical layer, administrative layer, and user layer. The software layer includes vulnerabilities which originate in the unintended behavior of applications. The physical layer includes vulnerabilities that result from physical access to hardware on which applications run. The administrative layer includes vulnerabilities caused by procedures, policies and configurations, etc. Vulnerabilities are often the result of poor policies. The last, and probably most damaging class of vulnerabilities are caused by users. No matter how good software is programmed, the hardware is protected, and how tight the security policies may be defined, if a user writes down the account password on a Post-it note and sticks it right on the monitor, the system remains highly vulnerable.

A vulnerability may have a number of *exploits* which describe how to cause unintended behavior based on the corresponding weakness. For each *exploit*, *mitigation* may be known to circumvent the exploitation of a *vulnerability*. Transitively, each *mitigation* relates to a specific *vulnerability*. All three, *vulnerability*, *exploit*, and *mitigation* belong to *description*, since they describe a situation, how to make use of it, as well as how to prevent it. Note the difference in *vulnerability* which describes a situation as is, and *an exploit at hand* or *mitigation* which describes a how-to that is a procedure or algorithm. *Descriptions* are not necessarily textual descriptions only, but can be available in the form of code examples, recommendations, or the like. A piece of software can be a *description* as well. Also, having an *exploit* at hand does not mean making use of it. For this, actions are used which are realizations of *descriptions*. The action making use of an *exploit* is called *attack*, the action applying a *mitigation* is called a *fix*.

Both actions, i.e., *attack* and *fix*, result in the change of *system properties* referred to by a *description* as *condition*. A *condition* is a characterization of *system properties*. Although there is no difference in the structure of a *precondition* and a *postcondition*, both are proposed to clarify the concept of *condition*.

Take the ***Downadup***[1] vulnerability as an example. The textual description of the CVE entry 2008-4250 states:

> The Server service in Microsoft Windows 2000 SP4, XP SP2 and SP3, Server 2003 SP1 and SP2, Vista Gold and SP1, Server 2008, and 7 Pre-Beta allows remote attackers to execute arbitrary code via a crafted RPC request that triggers the overflow during path canonicalization, as exploited in the wild by Gimmiv.A in October 2008, aka "Server Service Vulnerability."

Vulnerable applications were all Microsoft Windows versions from Windows 2000 to Windows Server 2008. The inflicted harm is described as "allows remote attackers to execute arbitrary code", which results in an integrity violation, but can easily turn into a confidentiality as well as an availability violation. Because the affected programs are operating systems, it is assumed that any data stored and program executed in the context of this operating system will be affected. The technique used is described as "RPC request that triggers the overflow during path canonicalization" and an exploit is available by the name "Gimmiv.A". Microsoft has released a patch as a fix on October 23rd, 2008[2].

2.2 Unification on Known Vulnerability Databases

Similar to the ***Downadup*** example given above, many other vulnerabilities emerge each year. Reports are provided by commercial organizations which

[1] `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4250`

[2] `http://www.microsoft.com/technet/security/bulletin/ms08-067.mspx`

offer security solutions, as well as by individual security researchers. Known vulnerabilities of programs are collected in vulnerability databases (VDB). Such databases comprehend large compilations of software weaknesses in a non-uniform manner. The reason is likely the different goals that different providers of VDBs pursuit. Currently, a large number of VDBs are maintained by commercial vendors, such as *SecurityFocus* (S.Focus)[3], *DragonSoft* (D.Soft)[4], Secunia [5], X-Force[6], etc. In comparison, there are also some non-commercial VDBs, such as *Open Source Vulnerability Database* (OSVDB)[7], *Cooperative Vulnerability Database* (CoopVDB)[8], *Department of Energy Cyber Incident Response Capability* (DoE-CIRC)[9], *National Vulnerability Database* (NVD)[10], and *United States Computer Emergency Readiness Team* (US-CERT)[11], etc.

To unify these VDBs, some efforts, usually resulting in meta vulnerability databases, are proposed. Most of them are independent from different VDB instances. The ultimate goal is to provide a unified view on different implementations and aspects of the same problem. An important role in this process is taken by the United States *National Cyber Security Division* (NCSD). The NCSD runs the US-CERT and sponsors vulnerability related projects, such as *National Vulnerability Database* (NVD), *Common Vulnerabilities and Exposures* (CVE) list [2], as well as *Common Vulnerability and Assessment Language* (OVAL) [3]. Besides, the *Common Vulnerability Scoring System* (CVSS) [4] maintained by the *Forum of Incident Response and Security Teams* (FIRST) is another recognized activity towards providing a unified view of exposed vulnerability information.

3 Data Structure for Modeling Vulnerability Information

As discussed above, a transitional database is usually required to link descriptions in VDBs and attack descriptions required by some automatic reasoning software, e.g., attack graph tools. A data structure for this intermediate database is proposed in this section, which can be used as an interface between vulnerability databases and attack graph construction tools.

3.1 Requirements

The required data structure should be capable of holding a broad range of knowledge. This information could be drawn from a number of different sources, such

[3] http://www.securityfocus.com/
[4] http://vdb.dragonsoft.com/
[5] http://secunia.com/advisories/
[6] http://xforce.iss.net/
[7] http://www.osvdb.org/
[8] https://cirdb.cerias.purdue.edu/coopvdb/public/
[9] http://doecirc.energy.gov/ciac/
[10] http://dnvd.nist.gov
[11] http://kb.cert.org/vuls/

as vulnerability databases, network scanners, policy parsers, firewall configurations, and user input. These pieces are based on semantically different things, although they all describe security aspects of a system. Also, different sources mean different semantics, because one source may describe and interpret data in another way, which might be different with what other sources do. This implies that a good data structure needs to be flexible or provide means to describe different aspects in different ways, while still providing a link between them.

Information on topology of the target network has been seen as an important type of knowledge by almost all of previous vulnerability models, such as [8] and [9]. It is relevant how hosts are connected, which systems can be reached from which computers, what firewall rules are active and restrict access, and how traffic flows are manipulated. Besides, vulnerabilities of installed software and used protocols are concerned, e.g., which version of software is vulnerable, what is necessary to exploit it, and what are the effects of such an exploitation. Some attack graph tools allow different types of attacker models, such as a novice attacker or an adversary with unlimited knowledge and resources. Moreover, information on deployed security policies gets more and more interests [10].

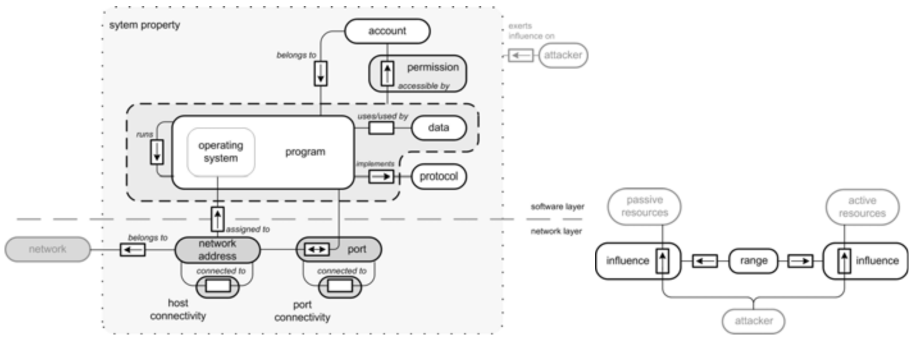
The existing vulnerability descriptions can be categorized as either *dynamic* or *static*. The network topology changes every time when a host joins or leaves the network. New software may be installed, reconfigured, or removed at any point in time, and the stored data can also change at any time. Therefore, a flexible structure is important. On the other hand, vulnerability information that are descriptions of vulnerabilities known to mankind, are generally static. Once identified and described they do not change.

For a useful data structure it is important to be related to a specific domain, because a clear conception of the modeled information eases the description of data and relationships between pieces of information. Therefore, some completely generic data structures, e.g., [11], do not meet the needs. The vulnerability information databases used as sources of input always refer to software vulnerabilities. This excludes weaknesses exploited through social engineering or direct physical access to a machine.

Additionally, [12], [13], and [14] have tried to use predicate and boolean logic to describe multi-step attacks. A predicate describes a property or a relation which objects may have in common. For example, data can or cannot be readable for a certain user. $is\_readable(x)$ then is a predicate which is true if x is readable and false if x is not readable. The use of predicates allows the description of a set of elements without the need to explicitly list them. Instead, the property all these elements have in common is described. In the context of attack descriptions, this allows to describe, for example, installed applications without the need to list all vulnerable versions, such as done by the NVD. The use of boolean logic gives further benefits. First of all, it reduces values to simple true or false characteristics. This increases the speed of evaluation of conditions, for example “is a vulnerable application installed” or “is a file readable to the attacker”. Also, boolean logic enables the connection of conditions, such as “application A is installed” *and* “write access to file F is given.”

3.2 A Conceptual Model of Computer Vulnerability

To propose a new data structure, several questions need to be considered, such as what information is used, how it is used, and how it relates to each other.



(a) System Properties (b) Influence Properties

Fig. 1. Properties of Computer Vulnerability

System properties are characteristics and resources of a computer system. Each *system property* describes one specific attribute of such a system, whereas *properties* are related to one another as depict in Figure 1(a). For example, the installed version of an application can be a system property. An application's version is meaningless if it cannot be linked to a certain application. Properties and their relations may change over time due to modifications, such that an application may be upgraded to a newer version. *System properties* can be found in two layers, the network layer and the software layer. The network layer describes properties of interconnected computers, such as *network addresses* and *port numbers*. The software layer describes properties of software systems, such as *programs*, *data*, and *account* information.

A *network* is a group of directly connected network addresses. A *network address* is an identifier of a host in a network. Directly connected means it is possible to reach from one host of network to another host of the same network. *Network addresses* may have a number of open *ports* per address which are used by *programs* to communicate with other *programs*.

The network layer properties also consist of *host connectivity* as well as *port connectivity*. Both are essential to capture which hosts and programs can be reached. *Host connectivity* is a boolean value to describe whether one host can be reached from another host. This may be influenced by the network the corresponding hosts are in or by firewall rules, preventing certain hosts to connect to others. *Port connectivity* is a boolean value to describe whether one port of a network address can be accessed from another port of a network address. Similar to *host connectivity*, this can be influenced by firewall rules or comparable system configuration tools.

The term *program* is used synonymously to application, service, and operating system. All of them are regarded as *programs*. *Programs* may run other *programs*, such as an operating system runs daemons or user applications, or an Internet browser allows to include add-ons. A *program* may use a number of *protocols* to communicate with other *programs*. A *protocol* is not necessarily linked to either a network port (e.g., TCP) or a network address (e.g., ICMP). *Protocols* may also be file-based or use other means to connect to programs. The decision to not distinguish between operating systems and applications was made to simplify the modeling of *programs*. Other approaches make a difference even between the operating system, services, and user-space applications, which can be omitted if all of them are regarded as *programs*.

Data is regarded as a collection of organized information. It is associated with *programs*, such as a database accessed through a database management system. Also, *data* may be used by many *programs* (e.g., the database may also be read by a file editor) and a *program* may use *data* from many sources (e.g., the database server reads from the database, but also from various configuration files). *Data* provide means for *programs* to exchange information. If you modify the *data* that a service provides, you effect all the consumers. This gives attackers two ways to influence a consuming *program*, either change the providing service or change the provided data.

Accounts are used to identify entities, e.g., users, and are interpreted by a *program*. Thus, *accounts* always belong to a *program*.

As shown in Figure 1(b), *influence properties* describe the relationship between a potential attacker and *system properties* which represent computer resources. The influence is different according to the type of resource. *System properties* can be categorized as either passively processed resources or actively processing resources. Passive resources are used by some kinds of processes, such as a file is read, or an account is created. The basic actions performed on passive resources are the creation and deletion as well as read and write operations. Active resources on the other hand implement a process which actively works on processed resources. A database application for example modifies the database and the operating system reads permissions to grant or deny access to users. Influence on passive resources is the creation and destruction as well as the reading and writing of data. For active resources, often the input data is influenced, e.g., to cause buffer overflows, the output data can be modified, e.g., to disguise the existence of malicious processes, or the existence of the active resource can be erased, e.g., in Denial of Service attacks. Vulnerabilities in server applications such as the Apache HTTP server can be exploited from remote machines without any local access to the target machine. This locality information can be specified with the *range* property and most commonly has the value *remote* or *local*. The *remote* indicates that an attack targets a resource on another host than the one from which the attack originates, whereas *local* indicates that the attack is aiming at a resource on the same host.

System properties represent the information which may be used or modified in an attack, while the attacker performs some kinds of actions on them. *Programs*

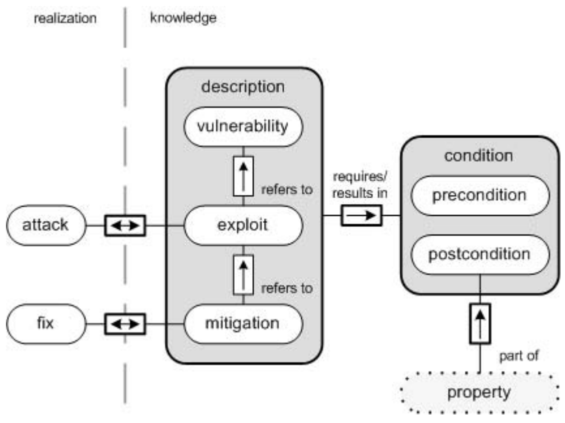


Fig. 2. A Conceptual Model of Computer Vulnerability

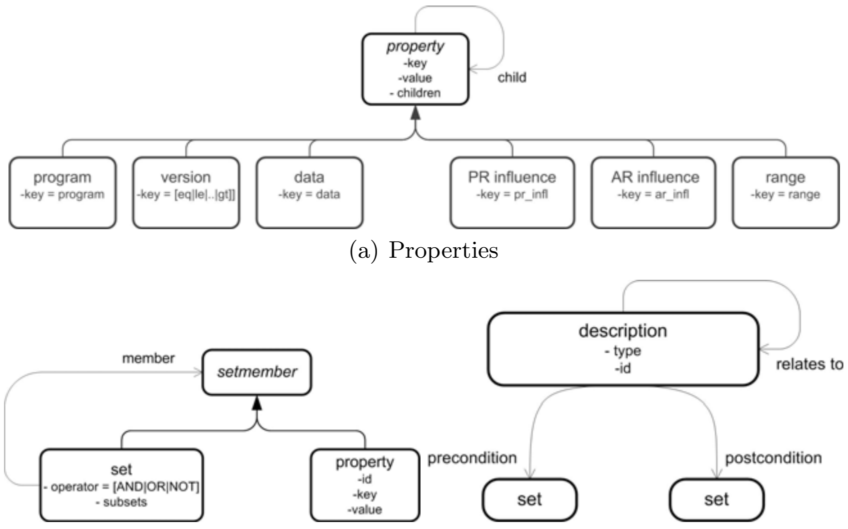
may be used, *data* can be read, or *ports* are blocked. Additionally, *influence properties* specify how *system properties* are used.

These properties are the essential elements to describe *preconditions* and *postconditions* of an attack. Together with the concepts mentioned in Section 2.1, *description* (i.e., *vulnerability*, *exploit*, and *mitigation*) and *condition* (i.e., *precondition* and *postcondition*), we propose a conceptual model of vulnerability in Figure 2. It shows that *condition* can be further described with the help of *properties*. *Conditions* may contain any number of *properties*, they can be anything from general to specific characterizations and only the relevant ones have to been known for a weakness. Based on this data, attack information can be linked and chained.

3.3 The Proposed Data Structure

With the requirements for vulnerability representation and the conceptual vulnerability model, it is now possible to propose a data structure. As outlined in the previous Section, *properties* will be used to describe attributes of relevant systems. These *properties* are then combined to *conditions* to specify the exact requirements and results of an attack. *Descriptions* are used to link *preconditions* and *postconditions*. Based on the conceptual model of the vulnerability information, we propose the new data structure which consists of *properties*, *sets* and *descriptions*.

Properties. As depict in Figure 3(a), properties are described with simple key-value pairs with the addition of a parent-child relationship among them. The key identifies the type of a property, for example, *program* or *data*. Value information then specify the corresponding instance, for example, *Windows XP*. This allows to translate information encoded in properties easily to relational database entries or serialization into an XML-based structure. An abstract base class can provide functionality which derived properties, such as *program* or *data*



(a) Properties

(b) Sets (c) Descriptions

Fig. 3. The Proposed Data Structure for Modeling Vunerability

properties, can re-use. This also allows to extend the data model easily if the need for further properties arises.

The child *attribute* of each property allows the definition of more complex *properties* similar to the composite design pattern, i.e., more details of a system can be described. Often, it is not sufficient to know that a certain *program* is running on a target host, but it has to be a specific version. An exploit which works for Windows XP Service Pack (SP) 1, may not work for Windows XP SP2. With child properties further details can be provided, such as the version information.

Note that the usage of *properties* implements a predicate logic. Each property describes an attribute different entities may have in common. Whether a certain entity has a characteristic can be checked with a simple comparison which returns either true or false. For example, to verify if a target host has Windows XP installed, it is checked if a *property* of type *program* with the value 'Windows XP' is assigned to this host.

Any two properties with the same key can be compared. If both have the same value, they represent the same predicate. But this does not automatically indicate that they represent the same entity. For this, the child properties have to be matched as well. If these all evaluate to equal values, the parent predicate will match, thus indicating that the compared objects are in the same category.

A set of predefined values for certain properties may be helpful, since many vulnerability descriptions refer to common values. For example, 'host' is a common generic identifier for the operating system used in many reports.

The complexity of nested properties in parent-child and possible grandchild relationships can be hidden behind a higher level programming interface. For example, programs and their corresponding versions can be hidden behind functionality which takes these two arguments and returns a single object. For a developer, only programs with a version attribute are visible.

Sets *Conditions* are represented as *sets*. A *set* is a collection of distinct objects with no repetition and insensible to an ordering of these objects. At the same time, a *set* is an object itself. To distinguish objects which can be part of a *set*, the abstract superclass *setmember* is defined (see Figure 3(b)). Aside from *set*, any *property* can be a *set* member. Therefore, a *set* may consist of other *sets* and *properties*. This leads to a similar nesting which is possible with parent-child relationship of *properties*, with similar evaluation characteristics. Because the atomic *set* members are *properties*, and *properties* represent predicates which are either true or false, a *set* resolves to either true or false as well.

With the help of *sets*, it is possible to group several *properties* which are not in parent-child relationship, but nevertheless relate to each other in terms of describing a system. *Properties* by themselves can for example only describe a specific program or a file, but with *sets* it is possible to describe a system where application A in version V is installed and at the same time another application B is installed or a file F is readable by the attacker.

To characterize the relationship of *set* members, the operator is used. This operator can take the value of the usual boolean operators of AND, OR, and NOT. AND indicates that all the *set* members must hold true for the *set* to evaluate to true. OR indicates that at least one of the members has to evaluate to true, and NOT, as a unary operator, negates the value of the *set*. If a *set* member is not a *property* but a *set* itself, this member has be evaluated before the parent *set* can be evaluated. Thus, *sets* allow to describe simple and complex *conditions* that a system is in, based on the aggregation of *properties* describing the system. Next, these *conditions* will be linked to represent vulnerability *descriptions*.

Descriptions Figure 3(c) depicts the data structure used to specify *descriptions*. Each description is of a certain type, either *vulnerability*, *exploit*, or *mitigation*. An *exploit* can be related to a *vulnerability* and a *mitigation*. Also, each *description* refers to a *precondition* and a *postcondition*. Note that several *preconditions* (e.g., different program versions) can be expressed with OR-connected *set* members. An identifier for each *description* can be stored as well, e.g., the CVE identifier. This is helpful to identify vulnerability information from different sources.

3.4 Applying the Data Structure to an Example

As shown in Figure 4, the vulnerability describing the previously mentioned ***Downadup*** worm is remodeled. The information can also be easily represented in an XML file, which is readable for both human beings and computer programs. A description of the type 'vulnerability' with the identifier 'CVE 2008-4250' represents the container for the vulnerability. It points to a *precondition* and a *postcondition*. Both conditions are *sets* which describe the state of a system. Members of a *set* are displayed as being inside the corresponding *set*. The

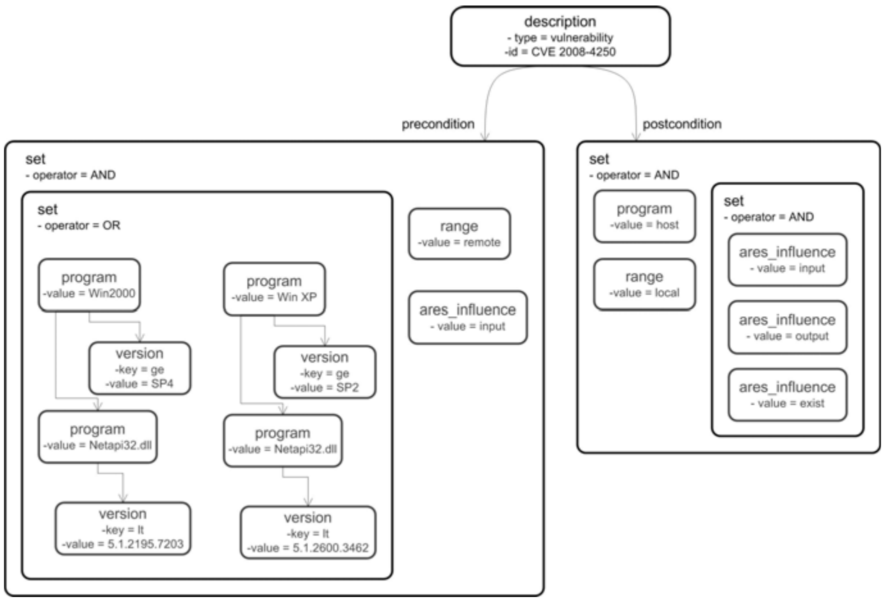


Fig. 4. Data Structure - Example

outermost *sets* have the AND operator, stating that all *set* members have to evaluate to true for the parent *set* to be true. The name of the *property* is the actual key used, e.g., a *range* property has the key 'range' and an additional value. This vulnerability involves more than 10 vulnerable system combinations, but only two are given in Figure 4. The arrow from one *property* to another indicates the parent-child relationship. In this case, a Windows 2000 program has the child version node value of Service Pack (SP) 4 or greater assigned and the 'Netapi32.dll'. Both *program* definitions are part of a *set* with the OR operator, which means either one can match for this *set* to be true.

The *precondition* includes a 'remote' *range* that an attacker does not need or has local access to the target system and can only influence the input data of the described programs. Because *range* and *influence* are the same for both programs, they can be included in the topmost *set*. Thus, the system state described by the *precondition* can be read as "Windows 2000 installations of SP 4 or later with 'Netapi32.dll' in version 5.1.2195.7203 as well as Windows XP installations of SP 2 or later with 'Netapi32.dll' in version 5.1.2600.3462, where an attacker can influence the input data remotely."

The *postcondition* can be more generic because the concrete program version is not relevant to describe the outcome of this vulnerability. Instead, a generic 'host' value is used to describe the affected program, in this case the entire host. This means that programs which evaluate this condition have to be aware of that 'host' indicates that the entire system is affected. All influences on active resources are possible to the attacker, i.e., on the input stream, the output stream, as well as on the existence. Everything can be done from a 'local' range.

4 Automatic Extraction of Meaningful Vulnerability Information

To apply and verify the proposed data structure, a vulnerability information extractor is required. The extractor includes several plugins for reading, transforming, and writing vulnerability information. The extracting components are referred to as readers, because they read information from a vulnerability database or some other sources. Every reader is able to extract information from a specific data source (i.e., a specific VDB, such as the NVD). The counterpart of readers are writers, which output vulnerability information in different formats. Gathered data can be read by various source, e.g., attack graph tools or vulnerability analysis programs. Thus, it is reasonable to provide a writer for each target application. All plugins provide a simple interface and communicate based on a common data structure. This allows to link existing plugins into a tool chain and therefore convert vulnerability information from any source format for which a reader exists to any target format for which a writer exists.

To prove the proposed concepts, we implement a prototype tool where the proposed data structure is used as an exchange format and the extractor is realized by several Reader and Writer plugins for providing compatibilities with known VDBs. Vulnerability information is transformed from VDBs to serve as input for the MulVAL tool [15], which is a known attack graph constructor. It was shown that sufficient data could be automatically transformed to identify an attack path in a company computer network using the proposed data structure as an intermediary format. Readers, such as the NVD Reader or the OVAL Reader, transform information from one XML representation into another XML representation, but the transformed information remains the same. The major benefit of this type of readers is the increased amount of available vulnerability information provided by a common vulnerability database, which is based on the data structure used in the implementation. The CVE Reader on the other hand extracts information from textual descriptions of vulnerabilities.

5 Conclusions

The contributions of this paper can be summarized as following: a) Proposing a unified data structure which can be used for representing the vulnerability information. b) Designing and implementing an easy way to extract and transform the vulnerability descriptions from existing databases to the proposed data model. The given structures are considered to be extendable and are not claimed to be complete. The goal is to represent the bulk of known vulnerabilities for a practical solution, not to provide a complete, but merely theoretical one. However, the data model is expected to be optimized to possibly cover all the information of a certain vulnerability description. Implementing more Reader and Writer plugins is helpful to extend the usage of the proposed model as well as to verify its completeness. A vulnerability exposure mechanism which can represent the vulnerability report directly using the given model is recommended.

References

1. CERT Vulnerability Analysis Blog, http://www.cert.org/blogs/vuls/ (accessed August 2009)
2. CVE Website, http://cve.mitre.org/ (accessed August 2009)
3. OVAL Website, http://oval.mitre.org/ (accessed August 2009)
4. Mell, P., Scarfone, K., Romanosky, S.: A Complete Guide to the Common Vulnerability Scoring System, Version 2.0. Technical Report, FIRST (June 2007)
5. Debar, H., Curry, D., Feinstein, B.: The Intrusion Detection Message Exchange Format, Internet Draft. Technical Report, IETF Intrusion Detection Exchange Format Working Group (July 2004)
6. Martin, R.A.: Transformational Vulnerability Management Through Standards Technical Report, MITRE Corporation (May 2005)
7. Roschke, S., Cheng, F., Schuppenies, R., Meinel, C.: Towards Unifying Vulnerability Information for Attack Graph Construction. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 218–233. Springer, Heidelberg (2009)
8. Phillips, C., Swiler, L.P.: A Graph-based System for Network-Vulnerability Analysis. In: Proceedings of the 1998 Workshop on New Security Paradigms (NSPW 1998), pp. 71–79. ACM Press, New York (September 1998)
9. Sheyner, O.M.: Scenario Graphs and Attack Graphs. PhD Thesis, CMU-CS-04-122, Carnegie Mellon University, USA (April 2004)
10. Jajodia, S., Noel, S.: Topological Vulnerability Analysis: A Powerful New Approach for Network Attack Prevention, Detection, and Response. In: Book Algorithms, Architectures, and Information Systems Security, pp. 285–306. World Scientific Press, Singapore (November 2008)
11. Schneier, B.: Attack Trees: Modeling Security Threats. Journal Dr. Dobb's Journal, http://www.ddj.com/architect/184411129 (December 1999)
12. Templeton, S.J., Levitt, K.: A Requires/Provides Model for Computer Attacks. In: Proceedings of the 2000 Workshop on New Security Paradigms (NSPW 2000), pp. 31–38. ACM Press, Ballycotton (September 2000)
13. Cuppens, F., Ortalo, R.: LAMBDA: A Language to Model a Database for Detection of Attacks. In: Debar, H., Mé, L., Wu, S.F. (eds.) RAID 2000. LNCS, vol. 1907, pp. 197–216. Springer, Heidelberg (2000)
14. Hale, J., Tidwell, T., Larson, R., Fitch, K.: Modeling Internet Attacks. In: Proceedings of the 2001 IEEE Workshop on Information Assurance and Security (IAS 2000), pp. 54–59. IEEE Press, West Point (June 2001)
15. Ou, X., Govindavajhala, S., Appel, A.W.: MulVAL: A Logic-based Network Security Analyzer. In: Proceedings of the 14th Usenix Security Symposium (SSYM 2005), p. 8. USENIX Association, Berkeley (August 2005)

Using Strategy Objectives for Network Security Analysis

Elie Bursztein[1] and John C. Mitchell[2]

[1] Stanford University and LSV, ENS Cachan, INRIA, CNRS
[2] Stanford University
{elie,mitchell}@cs.stanford.edu

Abstract. The anticipation game framework is an extension of attack graphs based on game theory. It is used to anticipate and analyze intruder and administrator concurrent interactions with the network. Like attack-graph-based model checking, the goal of an anticipation game is to prove that a safety property holds. However, expressing intruder goal as a safety property is tedious and error prone on large networks because it assumes that the analyst has prior and complete knowledge of critical network services and knows what the attacker targets will be.

In this paper we address this issue by introducing a new kind of goal called "*strategy objectives*". Strategy objectives mix logical constraints and numerical ones. In order to achieve these strategy objectives, we have extended the anticipation games framework with cost and reward. Additionally, this extension allows us to take into account the financial dimension of attacks during the analysis. We prove that finding the optimal strategy is decidable and only requires linear space. Finally we show that anticipation games with strategy objectives can be used in practice even on large networks by evaluating the performance of our prototype.

1 Introduction

With the increasing size and complexity of networks, attack modeling is now recognized as a key part of constructing an accurate network security for intrusion analysis, and prevention. Anticipation games (AG) [4] are an evolution of attack graphs based on game theory. More specifically, an anticipation game is a simultaneous game played between a network attacker and a network defender on a game-board consisting of a dependency graph. The dependency graph defines which services exist on the network and how they are related. The moves of the game do not change this dependency graph, but they do change the attributes, such as the compromise attribute which is associated with the nodes to reflect player's actions.

Typically an anticipation game is used to analyze how the network will be impacted by various attacks and how administrator actions can counter them. Using anticipation games instead of attack graphs offers the following advantages:

First it allows us to model the concurrent interaction of the intruder and the administrator with the network. For example, it is possible to model a case where the intruder is trying to exploit a vulnerability while the administrator is trying to patch it. Secondly, player interactions with the network are described by timed rules that use preconditions and postconditions written in a modal logic. Describing the model only with the network initial state and a set of rules relieves the security analyst from the tedious and error prone burden of explicitly describing each network state and the transitions between them. In AG the model-checking algorithm uses the set of rules to infer automatically every transition and network state reachable from the network's initial state [3]. As a result it is possible to express very large and complex models in a very compact form, which is handy while working on large networks and complex attacks.Thirdly, the use of timed rules allows us to model the temporal dimension of the attack. It captures the fact that each interaction with the network requires a different time. For instance, developing and launching an exploit is somewhat slower than downloading and launching a publicly available one. Modeling the time also models the so called "*element of surprise*" [7], which occurs when one player takes the other by surprise because he is faster. For example, when the administrator is patching a service she can be taken by surprise by the intruder if the intruder is able to exploit the vulnerability before the patch is complete.

Finally, since AG have been designed for network security analysis, they take into account network topological information such as dependency between network services, which allow them to model *collateral effects*. For example when a DNS server is unavailable due to a DDOS then by collateral effect, the web server is merely available because browsers can't perform DNS resolution.

Although using AG to analyze attacks provides a substantial improvement over standard attack graphs, there is still one side of attack modeling that remains tedious and error-prone: defining the analysis goal. So far as standard attack graph[16], the current AG analysis goal is to prove that a given safety property holds for a given model. However, network analysis makes the expression of security goal in term of reachability very hard because it is difficult to assert which services/hosts should be considered as a primary security objective, especially when working on large networks. Therefore in this paper we introduce a new kind of analysis goal called "**Strategy objectives**". Intuitively the idea is to combine a symbolic objective (logical formula) with numerical ones (time, cost, and reward). The logical formula is used to select *all the plays* that are valid strategies, and the numerical objectives are used to refine the analysis by selecting, among all of these possible strategies, *the one* that is the most relevant to the player according to his quantitative objectives. To the best of our knowledge this is the first time that symbolic and numerical objectives have been combined to express security goals. Note that being able to select the most relevant candidate is a central issue in network security as the number of possible candidates (*e.g* different attacks) to achieve a given goal is usually very large. The expressiveness offered by strategy objectives allows anticipation games to be used to answer a brand new range of question that more closely match administrator and security analysts needs. For example, using strategy objectives it is possible to answer the question: "*What is the most effective patching strategy in terms of cost or time ?*". Finally the introduction of action costs and rewards takes into account the financial dimension of attacks which is a central concern of network attacks. Taking into account action cost allows us to reason about the costs required to launch an attack, the loss induced by it, and the investment required to prevent it.

Our main contribution is the extension of AG with strategy objectives. This extension allows the analysis to answer key network security questions and to capture the financial dimension of the attack.

As far as we know, our extension the AG framework is the first attack model that covers both the financial and temporal aspects of attacks. Additionally we prove that the model-checking of AG with strategy objectives is decidable, and that deciding if a play is a valid strategy can be done in linear time. We also prove that using strategy objectives instead of a safety property adds only a linear space complexity to the analysis. The evaluation performed with our prototype shows that in practice this framework can be used to find strategy for large networks (Thousands of nodes). This evaluation also demonstrates that practical results are consistent with the theoretical bounds we have proven.

The reminder of this paper is organized as follows. In Sect. 2, we will survey related work. In Sect. 3, we recall what an anticipation game is and present how we have extended it to take into account cost and reward. We also present the game example that is used as a guideline for the rest of the paper. Sect. 4 details how strategies objectives are expressed and contains the strategy decidability and space complexity proofs. In sect. 5, we evaluate the impact of using strategy in term of speed and memory with our prototype. We show that our experiments are consistent with the theory and that strategies can be used in practice.

2 Related Work

Model checking for attack graphs was introduced by Ammann and Ritchey [16]. It is used to harden security [12]. Various methods have been proposed for finding attack paths, i.e., sequences of exploit state transitions, including logic-based approaches [17] and graph-based approaches [13].Research has also been conducted on formal languages to describe actions and states in attack graphs [5].Anticipation games are based on timed automata, timed games, and timed alternating-time temporal logic (TATL) [8], a timed extension to alternating-time Kripke structures and temporal logic (ATL) [1]. The TATL framework was specifically introduced in [7]. The notion of cost for attack appears in [6]. Mahimkar and Shmatikov have used the game theory to model denial of service in [11] The use of games for network security was introduced by Lye and Wing [10]. The Anticipation Game framework was introduced in [4]. A dedicated model-checker called NetQi [3] has been developed to accommodate anticipation game specificities.

3 Anticipation Games with Cost and Rewards

This section briefly recalls what an *anticipation game* (AG) is and explains the extension made to introduce strategy in the model. Intuitively, an AG can be represented as a graph. Each node of the graph describes the network state at any given moment; *e.g*, each state describes which services are compromised at this moment. The transitions represent the set of actions that both players, the administrator and the intruder, can perform to alter the network state. For example, an edge may represent the action of removing a service from the vulnerable set by patching it.

3.1 Network State

The network state is represented by a graph called a *Dependency Graph* (DG) and a finite set of states. DGs are meant to remain fixed over time and describe the relation between services and files. The figure 1 presents the DG used as an example in this paper. DG vertices are services and files present on the network and the set of directed edges is used to express the set of dependencies between them. In the example below, the direct edge that links the vertex *Email server (5)* to the vertex *User database (6)* is used to denote that the *email server* depends on the *user database* to identify its clients.

States	1	2	3	4	5	6
$\rho(\mathsf{Vuln})$	$\perp$	$\perp$	$\perp$	$\top$	$\top$	$\perp$
$\rho(\mathsf{Public})$	$\perp$	$\perp$	$\perp$	$\top$	$\top$	$\perp$
$\rho(\mathsf{Compr})$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
$\rho(\mathsf{NeedPub})$	$\perp$	$\perp$	$\perp$	$\top$	$\top$	$\perp$

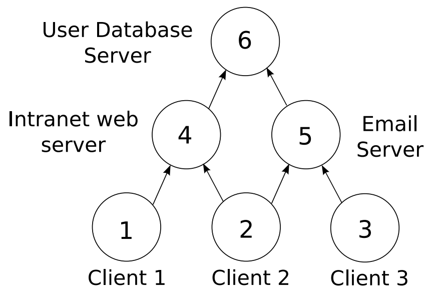


Fig. 1. The intial network state (left) and the dependency graph (right)

The set of *variables* (figure 1) is used to model information that does evolve over time. Intuitively this set describes which services and files are currently public, vulnerable, compromised, and so on. More formally, let $\mathcal{A}$ be a finite set of so-called *atomic propositions* $A_1, \ldots, A_n, \ldots$, denoting each base property. Thus each atomic proposition is true or false for each DG vertex. The complete initial mapping used in the example is detailed in figure 1. This mapping indicates that the *email server* and the *web server* need to be public, are vulnerable, and are public because ($\rho(\mathsf{NeedPub})$), $\rho(\mathsf{Public})$, and ($\rho(\mathsf{Vuln})$) return true ($\top$). It also indicates that no vertex is compromised as $\rho(\mathsf{Compr})$ returns false ($\perp$) for every vertex. Finally the set $\rho(\mathsf{NeedPub})$ is used by the `Unfirewall` rules to know which vertices should be made public.

3.2 Player's Actions

To describe which actions are legal for each player a set of timed rules is associated to the AG. Each rule is of the form $\mathbf{Pre}\ F \xrightarrow{\Delta,p,a,c} P$ where F is the *precondition*, stating when the rule applies, Δ is the amount of time needed to fire the rule, p is the name of the player that originates the rule, a is an action name, c is the rule cost, and P is a *command*, stating the effects of the rule. It is required for the precondition F to hold not just when the rule is selected, but during the whole time it takes the rule to actually complete (Δ time units). For example consider the following rule :

$$\mathbf{Pre}\ Vuln \wedge Public \xrightarrow{(30,I,Compromise,500)} Compr$$

This says that the intruder can compromise a vertex if it is vulnerable (`Vuln`) and public (`Public`) in 30 units of time. Compromise means here that the targeted vertex will be added to the state `Compr`. If the intruder chooses to apply this rule to the *Email server* then it is required that the preconditions are fulfilled when he chooses to apply it but also after the 30 units of time required to execute it because the network state might have changed during this time due to administrator actions. For example, the administrator could firewall the targeted vertex. In this case, the vertex is not public anymore, the intruder is taken by surprise, and the compromise rule fails. An AG play is a path (a sequence of action and states) $\rho : s_0 r_0 s_1 r_1 ...$ where $\forall j : s_j \rightarrow^{r_j} s_{j+1}$, s_j and $s_(j+1)$ are network states, and r_j is the rule used to make the transition.

3.3 Extending Anticipation Game for Strategy

Using strategy and analyzing the financial dimension of the attack requires that we extend the framework with costs and rewards. The natural way to do so is to add a cost to rules and an a reward to each DG vertex.

Costs are added to rules because it is obvious that some actions are more costly than others. For example, coding an exploit is more costly than using an existing one. Similarly, rewards are bound to DG vertices because some services and files are more valuable than others. In our example (figure 1), it is obvious that the *user database* is more important than any client. Formally we have a function $\mathsf{Value}(x) \rightarrow y / y \in \mathbb{N}$ that returns the value y associated to the DG vertex x. Costs are naturally added to rules because a rule execution is equivalent to a player action on the network. To take into account the fact that not all the rules grant a reward we use two types of rules: *regular rules* that have an execution cost and *granting rules* that have an execution cost and grant a reward. For example if the administrator objective is to secure her network, then firewalling a service (removing it from the `Public` set) will prevent it from being compromised but it is a temporary measure, and therefore should not grant a reward. At the opposite end of the spectrum, patching the service (removing it from the `Vuln` set) is a permanent measure and grants a reward. Note that computing the value of network assets is a topic by itself [14] and we we assume that one of the existing methods is used to compute the rewards associated to vertices.

3.4 Player Rules

The set of rules used for the example focuses on intrusion and is meant to be very general. It is just meant to give a flavor of what is possible with our model. It follows that the cost and time associated with each rule are meant to be on the order of magnitude of what is commonly accepted, but these measures are not necessarily completly accurate. The seven rules used in the example are shown in Figure 2.

We take the convention that a `granting rule` uses the $\Longrightarrow$ double arrow and that a `regular rule` uses the $\longrightarrow$ single arrow. Rules `Compromise 0day`(1) and `Compromise Public`(2) say that if a vertex is vulnerable (`Vuln`), public (`Public`) and not compromised (`Compr`), then it can be compromised. The difference between the two is the time required to compromise the service (2 or 7 units) and the cost required (20000 or 5000). The use of these two rules allows us to express the fact that

1) **Pre** : $Vuln \wedge Public \wedge \neg Compr$
   $\Longrightarrow$ 2, I, Compromise 0day, 20000
   Effect : $Compr$
2) **Pre** : $Vuln \wedge Public \wedge \neg Compr$
   $\Longrightarrow$ (7, I, Compromise public, 5000)
   Effect : $Compr$
3) **Pre** : $\neg Compr \wedge \Diamond Compr$
   $\Longrightarrow$ (4, I, Compromise backward, 5000)
   Effect : $Compr$
4) **Pre** : $Compr \wedge \Diamond \neg Compr$
   $\Longrightarrow$ (4, I, Compromise forward, 5000)
   Effect : $\Diamond Compr$
5) **Pre** $Public \wedge Vuln$
   $\longrightarrow$ (1, A, Firewall, 10000)
   Effect $\neg Public$
6) **Pre** $\neg Public \wedge \neg Vuln \wedge NeedPub$
   $\longrightarrow$ (1, A, UnFirewall,0)
   Effect $Public$
7) **Pre** $Vuln \wedge \neg Compr$
   $\longrightarrow$ (3, A, Patch, 500)
   Effect $\neg Vuln \wedge \neg Compr$

Fig. 2. Set of rules

using a 0 day exploit instead of a public exploit provides an advantage in terms of time and a disadvantage in terms of cost. To be consistent with this idea, the administrator `patch` rule (7) is slower than the `compromise 0day` rule and faster than the `compromise public` one. These three rules model the windows of vulnerability [9]. The rule `Compromise backward` says that the intruder can take advantage of a dependency relation to compromise a vertex that depends on a compromised one. The modal operator [2] $\Diamond$ allows preconditions and postconditions to speak about vertice successor. For example $\Diamond Compr$ means "*there exists a successor that is compromised*". This operator is used to model attacks that exploit trust relationships and collateral effects such as the attack where a compromised DNS server is used to redirect clients to spoofed sites. Similarly the rule `Compromise forward` (4) says that the intruder can take advantage of a dependency relation to compromise the successor of a compromised vertex. In our DG example (figure 1) if the intruder is able to compromise the intranet server, he can look in its configuration files to steal database credentials. The rule `Firewall` (5) says that if a service is vulnerable (`Vuln`) and Public (`Public`) it can be firewalled. The cost of the rule is very high (10000) compared to the patch rule cost (500) because firewalling a public service will indeed prevent the intruder to access it but also forbids legitimate access. Thus this action induces an activity disturbance and a possible financial loss. Notice the $\longrightarrow$ arrow of this rule that denotes that no reward is granted. Finally the rule `Unfirewall` (6) is used to make public services that are not vulnerable and need to be public (`NeedPub`).

3.5 Play Example

The play used as an example (figure 3) is an intruder strategy that aimed at compromising the network. Due to space constraints, rules name have been truncated. Column `Ti` stands for time, `Pl` for players, `Act` for action, `Ta` for target, `S` for successor node, `Pa` for payoff and `C` for cost. Furthermore, `I` is for intruder and `A` is for admin. Every strategy presented in this paper is the output result of NetQi using the DG, the initial mapping set, and the set of rules presented above, along with various strategy objectives. Even if this example seems simple, it still cannot be analyzed by hand because this game configuration leads to 4011 distinct plays.

Ti	Pl	Act	Rule	Ta	S	Pa	C
0	I	choose	Comp 0 Day	4	⊥	-	-
0	A	choose	Firewall	4	⊥	-	-
1	A	**execute**	Firewall	4	⊥	0	10000
1	A	choose	Patch	4	⊥	-	-
2	I	`fail`	Comp 0 Day	4	⊥	0	20000
2	I	choose	Comp 0 Day	5	⊥	-	-
4	I	**execute**	Comp 0 Day	5	⊥	31	40000
4	I	choose	Comp For	5	6	-	-
4	A	**execute**	Patch	4	⊥	21	10500
4	A	choose	UnFirewall	4	⊥	-	-
5	A	**execute**	UnFirewall	4	⊥	21	10500
5	A	choose	Patch	5	⊥	-	-
8	I	**execute**	Comp For	5	6	1382	45000
8	I	choose	Comp Back	2	5	-	-
8	A	**execute**	Patch	5	⊥	52	11000
12	I	`fail`	Comp Back	2	5	1382	50000
12	I	choose	Comp Back	4	6	-	-
16	I	**execute**	Comp Back	4	6	1403	55000
16	I	choose	Comp Back	1	4	-	-
20	I	**execute**	Comp Back	1	4	1404	60000
20	I	choose	Comp Back	2	4	-	-
24	I	**execute**	Comp Back	2	4	1405	65000
24	I	choose	Comp For	2	5	-	-
28	I	**execute**	Comp For	2	5	1436	70000
28	I	choose	Comp Back	3	5	-	-
32	I	**execute**	Comp Back	3	5	1437	75000

Fig. 3. Play example Intruder maximum payoff

The figure 3 example is read as follows: At *time 0* the intruder chooses to use an 0 day exploit against the Web server (Target 4). At the same time the administrator starts firewalling the Web server. Because firewalling is faster than exploiting the 0 Day vulnerability, the administrator is able to firewall the web server before the 0day exploitation is successful (*time 1*). The administrator starts to patch the web server. At *time 2* the intruder is taken by surprise by the administrator because the web server is firewalled before his exploitation is successful, hence the rule execution fails. He chooses to try another 0day exploit against the email server (target 5, *time 2*). At *time 4* the administrator has finished patching the web server and decides to unfirewall it since it is no longer vulnerable. Meanwhile, the intruder compromises the email server and decides to use his newly gained access to compromise the user database (target 6). At *time 5* the administrator decides to patch the email server (target 4). At *time 8* the intruder has compromised the user database (target 6). At the same moment, the administrator has finished patching the email server (node 5). Therefore at *time 12* the intruder fails to compromise the client 2 from the email server (Succ 4) because the email server is no longer vulnerable and compromised. However the intruder still has access to the database user server (node 6) and he uses this access to compromise the web server (*time 16*). From there he compromises the client 1 (*time 20*) and the client 2 (*time 28*). He uses his access on client 2 to compromise the web server again (node 5, *time 28*) and finally owns the network by compromising the client 3. This play illustrates that the interaction between players leads to very complex plays even when the initial situation is simple. This emphases that analyzing administrator and intruder interactions on a real network cannot be achieved by hand.

4 Strategy Objectives

In game theory, a strategy is the optimal succession of actions (plays) that a player can perform to achieve his goal. As said previously, translating real world network security goals into reachability properties is not expressive enough and is also error-prone. Therefore in this section, we introduce a new kind of analysis goal called **strategy objectives** that combine symbolic constraints and numerical objectives by leveraging the notion of cost and reward introduced previously.

Symbolic constraints, expressed as a CTL logical formula, are used to express which plays are acceptable strategies. Numerical objectives are used to select among these potential candidates, the one that fulfill the most player interests. To the best of our knowledge this the first time that symbolic and numerical objectives have been combined to express analysis goal. Strategy objectives allow security analysts to express naturally many network security goals. For example it allow security analysts to express that the goal of the administrator is to patch her network (logical formula) in minimum amount of time and for the lowest cost possible (numerical constraints). More formally we define strategy objectives as :

Definition 1 (Strategy objectives). *A set of strategy objectives is the tuple* $S : (name, \mathsf{P}, \mathcal{O}, \mathcal{R}, \varphi)$ *where* `name` *is the strategy name,* P *is its owner,* $\mathcal{O}$ *is the set of numerical objectives,* $\mathcal{R}$ *is the numerical objectives priority strict order, and* φ *is the logical formula that a play needs to satisfy to be a valid strategy.*

4.1 Numerical Objectives

Numerical objectives $\mathcal{O}$ are assigned on play outcomes ϕ:

Definition 2 (Play outcomes). *are the unordered set of natural numbers* $\phi : \{\mathtt{payoff}, \mathtt{cost}, \mathtt{opayoff}, \mathtt{ocost}, \mathtt{time}\}$ *where* `payoff` *is the player payoff ,* `cost` *is the player cost ,* `opayoff` *is the player opponent payoff,* `ocost` *is the opponent cost, and* `time` *is the duration of the play.*

The player P payoff for the play ρ is the sum of all the rewards granted by the successful execution of his granting rules. A rule reward is the value of the DG vertex targeted by the rule execution. The players P cost for ρ is the sum of all executed rule costs whether they are successful or not, because regardless of its success the player has invested the same amount of resource in it. A strategy numerical objective is either the maximization or the minimization of one of these play outcome. For instance, an administrator might want to find a patching strategy that *minimizes* the *cost* and the *time*.

4.2 Symbolic Constraints

Symbolic constraints are used to express which plays can be considered valid strategies. In the patch strategy example, valid plays are those in which every vulnerable service is patched. An additional constraint can be that no services are compromised. This constraint has two possible interpretations that lead to two very different results: first, it can mean that at the end of the play no service is compromised but that at some point a service could have been compromised and restored. Secondly, it can mean that no service is **ever** compromised during the play. The difference between the two interpretations is that with the first interpretation, having a service compromised for a brief moment is acceptable, whereas with the second interpretation, it is not.

To express the second type of constraint the CTL [2] operator $\Box$ is needed. This operator is used to express the fact that a constraint needs to be true for every state of the play. We also use the $\Diamond$ operator to express the fact that a constraint needs to be true at some point. This operator is used, for example to express information leak strategy where at some point a service was compromised. Thus the strategy formula is expressed in the following fragment of CTL:

$$\begin{array}{lll} \varphi ::= & A & \text{Atomic proposition} \\ | & \neg\varphi & \\ | & \varphi \wedge \varphi & \\ | & \forall A & \\ | & \exists A & \\ | & \Box\varphi & \\ | & \Diamond & \end{array}$$

We take the convention that if a constraint is specified without the $\Diamond$ and the $\Box$ operator then this constraint has to be true only on the last state of the play. Patching strategy symbolic constraints are used to ensure that no vertex is ever compromised (belongs to the set `Compr`) and that every vertex is not vulnerable at the end of the play are written: $\Box\neg Compr \wedge \neg Vuln$ in our CTL fragment.

4.3 Dominant Strategy

The natural question that arises is what class of strategy objectives should be considered for network security. A naive idea would be to consider the class of objectives that minimizes/maximizes player cost/reward only and ensures by a set of constraints that the player goals are fulfilled. The following patching strategy objectives minimize the cost and ensures that no vertex is ever compromised and that every vertex is not vulnerable at the end of the play is:

$$S : (patch, Admin, MIN(Cost), Cost, \Box\neg Compr \wedge \neg Vuln))$$

However these strategy objectives lead to an incorrect strategy because the cost is minimal when the opponent makes "mistakes" :

Ti	Pl	Act	Rule	Ta	S	Pa	C
0	I	choose	Comp Public	4	$\perp$	-	-
0	A	choose	Patch	5	$\perp$	-	-
3	A	**execute**	Patch	5	$\perp$	31	500
3	A	choose	Patch	4	$\perp$	-	-
6	A	**execute**	Patch	4	$\perp$	52	1000
7	I	`fail`	Comp Public	4	$\perp$	0	5000

The intruder could have been more effective. For example, he could have used an 0 day exploit at the beginning. Thus, a more interesting class of strategies to consider for network security is the ones that minimize/maximize the cost/reward and are successful whatever the opponent does. These strategies are the best set of actions against the worst case. Thus, in our patching strategy example, the administrator wants to have the least costly patching strategy that is effective even against the worst case attack. This class of strategies is computed by adding objectives that maximize/minimize the opponent's cost/reward. They are commonly called strictly dominant strategies [15]. Dominant strategies are effective against the worst case but also every less effective case. In other words, when a player has a strictly dominant strategy and he performs it, he is able to fulfill his objective regardless of what his opponent does. From the network security perspective, it means that when the administrator has a dominant strategy for a given set of goals, whatever her opponent do, this strategy will ensure her that she will always accomplish her objective. In the example, a dominant strategy exists for the administrator.When used, it ensures that the administrator, regardless of the intruder actions, will be able to patch the two vulnerable services before any service is compromised.Note that in the general case dominant strategies do not always exist.

4.4 Complexity

We now study the decidability of finding the best strategy for a given set of strategy objectives. The key issue is that plays can be infinite. However, they are ultimately periodic paths because an AG is finite and therefore even when a game has an infinite play it is possible to decide which play is the best for a given set of objectives. To decide so, two things must be known: the play outcome and if the play satisfies the logic formula used to express the symbolic constraints. For the play outcome we have the following result:

Lemma 1. *An infinite play outcome can be computed by examining a short finite prefix.*

For formula satisfiability we have the following result:

Lemma 2. *For an infinite play the decidability of objectives formula satisfaction can be reduced to verifying the formula on a short finite prefix.*

This leads us to the central theorem for decidability:

Theorem 1. *Deciding if a play is the one that fulfills the most strategy objectives is decidable for any play by looking at a finite number of states.*

Moreover we can prove that deciding if a play satisfies the most strategy objectives can be done in polynomial time:

Theorem 2. *Deciding if a play satisfies the most strategy objectives can be decided in polynomial time* $\mathsf{O}(s \times |\varphi|)$ *where* s *is the number of states in the finite prefix and* φ *is the formula to verify.*

Ultimately we have the general decidability theorem:

Theorem 3 (Decidability). *Finding the strategy that fulfills the most strategy objectives over an anticipation game is decidable.*

A key property for implementation is that the memory required to find a strategy for a given set of objectives is linear:

Theorem 4. *Memory space complexity worst case is:* $WC = Set \times V \times R \times (S+1)$ *Where Set is the finite memory space required to hold sets mapping values, V is the number of vertices of the dependency graph, R is the number of rules and S is the number of strategies researched.*

5 Evaluation

In order to evaluate that AG can be used in practice, we have conducted a set of evalutions with our prototype on a standard Intel 2.9GHz core 2 Linux. Each benchmark was run three times and the reported time is the mean. Two sets of benchmarks have been conducted. The first set was used to determine if the AG framework is usable in practice. The second set was used to measure the impact of strategy on the analyzer performance and ensure that they are consistent with theoretical bounds.

The first set of benchmarks was done by running the analyzer against a large example. This example uses the set of rules presented in the Sec. 3.4, an initial random network state, and looks for the intrusion and defense strategy presented in Sec 4. The random initial state is composed of 5200 nodes, 27000 dependencies and 3 random vulnerabilities. Each of the 200 server nodes has 10 random dependencies and each of the 5000 client nodes has 5 random dependencies. To ensure that the generated initial state is not a degenerate case, we used 10 different initial states. Our prototype has the same performance regardless of the initial states used. In order to deal with such a large example, our analyzer uses numerous optimization tricks, including a static analysis of the dependency graph shape and a static analysis of the rules set based on the shape of the symbolic constraints [3]. Benchmark results presented in figure 4 show that, in practice AG can be used to analyze a new situation even for a complex network in a matter of seconds.

Nb Nodes	Nb Dep	Strategy	type	Time
5200	27000	Defense	Exact	2.4 sec
5200	27000	Intrusion	Approximate	55 sec

Fig. 4. Analyzer performance benchmark

We evaluate the performance impact of using strategy objectives instead of verifying a security property by conducting two types of benchmark. The first type was designed to measure the impact of strategy on analyzer speed and the second to measure memory usage. In order to have the most accurate evaluation possible, all analyzer optimizations were disabled. The game we choose as a baseline for our benchmark is an expanded version of the example presented in this paper with ten additional clients. Without optimization, adding clients increases drastically the interleaving generated by the rules `compromise forward` and `compromise backward`. For this speed performance benchmark, we used as a baseline the time required by the analyzer to run every possible play generated by the game without strategy. Then we ran the analyzer with an increasing number of strategy requests. The requested number of strategies ranges between 0 and 100. Every strategy symbolic objectives contains a $\Box$ constraint to ensure that we are in the worst case possible: Our prototype has to evaluate the symbolic constraints satisfaction on every state of every play. Experimental results show that that the time requested to analyze the game grows linearly in the number of strategies, which is consistent with theorem 2. Secondly, we have used the memory profiler **massif** from the **valgrind** tool suite to verify that the memory needed by the analyzer grows linearly in the number of strategies as proved in theorem 4.

6 Conclusion

In this paper we have introduced strategies for anticipation games. We have shown that using strategy objectives as analysis goals allow us to find answers to key security questions such as ”What is the best patching strategy in term of time and cost”. We have described how our extension takes into account the financial dimension of the network security, making the AG the first framework that deals with time and the financial dimension of attack at the same time. We have proved that finding the strategy that fulfills the most strategy objectives is decidable and that it only requires a linear memory space. Finally, we have demonstrated the suitability of AG with strategies for practical uses by fully implementing AG with strategies in our prototype . Future work involves extending strategies with non-determinism to model attackers with various levels of knowledge and skill.

References

1. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM 49(5), 672–713 (2002)
2. Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, P.: Systems and Software Verification. In: Model-Checking Techniques and Tools. Springer, Heidelberg (2001)
3. Bursztein, E.: NetQi: A model checker for anticipation game. In: Cha, S(S.), Choi, J.-Y., Kim, M., Lee, I., Viswanathan, M. (eds.) ATVA 2008. LNCS, vol. 5311, pp. 246–251. Springer, Heidelberg (2008)
4. Bursztein, E., Goubault-Larrecq, J.: A logical framework for evaluating network resilience against faults and attacks. In: Cervesato, I. (ed.) ASIAN 2007. LNCS, vol. 4846, pp. 212–227. Springer, Heidelberg (2007)
5. Cuppens, F., Ortalo, R.: Lambda: A language to model a database for detection of attacks. In: Debar, H., Mé, L., Wu, S.F. (eds.) RAID 2000. LNCS, vol. 1907, pp. 197–216. Springer, Heidelberg (2000)
6. Dacier, M., Deswarte, Y., Kaaniche, M.: Models and tools for quantitative assessment of operational security. In: 12th International Information Security Conference, pp. 177–186 (May 1996)
7. de Alfaro, L., Faella, M., Henzinger, T., Majumdar, R., Stoelinga, M.: The element of surprise in timed games. In: Amadio, R.M., Lugiez, D. (eds.) CONCUR 2003. LNCS, vol. 2761, pp. 144–158. Springer, Heidelberg (2003)
8. Henzinger, T., Prabhu, V.: Timed alternating-time temporal logic. In: Asarin, E., Bouyer, P. (eds.) FORMATS 2006. LNCS, vol. 4202, pp. 1–17. Springer, Heidelberg (2006)

9. Lippmann, R., Webster, S., Stetson, D.: The effect of identifying vulnerabilities and patching software on the utility of network intrusion detection. In: Wespi, A., Vigna, G., Deri, L. (eds.) RAID 2002. LNCS, vol. 2516, pp. 307–326. Springer, Heidelberg (2002)
10. Lye, K.-w., Wing, J.M.: Game strategies in network security. Int. J. Inf. Sec. 4(1-2), 71–86 (2005)
11. Mahimkar, A., Shmatikov, V.: Game-based analysis of denial-of-service prevention protocols. In: 18th IEEE Computer Security Foundations Workshop (CSFW), Aix-en-Provence, France, pp. 287–301. IEEE Computer Society, Los Alamitos (June 2005)
12. Noel, S., Jajodia, S., O'Berry, B., Jacobs, M.: Efficient minimum-cost network hardening via exploit dependency graphs. In: 19th Annual Computer Security Applications Conference, pp. 86–95 (December 2003)
13. Noel, S., Jajodia, S.: Managing attack graph complexity through visual hierarchical aggregation. In: VizSEC/DMSEC 2004: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, pp. 109–118. ACM Press, New York (2004)
14. Pamula, J., Jajodia, S., Ammann, P., Swarup, V.: A weakest-adversary security metric for network configuration security analysis. In: QoP 2006: Proceedings of the 2nd ACM Workshop on Quality of Protection, pp. 31–38. ACM Press, New York (2006)
15. Rasmusen, E.: Games and Information. Blackwell Publishing, Malden (2007)
16. Ritchey, R.W., Ammann, P.: Using model checking to analyze network vulnerabilities. In: SP 2000: Proceedings of the 2000 IEEE Symposium on Security and Privacy, Washington, DC, USA, pp. 156–165. IEEE Computer Society, Los Alamitos (2000)
17. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: SP 2002: Proceedings of the 2002 IEEE Symposium on Security and Privacy, Washington, DC, USA, pp. 273–284. IEEE Computer Society, Los Alamitos (2002)

A DAA Scheme Requiring Less TPM Resources

Liqun Chen

Hewlett-Packard Laboratories
liqun.chen@hp.com

Abstract. Direct anonymous attestation (DAA) is a special digital signature primitive, which provides a balance between signer authentication and privacy. One of the most interesting properties that makes this primitive attractive in practice is its construction of signers. The signer role of DAA is split between two entities, a principal signer (a trusted platform module (TPM)) with limited computational capability and an assistant signer (a computer platform into which the TPM is embedded) with more computational power but less security tolerance. Our first contribution in this paper is a new DAA scheme that requires very few TPM resources. This new scheme has better performance than the existing DAA schemes and is provable secure based on the q-SDH problem and DDH problem under the random oracle model. Our second contribution is a modification of the DAA security model defined in [12] to cover the property of non-frameability.

Keywords: direct anonymous attestation, trusted platform module, bilinear map.

1 Introduction

Many types of digital signatures have been developed to achieve signer authentication as well as signer privacy. Generally speaking, there are three categories of signature primitives depending on which type of public keys is used for signature verification. Given a signature, if a verifier makes use of the signer's public key, like an ordinary signature scheme, that shows the signer's unique information and therefore this type of signatures does not provide signer privacy. If a verifier makes use of a set of public keys each binding to one potential signer, such as ring signatures, designated verifier signatures and concurrent signatures, signer privacy is held but the level of privacy is dependent on the size of the public key set. If a verifier makes use of a group public key, such as group signatures and Direct Anonymous Attestation (DAA), signer privacy is also held and the level of privacy is dependent on the size of the group.

The concept and a concrete scheme of DAA were first introduced by Brickell, Camenisch, and Chen [10] for remote anonymous authentication of a trusted computing platform. The first DAA scheme was adopted by the Trusted Computing Group (TCG). The DAA scheme was specified in the TCG TPM Specification Version 1.2 [36] that has recently been adopted by ISO/IEC as an international standard [27]. A historical perspective on the development of DAA was provided by the DAA authors in [11]. Since then, DAA has drawn a lot of attention from both industry and cryptographic researchers, e.g. [2,3,29,32,33,35].

The following two unique properties make DAA attractive in practice.

The first one is that the signer role of DAA is split between two entities, a principal signer with limited computational and storage capability, e.g. a trusted platform module (TPM), and an assistant signer with more computational power but less security tolerance, e.g. an ordinary computer platform (namely the Host with the TPM embedded in). The TPM is the real signer and holds the secret signing key, whereas the host helps the TPM to compute the signature under the credential, but is not allowed to learn the secret signing key and to forge such a signature without the TPM involvement.

The second one is to provide different degrees of privacy. A DAA scheme can be seen as a special group signature scheme without the feature of opening the signer's identity from its signature by the issuer. Interactions in DAA signing and verification are anonymous, that means the verifier, the issuer or both of them colluded cannot discover the signer's identity from its DAA signature. Instead of full-traceability as held in group signatures [4], DAA has user-controlled-traceability, that we mean the DAA signer is able to control if a verifier enables to determine whether any two signatures have been produced by the same signer. Moreover, the signer and verifier may negotiate as to whether or not the verifier is able to link different signatures signed by the signer.

The original DAA scheme [10] and another DAA scheme by Ge and Tate [26] are based on the strong-RSA problem. We call them RSA-DAA for short. In an independent work [16], Canard and Traore proposed the concept of list signatures, in which signatures within a certain time frame are linkable. This property is similar to the user-controlled-traceability in DAA. Also in [16], the authors proposed a concrete list signature scheme, that, as the same as the first DAA scheme, is based on the strong RSA problem.

Recently, many researchers have been working on how to create DAA schemes with elliptic curves and pairings. We call these DAA schemes ECC-DAA for short. Generally speaking, ECC-DAA is more efficient in both computation cost and communication cost than RSA-DAA. The TPM's operation is much simpler and the key/signature length is much shorter in ECC-DAA than in RSA-DAA.

To our best knowledge, there are five ECC-DAA schemes available. The first one was proposed by Brickell, Chen and Li [12,13]. This scheme is based on symmetric pairings. For the purpose of increasing implementation flexibility and efficiency, Chen, Morrissey and Smart proposed two extensions of this scheme [20,21,22]. Their schemes are based on asymmetric pairings. A flaw in the first one was pointed out by Li and further discussed in [19,22]. Security of these three DAA schemes are based on the LRSW problem [31] and DDH problem. The other two DAA schemes were proposed by Chen and Feng [23] and Brickell and Li [15], respectively. Security of these two schemes are based on the q-SDH problem [7] and DDH problem.

We have two contributions in this paper. The first one is a new ECC-DAA scheme, which takes the advantages of multiple existing pairing-based DAA schemes. More specifically, our new scheme is a modification of the last two DAA schemes above. One of the most significant benefits is that the scheme requires very few TPM resources because it places a small computational requirement on a TPM. In fact the TPM has only to perform two exponentiations for the DAA Join protocol and three exponentiations for the DAA Signing protocol. This computational workload is equivalent to a couple of ordinary standard digital signatures, such as EC-DSA or EC-SDSA [28].

We will give some comparison between the proposed scheme and all the existing ECC-DAA schemes, and show that this new scheme has better performance than all the existing schemes. The second contribution is a modification of the DAA security model defined in [12] to cover the property of non-frameability.

The rest of this paper is organized as follows. We first introduce some preliminaries in the next section. We then specify our new DAA scheme in Section 3. For the reason of the limited space, we only describe the security result of our scheme in Section 4, but leave the actual security proof in the full paper [17]. We show some comparison between this scheme and all the existing ECC-DAA schemes in Section 5. We conclude the paper in Section 6.

2 Preliminaries

Throughout the paper, we will use some standard notation as follows. If S is any set then we denote the action of sampling an element from S uniformly at random and assigning the result to the variable x as $x \leftarrow S$. If A is any algorithm then we denote the action of obtaining x by running A on inputs $y_1, \ldots, y_n$ as $x \leftarrow A(y_1, \ldots, y_n)$. We denote concatenation of two date strings x and y as $x \| y$. We write $\{0,1\}^\ell$ for the set of binary strings of length ℓ and $\{0,1\}^*$ for the set of binary strings of arbitrary length.

2.1 Formal Definition and Security Model of DAA

We modify the DAA security model described in [12] by adding the property of non-frameability (as described in [6]) or called exculpability (as described in [1]), i.e., the dishonest issuer and signers together are unable to create a judge-accepted proof that an honest signer produced a certain valid signature σ_0, unless this honest signer really did produce this signature σ_0.

A DAA scheme involves four types of players: a set of DAA issuers $\mathfrak{i}_k \in \mathfrak{I}$, TPM $\mathfrak{m}_i \in \mathfrak{M}$, host $\mathfrak{h}_i \in \mathfrak{H}$ and verifier $\mathfrak{v}_j \in \mathfrak{V}$. The index values, k, i, j, are polynomial. $\mathfrak{m}_i$ and $\mathfrak{h}_i$ form a computer platform in the trusted computing environment and share the role of a DAA signer. Throughout the paper, for the purpose of simplicity, we may omit some of the index values if it does not occur any confusion; for example, we make use of $\mathfrak{i}$ instead of $\mathfrak{i}_k$.

A DAA scheme $\mathcal{DAA} = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Link})$ consists of the following five polynomial-time algorithms and protocols:

- Setup: On input of a security parameter 1^t, an issuer $\mathfrak{i}$ uses this randomized algorithm to produce its secret key isk, public key ipk and the global public parameters par. We will assume that par are publicly known so that we do not need to explicitly provide them as input to other algorithms.
- Join: This protocol, run between a signer $(\mathfrak{m}_i, \mathfrak{h}_i)$ and an issuer $\mathfrak{i}$, consists of two randomized algorithms, Join_t for the TPM and Join_i for the issuer, and it creates the TPM's secret key $\mathtt{tsk}_i$ and its DAA credential cre_i. The value cre_i is given to both $\mathfrak{m}_i$ and $\mathfrak{h}_i$, but the value $\mathtt{tsk}_i$ is known to $\mathfrak{m}_i$ only.
- Sign: On input of $\mathtt{tsk}_i$, cre_i, a basename bsn_j (the name string of $\mathfrak{v}_j$ or a special symbol $\perp$), and a message m to be signed and the verifier's nonce n_V for freshness, $\mathfrak{m}_i$ and $\mathfrak{h}_i$ run this protocol to produce a signature σ.
- Verify: On input of m, bsn_j, a candidate signature σ for m, and a set of rogue signers' secret keys $\mathsf{RogueList}$, $\mathfrak{v}_j$ uses this deterministic algorithm to return either 1 (accept) or 0 (reject). Note that how to build the set of $\mathsf{RogueList}$ is out the scope of the DAA scheme.
- Link: On input of two signatures σ_0 and σ_1, $\mathfrak{v}_j$ uses this deterministic algorithm to return 1 (linked), 0 (unlinked) or $\perp$ (invalid signatures). Note that, unlike Verify, the result of Link is not relied on $\mathsf{RogueList}$.

A DAA scheme must hold the notions of *correctness*, *user-controlled-anonymity* and *user-controlled-traceability*. The first two notions are the same as these defined in [12], so we do not recall them here. The notion of user-controlled-traceability is defined via a game played by a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ as follows:

- *Initial:* There are two initial cases. In Initial Case 1. $\mathcal{C}$ executes $\mathsf{Setup}(1^t)$ and gives the resulting par to $\mathcal{A}$, and $\mathcal{C}$ keeps isk secret. In Initial Case 2. $\mathcal{C}$ receives par from $\mathcal{A}$ and does not know the value of isk.
- *Probing:* $\mathcal{C}$ is probed by $\mathcal{A}$ who makes the following queries:
  - Sign. $\mathcal{A}$ submits a signer's identity ID, a basename bsn (either $\perp$ or a data string) and a message m of his choice to $\mathcal{C}$, who runs Sign to get a signature σ and responds with σ.
  - Semi-sign. $\mathcal{A}$ submits a signer's identity ID along with the data transmitted from $\mathfrak{h}_i$ to $\mathfrak{m}_i$ in Sign of his choice to $\mathcal{C}$, who acts as $\mathfrak{m}_i$ in Sign and responds with the data transmitted from $\mathfrak{m}_i$ to $\mathfrak{h}_i$ in Sign.
  - Join. There are three join cases of this query; the first two are used associated with the Initial Case 1, and the last one is used associated with the Initial Case 2. Suppose that $\mathcal{A}$ does not use a single ID for more than one join case or more than one time.
    - Join Case 1: $\mathcal{A}$ submits a signer's identity ID of his choice to $\mathcal{C}$, who runs Join to create $\mathtt{tsk}$ and cre for the signer, and finally $\mathcal{C}$ sends cre to $\mathcal{A}$ and keeps $\mathtt{tsk}$ secret.
    - Join Case 2: $\mathcal{A}$ submits a signer's identity ID with a $\mathtt{tsk}$ value of his choice to $\mathcal{C}$, who runs Join_i to create cre for the signer and puts the given $\mathtt{tsk}$ into the list of $\mathsf{RogueList}$. $\mathcal{C}$ responds $\mathcal{A}$ with cre.

    - $*$ Join Case 3: $\mathcal{A}$ submits a signer's identity ID of his choice to $\mathcal{C}$, who runs Join_t with $\mathcal{A}$ to create $\mathtt{tsk}$ and to obtain cre from $\mathcal{A}$. $\mathcal{C}$ verifies the validation of cre and keeps $\mathtt{tsk}$ secret.
  - Corrupt. $\mathcal{A}$ submits a signer's identity ID of his choice to $\mathcal{C}$, who responds with the value $\mathtt{tsk}$ of the signer, and puts the revealed $\mathtt{tsk}$ into the list of $\mathsf{RogueList}$.
- *Forge:* $\mathcal{A}$ returns a signer's identity ID, a signature σ, its signed message m and the associated basename bsn. We say that the adversary wins the game if either of the following two situations is true:
  1. With the Initial Case 1 ($\mathcal{A}$ does not have access to isk),
     (a) $\mathsf{Verify}(m, \mathsf{bsn}, \sigma, \mathsf{RogueList}) = 1$ (accepted), but σ is neither a response of the existing Sign queries nor a response of the existing Semi-sign queries (partially); and/or
     (b) In the case of $\mathsf{bsn} \neq \perp$, there exists another signature σ' associated with the same identity and bsn, and the output of $\mathsf{Link}(\sigma, \sigma')$ is 0 (unlinked).
  2. With the Initial Case 2 ($\mathcal{A}$ knows isk), the same as the item (a), in the condition that the secret key $\mathtt{tsk}$ used to create σ was generated in the Join Case 3 (i.e., $\mathcal{A}$ does not have access to $\mathtt{tsk}$).

Definition 1. *Let $\mathcal{A}$ be an adversary that plays the game above. We denote $\mathbf{Adv}[\mathcal{A}^{trace}_{\mathcal{DAA}}] = \mathbf{Pr}[\mathcal{A}\ wins]$ as the advantage that $\mathcal{A}$ breaks the user-controlled-traceability of $\mathcal{DAA}$. We say that a DAA scheme is user-controlled-traceable if for any probabilistic polynomial-time adversary $\mathcal{A}$, the quantity $\mathbf{Adv}[\mathcal{A}^{trace}_{\mathcal{DAA}}]$ is negligible.*

Note that in the above game of the user-controlled-traceability, we allow the adversary to corrupt the issuer. This is an important difference from the game in [12], since it covers the requirement of non-frameability or called exculpabilty.

2.2 Pairings and Relevant Hard Problems

Our new DAA scheme is based on asymmetric pairings. Throughout we let $\mathbb{G}_1 = \langle P \rangle, \mathbb{G}_2 = \langle Q \rangle$ and $\mathbb{G}_T$ be groups of large prime exponent $p \approx 2^t$ for security parameter t. All the three groups will be written multiplicatively. If $\mathbb{G}$ is some group then we use the notation $\mathbb{G}^\times$ to mean the non-identity elements of $\mathbb{G}$.

Definition 2 (Pairing). *A* pairing (or bilinear map) *is a map $\hat{t} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ such that:*

1. *The map $\hat{t}$ is* bilinear. *This means that $\forall P, P' \in \mathbb{G}_1$ and $\forall Q, Q' \in \mathbb{G}_2$ that*
   - $\hat{t}(P \cdot P', Q) = \hat{t}(P, Q) \cdot \hat{t}(P', Q) \in \mathbb{G}_T$.
   - $\hat{t}(P, Q \cdot Q') = \hat{t}(P, Q) \cdot \hat{t}(P, Q') \in \mathbb{G}_T$.
2. *The map $\hat{t}$ is* non-degenerate. *This means that*
   - $\forall P \in \mathbb{G}_1^\times \exists Q \in \mathbb{G}_2$ *such that* $\hat{t}(P, Q) \neq 1_{\mathbb{G}_T} \in \mathbb{G}_T$.
   - $\forall Q \in \mathbb{G}_2^\times \exists P \in \mathbb{G}_1$ *such that* $\hat{t}(P, Q) \neq 1_{\mathbb{G}_T} \in \mathbb{G}_T$.

3. *The map* $\hat{t}$ *is* computable *i.e. there exist some polynomial time algorithm to compute* $\hat{t}(P,Q) \in \mathbb{G}_T$ *for all* $(P,Q) \in \mathbb{G}_1 \times \mathbb{G}_2$.

Our DAA scheme is based on the pairing based signature scheme that is used by Delerablee-Pointcheval [24] as a group membership certificate and also mentioned by Boneh-Boyen-Shacham in [8] as an alternative group member credential. This scheme is given by the following triple of algorithms:

– **KeyGeneration:** Select $P_1, P_2 \leftarrow \mathbb{G}_1$, $Q \leftarrow \mathbb{G}_2$ and $x \leftarrow \mathbb{Z}_p^*$, and compute $X \leftarrow Q^x \in \mathbb{G}_2$. The public key is the tuple (P_1, P_2, Q, X) and the private key is x.
– **Signing:** On input of a message $m \in \mathbb{Z}_p$ the signer chooses $e \leftarrow \mathbb{Z}_p$ and computes $A \leftarrow (P_1 \cdot P_2^m)^{1/(x+e)} \in \mathbb{G}_1$. The signature on m is $\sigma \leftarrow (A, e)$.
– **Verification:** To verify a signature σ on a message m the verifier checks whether $\hat{t}(A, X \cdot Q^e) = \hat{t}(P_1 \cdot P_2^m, Q)$.

The security of the above signature scheme is related to the hardness of the q-SDH problem introduced by Boneh and Boyen [7]. The q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follows:

Definition 3 (q-SDH). *Given a* $(q+2)$*-tuple* $(P, Q, Q^x, Q^{x^2}, ..., Q^{x^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$ *as input output a pair* $(e, P^{1/(x+e)})$ *where* $e \in \mathbb{Z}_p^*$*. An algorithm* $\mathcal{A}$ *has advantage* ϵ *in solving q-SDH in* $(\mathbb{G}_1, \mathbb{G}_2)$ *if*

$$P_r[\mathcal{A}(P, Q, Q^x, Q^{x^2}, ..., Q^{x^q}) = (e, P^{1/(x+e)})] \geq \epsilon,$$

where the probability is over the random choice of generator Q *in* $\mathbb{G}_2$ *(with* $P \leftarrow \psi(Q)$*), of* x *in* $\mathbb{Z}_p^*$ *and of the random bits of* $\mathcal{A}$*. We say that the* (q,t,ϵ)*-SDH assumption holds in* $(\mathbb{G}_1, \mathbb{G}_2)$ *if no t-time algorithm has advantage at least* ϵ *in solving the q-SDH problem in* $(\mathbb{G}_1, \mathbb{G}_2)$.

As the same as in [22], our DAA scheme requires the DDH problem for $\mathbb{G}_1$ to be hard. The formal definition of this problem is defined as follows:

Definition 4 ($\mathbb{G}_1$-DDH). *We define the* $\mathbf{Adv}_{\mathcal{A}}^{\mathrm{DDH}}(t)$ *of an* $\mathbb{G}_1$*-DDH adversary* $\mathcal{A}$ *against the set of parameters* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, Q, p, \hat{t})$ *as*

$$\begin{aligned}&\big|\Pr\big[x,y,z \leftarrow \mathbb{Z}_p; X \leftarrow xP, Y \leftarrow yP, Z \leftarrow zP; \mathcal{A}(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{t}, P, Q, X, Y, Z, p) = 1\big] \\ &-\Pr\big[x,y \leftarrow \mathbb{Z}_p; X \leftarrow xP, Y \leftarrow yP; Z \leftarrow xyP; \mathcal{A}(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{t}, P, Q, X, Y, Z, p) = 1\big]\big|\end{aligned}$$

We then say a tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, Q, p, \hat{t})$ *satisfies the DDH assumption for* $\mathbb{G}_1$ *if for any* p.p.t. *adversary* $\mathcal{A}$ *its advantage* $\mathbf{Adv}_{\mathcal{A}}^{\mathrm{DDH}}(t)$ *is negligible in* t.

Often this problem in the context of pairing groups is called the *external Diffie–Hellman* problem, or the XDH problem.

3 The Proposed DAA Scheme

In this section, we give a detailed description of the new DAA scheme. Before proceeding we note a general point which needs to be born in mind for each of the following protocols and algorithms. Every group element received by any party needs to be checked that it lies in the correct group, and in particular does not lie in some larger group which contains the specified group. This is to avoid numerous attacks such as those related to small subgroups etc (e.g. [9,30]). In asymmetric pairings this is particularly important since $\mathbb{G}_1$ and $\mathbb{G}_2$ can be considered as distinct subgroups of a large group $\mathbb{G}$. If transmitted elements are actually in $\mathbb{G}$, as opposed to $\mathbb{G}_1$ and $\mathbb{G}_2$, then various properties can be broken such as anonymity and linkability.

Hence, our security proofs implicitly assume that all transmitted group elements are indeed elements of the specified groups. For the situation under consideration, namely Type-III pairings [25], efficient methods for checking subgroup membership are given in [18]. Note, we do not count the cost of these subgroup checks in our performance considerations later on, as their relative costs can be quite dependent on the specific groups and security parameters under consideration.

3.1 The Setup Algorithm

On input of the security parameter 1^t, the setup algorithm executes the following:

1. Select three groups, $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$, of sufficiently large prime order p along with a pairing $\hat{t} : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_T$. Select four random generators, P_1, P_2, P_3 and Q, such that $\mathbb{G}_1 = \langle P_1 \rangle = \langle P_2 \rangle = \langle P_3 \rangle$ and $\mathbb{G}_2 = \langle Q \rangle$ and compute $T_1 = \hat{t}(P_1, Q)$, $T_2 = \hat{t}(P_2, Q)$ and $T_3 = \hat{t}(P_3, Q)$. Select five hash functions $H_1 : \{0,1\}^* \mapsto \mathbb{Z}_p$, $H_2 : \{0,1\}^* \mapsto \mathbb{Z}_p$, $H_3 : \{0,1\}^* \mapsto \mathbb{G}_1$, $H_4 : \{0,1\}^* \mapsto \mathbb{Z}_p$ and $H_5 : \{0,1\}^* \mapsto \mathbb{Z}_p$.
2. For each issuer $\mathfrak{i} \in \mathfrak{I}$, select an integer $x \leftarrow \mathbb{Z}_p$ and compute $X = Q^x \in \mathbb{G}_2$ and $T_4 = \hat{t}(P_3, X)$. The issuer secret key isk is assigned to be x and the corresponding public key ipk is assigned to be X.
3. For each TPM $\mathfrak{m} \in \mathfrak{M}$, select a sufficiently large integer $\mathsf{DAAseed}$ at random (e.g. choose $\mathsf{DAAseed}$ from $\{0,1\}^t$) and keep it inside of the TPM secretly.
4. Describe a DAA credential space C, a finite message space M and a finite signature space Σ. The spaces C and Σ will be defined in the Join protocol and Sign protocol respectively. The space M is dependent upon applications.
5. Finally, the system public parameters par are set to be ($\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, p, $\hat{t}$, P_1, P_2, P_3, Q, T_1, T_2, T_3, T_4, H_1, H_2, H_3, H_4, H_5, ipk) together with C, M and Σ, and are published.

The group order p is selected so that solving the $\mathbb{G}_1$-DDH problem or the q-SDH problem takes time 2^t. The three generators P_1, P_2, P_3 are selected so that the discrete logarithm relation between each other, e.g. $\log_{P_1} P_2$, is unknown. Including the four pairing values, T_1, T_2, T_3 and T_4, in par is optional; alternatively these values are computed by hosts and verifiers. To make sure these four values are formed correctly, it is recommended that each host and verifier should compute them once before storing them with other values of par.

We assume that prior to any system setup each TPM has its private endorsement key SK embedded into it and that each issuer has access to the corresponding public endorsement key PK. This key pair is used to build up an authentic channel between the TPM and issuer in the following Join protocol.

3.2 The Join Protocol

This protocol is run between a given TPM $\mathfrak{m} \in \mathfrak{M}$, the corresponding Host $\mathfrak{h} \in \mathfrak{H}$ and an Issuer $\mathfrak{i} \in \mathfrak{I}$. Our protocol proceeds as shown in Figure 1. The communication between $\mathfrak{m}$ and $\mathfrak{i}$ is via $\mathfrak{h}$. This communication must be authentic, that we mean the issuer must be sure that he only creates the DAA credential for a genuine TPM. An authentic channel between the TPM and issuer can be built by using the TPM endorsement key pair $(\mathsf{SK}, \mathsf{PK})$. We suggest using the same mechanism as in the RSA-DAA scheme [10,36] to achieve this. A TPM DAA secret key f is computed from $\mathsf{DAAseed}$ along with a count number cnt and an issuer's public data string K_I.

We note that one of the reasons why our DAA scheme is more efficient than the DAA schemes in [15,23] is that we make use of a simplified construction of the DAA credential cre. In our scheme $\mathsf{cre} = (A, e)$ where $A = (P_1 \cdot P_2^f)^{1/(x+e)}$, but in the both schemes of [15,23], $\mathsf{cre} = (A, e, y)$ where $A = (P_1 \cdot P_2^f \cdot P_3^y)^{1/(x+e)}$ and the value y is contributed by both the TPM and Issuer. Our security proof in the full paper [17] will show that our cre construction is sufficient for achieving required security features of the new scheme.

TPM ($\mathfrak{m}$)		Host ($\mathfrak{h}$)		Issuer ($\mathfrak{i}$)
				$n_I \leftarrow \{0,1\}^t$
$\mathsf{str} \leftarrow X \Vert P_1 \Vert P_2 \Vert P_3 \Vert Q \Vert n_I$	$\xleftarrow{\text{req}}$		$\xleftarrow{\text{req}}$	$\text{req} \leftarrow n_I$
$f \leftarrow H_1(\mathsf{DAAseed} \Vert \mathsf{cnt} \Vert \mathrm{K}_I)$				
$u \leftarrow \mathbb{Z}_p$				
$F \leftarrow P_2^f ; U \leftarrow P_2^u$				
$v \leftarrow H_2(\mathsf{str} \Vert F \Vert U)$				
$w \leftarrow u + f \cdot v \pmod p$				
$\mathsf{comm} \leftarrow (F, v, w, n_I)$	$\xrightarrow{\mathsf{comm}}$		$\xrightarrow{\mathsf{comm}}$	If $n_I \notin \{\text{req}\}$, or
				$U' \leftarrow P_2^w \cdot F^{-v}$
				$\mathsf{str} \leftarrow X \Vert P_1 \Vert P_2 \Vert P_3 \Vert Q \Vert n_I$
				$v \neq H_2(\mathsf{str} \Vert F \Vert U')$
				then **abort**
				$e \leftarrow \mathbb{Z}_p$
				$A \leftarrow (P_1 \cdot F)^{1/(x+e)}$
		If $\hat{t}(A, X \cdot Q^e)$	$\xleftarrow{\mathsf{cre}}$	$\mathsf{cre} \leftarrow (A, e)$
		$\neq \hat{t}(P_1 \cdot F, Q)$		
		then **abort**		

Fig. 1. The Join Protocol

3.3 The Sign Protocol

This protocol is run between a given TPM $\mathfrak{m} \in \mathfrak{M}$ and the corresponding Host $\mathfrak{h} \in \mathfrak{H}$. During the protocol $\mathfrak{m}$ and $\mathfrak{h}$ work together to produce a DAA signature on some message. We note that the Host will know a lot of the values needed in the computation and will be able to take on a lot of the computational workload. However, if the TPM has not had its secret f published (i.e. it is not a rogue module) then the Host $\mathfrak{h}$ will not know f and will be unable to compute the whole signature without the aid of the TPM. Therefore, we say that the TPM is the real signer and the Host is a helper.

TPM ($\mathfrak{m}$)		Host ($\mathfrak{h}$)
		$\mathsf{msg} \in \mathsf{M}$
		If $\mathsf{bsn} = \perp$ then $J \leftarrow \mathbb{G}_1$
		else $J \leftarrow H_3(\mathsf{bsn})$
		Either $n_V \leftarrow \{0,1\}^t$ or receive
		$n_V \in \{0,1\}^t$ from the verifier
		$\mathsf{str} \leftarrow X \Vert P_1 \Vert P_2 \Vert P_3 \Vert Q \Vert n_V$
		$a, r_a, r_e, r_{ae} \leftarrow \mathbb{Z}_p$
		$R \leftarrow A \cdot P_3^a$
		$\hat{S} \leftarrow T^{r_e} \cdot T_3^{a \cdot r_e + r_{ae}} \cdot T_4^{r_a}$
$r_f \leftarrow \mathbb{Z}_p$	$\xleftarrow{h,\, J,\, \hat{S},\, \mathsf{msg}}$	$h \leftarrow H_4(\mathsf{str} \Vert R)$
$K \leftarrow J^f;\, L \leftarrow J^{r_f}$		
$S \leftarrow \hat{S} \cdot T_2^{r_f}$		
$n_T \leftarrow \{0,1\}^t$		
$c \leftarrow H_5(h \Vert J \Vert K \Vert L \Vert S \Vert \mathsf{msg} \Vert n_T)$		
$s_f \leftarrow r_f + f \cdot c \pmod p$	$\xrightarrow{(K,\, c,\, s_f,\, n_T)}$	$s_a = r_a + a \cdot c \pmod p$
		$s_e = r_e - e \cdot c \pmod p$
		$s_{ae} = r_{ae} + a \cdot e \cdot c \pmod p$
		$\sigma \leftarrow (R, J, K, c, s_f, s_a, s_e, s_{ae})$

Fig. 2. The Sign Protocol

The protocol then proceeds as in Figure 2. We note that in this scheme, the Host $\mathfrak{h}$ precomputes $T = \hat{t}(A, Q)$ and stores it as a long-term parameter. In Table 1 of Section 5, we do not list this pairing computation in the signing computational cost since it will only be computed once. We also note that the following equation holds:

$$\begin{aligned} S &= \hat{S} \cdot T_2^{r_f} \\ &= T^{r_e} \cdot T_3^{a \cdot r_e + r_{ae}} \cdot T_4^{r_a} \cdot T_2^{r_f} \\ &= \hat{t}(R, Q^{s_e} \cdot X^{-c}) \cdot T_2^{s_f} \cdot T_4^{s_a} \cdot T_3^{s_{ae}} \cdot T_1^{c}. \end{aligned}$$

3.4 The Verification Algorithm

This algorithm is run by a verifier $\mathfrak{v}$. On input of a signature $\sigma = (R, J, K, c, s_f, s_a, s_e, s_{ae})$, two nonces (n_V, n_T), a message msg, a basename bsn, an issuer public key $\mathsf{ipk} = X$ and the public system parameters par, this algorithm performs the following steps:

1. *Check Against* $\mathsf{RogueList}$. If $K = J^{f_i}$ for any f_i in the set of rogue secret keys then return *reject*.
2. *Check J computation.* If $\mathsf{bsn} \neq \perp$ and $J \neq H_3(\mathsf{bsn})$ then return *reject*.
3. *Verify Correctness of Proofs.* This is done by performing the following sets of computations:
   - $S' \leftarrow \hat{t}(R, Q^{s_e} \cdot X^{-c}) \cdot T_2^{s_f} \cdot T_4^{s_a} \cdot T_3^{s_{ae}} \cdot T_1^{c}$.
   - $L' \leftarrow J^{s_f} \cdot K^{-c}$.
   - $\mathsf{str} \leftarrow X \| P_1 \| P_2 \| P_3 \| Q \| n_V$.
   - $h' \leftarrow H_4(\mathsf{str} \| R)$.

   If $c \neq H_5\,(h' \| J \| K \| L' \| S' \| \mathsf{msg} \| n_T)$ return *reject* and otherwise return *accept*.

3.5 The Linking Algorithm

This algorithm is run by a given verifier $\mathfrak{v}_j \in \mathfrak{V}$ which has a set of basenames $\{\mathsf{bsn}\}_j$ in order to determine if a pair of signatures were produced by the same TPM. This algorithm is the same as in the DAA schemes of [12,22]. Signatures can only be linked if they were produced by the same TPM and the user wanted them to be able to be linked together. Formally, on input a tuple $((\sigma_0, \mathsf{msg}_0), (\sigma_1, \mathsf{msg}_1), \mathsf{bsn}, \mathsf{ipk})$ the algorithm performs the following steps:

1. *Verify Both Signatures.* For each signature σ_b, for $b \in \{0, 1\}$ the verifier runs the algorithm $\mathsf{Verify}(\sigma_b, \mathsf{msg}_b, \mathsf{bsn}, \mathsf{ipk})$ and if either of these returns *reject* then the value $\perp$ is returned.
2. *Compare J and K values.* If $J_0 = J_1$ and $K_0 = K_1$ then return *linked*, else return *unlinked*.

3.6 Revocation Consideration

In the literature, there are four types of revocation solutions are known for DAA. The first two solutions were proposed in the original DAA paper [10]: (1) anybody can verify whether a given DAA signature was signed under a key listed in the rogue list $\mathsf{RogueList}$, and (2) without $\mathsf{RogueList}$, a verifier can build his own black list of unwelcome signers and reject these signer's signatures by using a basename. The third was proposed by Brickell and Li in [14]: a signer can prove that he is not anyone listed in the black list. The last one was proposed by Chen, Morrissey and Smart in [22]: the issuer can update his public key and each legitimate singer's credential efficiently, and this process is transparent to a TPM. Our new DAA scheme is suitable for all of these revocation solutions. Choice of them is dependent upon applications. This feature has no difference from the existing DAA schemes.

4 Security of the DAA Scheme

In this section, we will state the security results for the new DAA scheme under the definitions of security notions in Section 2.1. In general, we will argue that our new DAA scheme is secure, i.e., correct, user-controlled-anonymous and user-controlled-traceable, as addressed in the following theorems.

Our security results are based on the q-SDH assumption and the $\mathbb{G}_1$-DDH assumption as defined in Section 2.2. The security analysis of the notions of user-controlled-anonymity and user-controlled-traceability is in the random oracle model [5], i.e., we will assume that the hash functions H_2 H_3 and H_5 in the new DAA scheme are random oracles. Note that the hash function H_1 used to compute the value f and H_4 used in the Sign protocol do not have to be random oracles, since they are internal functions.

Theorem 1. *The DAA scheme specified in Section 3 is correct.*

Proof. This theorem follows directly from the specification of the scheme. □

Theorem 2. *Under the $\mathbb{G}_1$-DDH assumption in Definition 4, the above DAA scheme is user-controlled-anonymous. More specifically, if there is an adversary $\mathcal{A}$ that succeeds with a non-negligible probability to break user-controlled-anonymity of the scheme, then there is a simulator $\mathcal{S}$ running in polynomial time that solves the $\mathbb{G}_1$-DDH problem with a non-negligible probability.*

The proof of this theorem is given in the full paper [17].

Theorem 3. *Under the q-SDH assumption, the above DAA scheme is user-controlled-traceable. More specifically, if there is an adversary $\mathcal{A}$ that succeeds with a non-negligible probability to break user-controlled-traceability of the scheme, then there is a simulator $\mathcal{S}$ running in polynomial time that solves the q-SDH problem with a non-negligible probability.*

Again, the proof of this theorem is given in the full paper [17].

5 Performance Comparison

In this section, we compare efficiency of the proposed DAA scheme with all the existing ECC-DAA schemes, based on our best knowledge. We do not include RSA-DAA schemes such as these in [10,26], since the comparison between the RSA-based schemes and pairing-based schemes has been presented in a number of papers. In general speaking, we see that the ECC-DAA schemes are a lot more efficient than the one based on factoring. We refer to [12,20,22,23] for the detailed information.

In Table 1, we present some performance figures for the six ECC-DAA schemes. For the computational cost, we consider the Join protocol, Sign protocol and Verify algorithm, with respect to each player. We do not specify the computational cost of the Setup algorithm and its verification, since this is only run once and the resulting parameters are only verified once by each part. We do not specify the cost for the linking algorithm either, as it is closely related to that of the verification

Table 1. Cost Comparison of the Six Pairing-Based DAA Protocols

Operation	Party	Computational Cost	Credential Size	Signature Size
Scheme of [12]				
Join	TPM	$3\mathbb{G}_1$		
	Issuer	$2\mathbb{G}_1 + 2\mathbb{G}_1^2$		
	Host	$6P$	$3\mathbb{G}_1$	$2p + 3\mathbb{G}_1 + 2\mathsf{G} + 1h$
Sign	TPM	$3\mathbb{G}_T$		
	Host	$3\mathbb{G}_1 + 1\mathbb{G}_T + 3P$		
Verify	Verifier	$1\mathbb{G}_T^2 + 1\mathbb{G}_T^3 + 5P + (n+1)\mathbb{G}_T$		
Scheme of [20]				
Join	TPM	$3\mathbb{G}_1$		
	Issuer	$2\mathbb{G}_1 + 2\mathbb{G}_1^2$		
	Host	$6P$	$3\mathbb{G}_1$	$1p + 4\mathbb{G}_1 + 1h$
Sign	TPM	$1\mathbb{G}_1$		
	Host	$4\mathbb{G}_1 + 3\mathbb{G}_T + 1P$		
Verify	Verifier	$1\mathbb{G}_1^2 + 1\mathbb{G}_T^2 + 5P + n\mathbb{G}_1$		
Scheme of [22]				
Join	TPM	$3\mathbb{G}_1$		
	Issuer	$2\mathbb{G}_1 + 2\mathbb{G}_1^2$		
	Host	$4P$	$3\mathbb{G}_1$	$1p + 5\mathbb{G}_1 + 1h$
Sign	TPM	$2\mathbb{G}_1 + 1\mathbb{G}_T$		
	Host	$3\mathbb{G}_1 + 1P$		
Verify	Verifier	$1\mathbb{G}_1^2 + 1\mathbb{G}_T^2 + 5P + n\mathbb{G}_1$		
Scheme of [23]				
Join	TPM	$3\mathbb{G}_1^2 + (2P)$		
	Issuer	$1\mathbb{G}_1^2 + 1\mathbb{G}_1^3$		
	Host	$(2P)$	$2q + 1\mathbb{G}_1$	$6p + 2\mathbb{G}_1 + 2\mathsf{G} + 1h$
Sign	TPM	$2\mathbb{G}_1 + 1\mathbb{G}_T^2$		
	Host	$1\mathbb{G}_1 + 2\mathbb{G}_1^2 + 1\mathbb{G}_1^3 + 1\mathbb{G}_T^3$		
Verify	Verifier	$1\mathbb{G}_1^2 + 2\mathbb{G}_1^3 + 1\mathbb{G}_T^5 + 3P + n\mathbb{G}_T$		
Scheme of [15]				
Join	TPM/Host	$2\mathbb{G}_1^2 + 1\mathbb{G}_2 + 2P + proof$		
	Issuer	$1\mathbb{G}_1^2 + verify$	$2q + 1\mathbb{G}_1$	$4p + 1\mathbb{G}_1 + 2\mathsf{G} + 1h$
Sign	TPM/Host	$1\mathbb{G}_1 + 2\mathbb{G}_T + 1\mathbb{G}_T^4$		
Verify	Verifier	$1\mathbb{G}_T^2 + 1\mathbb{G}_T^5 + 2P + n\mathbb{G}_T$		
Scheme of this paper				
Join	TPM	$2\mathbb{G}_1$		
	Issuer	$1\mathbb{G}_1 + 1\mathbb{G}_1^2$		
	Host	$1\mathbb{G}_1 + 2P$	$1q + 1\mathbb{G}_1$	$4p + 3\mathbb{G}_1 + 1h$
Sign	TPM	$2\mathbb{G}_1 + 1\mathbb{G}_T$		
	Host	$1\mathbb{G}_1 + 1\mathbb{G}_T^3$		
Verify	Verifier	$1\mathbb{G}_1^2 + 1\mathbb{G}_2^2 + 1\mathbb{G}_T^4 + 1P + n\mathbb{G}_1$		

algorithm. For the communication cost and storage cost, we consider the credential size and signature size, but ignore the size of TPM secret key (i.e. the values of f and $\mathsf{DAAseed}$) because this could be the same in all of the six schemes.

In this table, for the computational cost, we let $\mathbb{G}_i$ $(i = \{1, 2, T\})$ denote the cost of an exponentiation in the group $\mathbb{G}_i$, and $\mathbb{G}_i^m$ denote the cost of a multi-exponentiation of m values in the group $\mathbb{G}_i$. Note, that a multiexponentiation with m exponents can often be performed significantly faster than m separate exponentiations, which is why we separate this out. We also let P denote the cost of a pairing computation. In addition in the table we let n denote the number of keys in the verifier's rogue secret key list. For the credential and signature sizes, we let p denote the size of the prime order p, $\mathbb{G}_i$ $(i = \{1, 2, T\})$ denote the size of an element of the group $\mathbb{G}_i$, G denote the size of an element of the group G, which is used in [12,15,23] as a group that might be separated from $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, and h denote an output of a hash-function used in the Schnorr-type signature schemes [34]. The Brickell et al. scheme [12] uses symmetric pairings $\hat{t} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_T$, and the other five schemes in [15,20,22,23] and this paper use asymmetric pairings $\hat{t} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$.

Observe that in [23], the rogue ragging operation is not defined in the Verify algorithm, but it can be easily added in the same way as every existing DAA scheme does. So in Table 1, we add this computation $n \cdot \mathbb{G}_T$. Another observation of this scheme is that the pairing computation in the Join protocol can be done by the Host instead of the TPM, because it is expensive to implement the pairing operation in TPMs. We mark this change as $(2P)$ in Table 1. Observe also that in the scheme in [15], the signer is a single entity, but it is not difficult to split the signer role between a TPM and a Host. So in the following comparison, we assume that these changes have been taken into account.

Our proposed DAA scheme has the most efficient computational cost in the Join protocol, that includes not only the computational cost of the TPM but also the computational cost of the whole signer (the TPM and Host) and the computational cost of the issuer. In the Sign protocol, the most efficient scheme regarding the TPM's operation is the scheme in [20]. However, this scheme is not secure as the attacks demonstrated in [19,22]. Especially in the important operation of signing by the TPM, our scheme is the same as the scheme in [22], which is a repair of the scheme in [20] and these two schemes are more efficient than the other three schemes [12,15,23]. Note that in the scheme in [22], the pairing computation of the Host in the Sign protocol can be replaced by an exponentiation in $\mathbb{G}_T$ with the precompution on $\hat{t}(B, X)$. In the Verification algorithm, our scheme has the same computation cost of the verifier as in [15], and they are more efficient than all of the other schemes in [12,20,22,23].

Except the efficiency in the computational cost, the attractive performance of our scheme is that it has the smallest credential size and signature size, comparing with the other schemes in the table. This comparison is based on the fact that the size of G is various, and to our best knowledge it can be chosen between the sizes of $\mathbb{G}_1$ and $\mathbb{G}_T$. In summary, our scheme is the most efficient DAA scheme so far with the acceptable security level.

6 Conclusions

Recently TCG TPM working group has been working on the next generation of TPM. One of the most interesting features is that the new TPM supports algorithm agility. If a TPM supports multiple DAA algorithms, such as both RSA-DAA and ECC-DAA, we can make use of DAAseed as a master secret and create multiple DAA secret f values from it. In that case, we do not require extra internal storage for long-term secrets. Our conclusion is that DAA is algorithm agile and our ECC-DAA implementation is fairly efficient.

Acknowledgements

The author would like to thank Jiangtao Li for his invaluable discussion and comments on this work, and also to thank the anonymous reviewers of Inscrypt'09 for their thoughtful comments, which are very helpful to improve the paper.

References

1. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k-TAA. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 111–125. Springer, Heidelberg (2006)
2. Backes, M., Maffei, M., Unruh, D.: Zero knowledge in the applied Pi–calculus and automated verification of the direct anonymous attestation protocol. In: IEEE Symposium on Security and Privacy – SSP 2008, pp. 202–215 (2008)
3. Balfe, S., Lakhani, A.D., Paterson, K.G.: Securing peer-to-peer networks using trusted computing. In: Mitchell (ed.) Trusted Computing, ch. 10, pp. 271–298. IEEE, London (2005)
4. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)
5. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: The 1st ACM Conference on Computer and Communications Security, pp. 62–73. ACM Press, New York (1993)
6. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)
7. Boneh, D., Boyen, X.: Sort signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
8. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
9. Boyd, C., Pavlovski, C.: Attacking and repairing batch verification schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 58–71. Springer, Heidelberg (2000)
10. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: The 11th ACM Conference on Computer and Communications Security, pp. 132–145. ACM Press, New York (2004)

11. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation in context. In: Mitchell (ed.) Trusted Computing, ch. 5, pp. 143–174. IEEE, London (2005)
12. Brickell, E., Chen, L., Li, J.: Simplified security notions for direct anonymous attestation and a concrete scheme from pairings. Int. Journal of Information Security 8, 315–330 (2009)
13. Brickell, E., Chen, L., Li, J.: A new direct anonymous attestation scheme from bilinear maps. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) Trust 2008. LNCS, vol. 4968, pp. 166–178. Springer, Heidelberg (2008)
14. Brickell, E., Li, J.: Enhanced privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. In: The 6th ACM Workshop on Privacy in the Electronic Society – WPES 2007, pp. 21–30. ACM Press, New York (2007)
15. Brickell, E., Li, J.: Enhanced privacy ID from bilinear pairing. In: Cryptology ePrint Archive. Report 2009/095, http://eprint.iacr.org/2009/095
16. Canard, S., Traore, J.: List signature schemes and application to electronic voting. Presented in International Workshop on Coding and Cryptography 2003 (2003); See also the Journal Version of This Paper by Canard, S., Schoenmakers, B., Stam, M., Traore, J.: List signature schemes. Discrete Applied Mathematics 154(2), 189–201 (2006)
17. Chen, L.: A DAA scheme requiring less TPM resources. In: Cryptology ePrint Archive. Report 2010/008, http://eprint.iacr.org/2010/008
18. Chen, L., Cheng, Z., Smart, N.P.: Identity-based key agreement protocols from pairings. Int. Journal of Information Security 6, 213–242 (2007)
19. Chen, L., Li, J.: A note on the Chen-Morrissey-Smart Direct Anonymous Attestation scheme (preprint)
20. Chen, L., Morrissey, P., Smart, N.P.: Pairings in trusted computing. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 1–17. Springer, Heidelberg (2008)
21. Chen, L., Morrissey, P., Smart, N.P.: On proofs of security of DAA schemes. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) ProvSec 2008. LNCS, vol. 5324, pp. 156–175. Springer, Heidelberg (2008)
22. Chen, L., Morrissey, P., Smart, N.P.: DAA: Fixing the pairing based protocols. In: Cryptology ePrint Archive. Report 2009/198, http://eprint.iacr.org/2009/198
23. Chen, X., Feng, D.: Direct anonymous attestation for next generation TPM. Journal of Computers 3(12), 43–50 (2008)
24. Delerablee, C., Pointcheval, D.: Dynamic fully anonymous short group signatures. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 193–210. Springer, Heidelberg (2006), http://www.di.ens.fr/users/pointche/Documents/Papers/2006_vietcrypt.pdf for a corrected version of this paper
25. Galbraith, S., Paterson, K., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156, 3113–3121 (2008)
26. Ge, H., Tate, S.R.: A Direct anonymous attestation scheme for embedded devices. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 16–30. Springer, Heidelberg (2007)
27. ISO/IEC 11889:2009 Information technology – Security techniques – Trusted Platform Module
28. ISO/IEC 14888-3 Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms
29. Leung, A., Chen, L., Mitchell, C.J.: On a possible privacy flaw in direct anonymous attestation (DAA). In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) Trust 2008. LNCS, vol. 4968, pp. 179–190. Springer, Heidelberg (2008)

30. Lim, C.H., Lee, P.J.: A key recovery attack on discrete log-based schemes using a prime order subgroup. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 249–263. Springer, Heidelberg (1997)
31. Lysyanskaya, A., Rivest, R., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)
32. Pashalidis, A., Mitchell, C.J.: Single sign-on using TCG-conformant platforms. In: Mitchell (ed.) Trusted Computing, ch. 6, pp. 175–193. IEEE, London (2005)
33. Rudolph, C.: Covert identity information in direct anonymous attestation (DAA). In: The 22nd IFIP TC-11 International Information Security Conference – SEC 2007 (2007)
34. 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)
35. Smyth, B., Chen, L., Ryan, M.: Direct Anonymous Attestation (DAA): Ensuring privacy with corrupt administrators. In: Stajano, F., Meadows, C., Capkun, S., Moore, T. (eds.) ESAS 2007. LNCS, vol. 4572, pp. 218–231. Springer, Heidelberg (2007)
36. Trusted Computing Group. TCG TPM specification 1.2 (2003), http://www.trustedcomputinggroup.org

Full-Custom VLSI Design of a Unified Multiplier for Elliptic Curve Cryptography on RFID Tags

Johann Großschädl[1,2]

[1] University of Luxembourg,
Laboratory of Algorithmics, Cryptology and Security (LACS),
6, rue Richard Coudenhove-Kalergi, L–1359 Luxembourg, Luxembourg
johann.groszschaedl@uni.lu
[2] University of Bristol,
Department of Computer Science,
Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, U.K.
johann.groszschaedl@cs.bris.ac.uk

Abstract. The question of whether elliptic curve cryptography (ECC) can be implemented efficiently enough to meet the strict power and area constraints of passive RFID tags has received considerable attention in recent years. While numerous algorithmic and architectural approaches for reducing the footprint of ECC hardware have been investigated, the potential of full-custom VLSI design is still completely unexplored. In this paper we present the design of a radix-2 and a radix-4 version of a unified (16×16)-bit multiplier with a 40-bit accumulator that provides all the arithmetic functionality needed to perform ECC over prime and binary fields. The term "unified" means that our multiply/accumulate (MAC) unit uses the same datapath for the multiplication of integers as well as binary polynomials. We designed a full-custom layout of both the radix-2 and the radix-4 multiplier on basis of a conventional array architecture. Simulation of netlists showed a power saving of 22% and an energy-delay advantage of 48% for the radix-4 multiplier compared to the radix-2 version. The multiplication of binary polynomials consumes about 39% less power than integer multiplication.

1 Introduction

Radio-Frequency Identification (RFID) is a technology that uses radio waves to determine the identity of objects or subjects. Originally developed as a "barcode replacement," RFID technology has found widespread adoption in supply chain management, transportation, and logistics. An RFID system consists of three basic building blocks: RFID tags, RFID readers, and a backend server. Each tag contains a tiny micro-chip (in which a unique ID is stored) and an RF antenna for communication with the reader. Passive RFID tags have no internal power supply but obtain their operating power from a magnetic field generated by the reader through inductive coupling. The reader interacts with the tags and passes their responses to the backend server, which can retrieve detailed information about the tagged objects or subjects by querying its database.

The growth and proliferation of RFID technology in recent years has raised a number of security and privacy concerns [20]. As stated in [36], unprotected RFID tags may have vulnerabilities to eavesdropping, traffic analysis, spoofing or denial of service, whereby the concrete threat scenario depends on the application. For example, the main threat to RFID-based anti-counterfeiting systems is the spoofing (or "cloning") of tags, which can be prevented by securing the tags' identifying information against unauthorized access. A privacy concern in the realm of RFID is the surreptitious tracking of tags through time and space [36]. These concerns have triggered a large body of research on lightweight cryptosystems and security protocols, taking the specific constraints and adversary models of RFID into account. A good example of a lightweight cryptosystem is the block cipher PRESENT [5], which can be implemented in hardware with as few as 1000 gates [31]. However, RFID authentication schemes based on secret-key algorithms suffer from poor scalability due to the key search problem [2]. In addition, they require the distribution of keys across organizational boundaries (e.g. from a manufacturer to a retailer), which may be non-trivial to realize in practice. A third problem relates to the privacy of these schemes. For example it was shown in [35] that narrow-strong privacy in an RFID system can only be achieved with tags being able to perform public-key cryptography.

Among the public-key techniques that have been examined regarding their suitability for securing RFID are elliptic and hyperelliptic curve cryptography (ECC, HECC) [4,11], NTRU [21], WIPR [29], and GPS [30]. However, ECC has the advantage of being promoted by various standardization bodies such as the NIST. Wolkerstorfer [38] was the first to demonstrate that ECC (or, more precisely, ECDSA signature generation) is feasible for passive RFID tags in terms of silicon area and power consumption. The design of low-cost ECC hardware has progressed considerably during the past five years, thanks to innovation on both the algorithmic and architectural level. An example for the former is Lee et al's common Z-coordinate technique for scalar multiplication, which reduces the storage requirements and, hence, size of ECC hardware [24]. One of the architectural innovations that led to a reduction of silicon area is the combination of different arithmetic operations into a single datapath; examples include the integration of addition into a multiplier datapath for binary polynomials such as proposed in [32, Chapter 6.2] (and previously in [14]), as well as the unified multiplier/inverter datapath from [10]. The majority of low-cost ECC hardware implementations reported in the literature are based on a binary field due to the "carry-free" arithmetic and efficiency of the squaring operation. However, it was shown in [12] that ECC hardware using a prime field as underlying algebraic structure is also suitable for RFID tags. Wolkerstorfer's ECC implementation [38] supports both prime and binary fields.

In 2008, Hein et al [19] introduced a completely new architecture for small-footprint ECC hardware that advanced the state-of-the-art in several ways. The ECCon Processor described in [19] uses a (16×16)-bit multiplier (instead of a bit-serial or digit-serial multiplier) to perform the field arithmetic—a distinctive feature that makes ECCon radically different from other ECC implementations

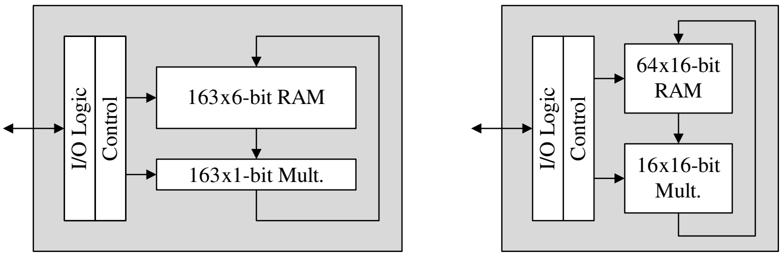


Fig. 1. Bit-serial architecture vs. word-level architecture (source: [19])

for RFID tags. Figure 1 illustrates a classical architecture based on a bit-serial multiplier for the binary field GF(2^{163}) and Hein et al's word-level architecture using a (16×16)-bit multiplier for binary polynomials. The latter operates on 16-bit words ("digits") at a time and performs a multiplication of two elements of GF(2^m) via well-known algorithms for multiple-precision arithmetic such as the product scanning method [18]. Also the reduction of a $2m$-bit product modulo an irreducible polynomial as well as the inversion of an element of GF(2^m) can be efficiently realized through multiplications of 16-bit words [16,22]. The idea of splitting up binary-field arithmetic into operations on small words was first described in the context of instruction set extensions for general-purpose processors [16]. Hein et al applied this concept to RFID tags by replacing the processor by a hardwired finite-state machine in order to save silicon area. The area requirements of a (16×16)-bit multiplier are very similar to that of a bit-serial multiplier for GF(2^{163}). However, the big advantage of the word-level architecture is low power consumption in combination with a flat (and nearly constant) power profile suitable for passive RFID tags. Bit-serial multipliers, in contrast, require the distribution of certain signals (e.g. the operand bit a_i [14]) over the full datapath, resulting in long wires with high capacitive load that can cause significant power peaks.

Besides algorithmic and architectural optimizations, there is a third avenue for reducing area and power consumption of ECC hardware, namely the VLSI design methodology. Applying a *Full-Custom Design Flow* allows a designer to hand-craft all circuits (using, for example, a special logic style) and optimize transistor sizes [37]. Dally and Chang demonstrated through a number of case studies that employing a full-custom design methodology can yield substantial savings in area (up to a factor of 14.5 [8]) and power consumption (between a factor of 3 and 10 [7]) compared to a standard-cell based design. However, while algorithmic and architectural optimizations have received a lot of attention in recent years, the potential of full-custom design for the implementation of low-footprint ECC hardware is yet to be explored. In the present paper we take a first step in this direction by introducing a full-custom VLSI design of a unified (16×16)-bit multiplier with a 40-bit accumulator. The term "unified" in the context of our multiplier (or, more precisely, multiply/accumulare unit) refers to its ability to process integers and binary polynomials [34]. Consequently, the multiply/accumulate (MAC) unit provides all the functionality needed for the implementation of ECC over both prime and binary fields. In the latter case, a unified multiplier or MAC unit allows for performing a full ECDSA signature generation[1], whereas Hein et al's ECCon processor [19] can just accomplish a scalar multiplication. We actually implemented and analyzed two versions of a unified $(16 \times 16 + 40)$-bit MAC unit; the first uses a conventional radix-2 representation of operands, while the second variant is based on the radix-4 design from [17] but adapted for full-custom implementation. Optimized for small area and low power dissipation, the unified MAC unit(s) can be used as functional unit of a general-purpose processor [17] or as arithmetic core of a cryptographic co-processor [33]. The MAC unit is particularly suited for the implementation of low-footprint ECC hardware following Hein et al's approach [19].

2 Preliminaries and Basic Design Decisions

Design of Unified Multipliers. The elements of a prime field GF(p) are the residue class modulo p (i.e. the integers $0, \ldots, p-1$), whereas the elements of a binary extension field GF(2^m) are commonly represented by binary polynomials of degree up to $m-1$. However, there exist also some structural similarities between prime and binary fields. First, the elements of either type of field can be represented by a bit-string. Second, the multiplication in both GF(p) and GF(2^m) implies a modular reduction. Third, the multiplication of both integers and binary polynomials can be performed by means of generation and addition of partial products [1]. These similarities facilitate the design of a unified multiplier, which is simply a multiplier that uses the same datapath for integers and binary polynomials [33,34]. Having a unified multiplier instead of two separate multipliers can result in considerable savings in area when an application needs to deal with both types of operand. For example, the silicon area of the unified multiplier introduced in [34] is only slightly larger than the area of a standard (i.e. integer-only) multiplier.

When classifying previous work on unified multipliers, three principal design approaches can be identified. The first approach, introduced by Garcia et al in [13], uses a special wiring methodology to integrate the multiplication of binary polynomials into a tree multiplier for integers. More precisely, the architecture presented in [13] (and also the one in [33]) implements two separate sub-trees for the addition of sum and carry vectors. Another approach for the unification of integer and polynomial arithmetic is the integration of some extra logic into half and full adders such that all carry bits of the multiplier datapath can be suppressed (i.e. set to zero) [9,34]. Finally, it is also possible to design a unified multiplier on basis of a redundant binary representation as shown in [1].

[1] Computing an ECDSA signature requires, besides the field arithmetic (i.e. polynomial arithmetic when using a binary field), also integer arithmetic (addition and multiplication) modulo the order of the base point [18]. Lee et al [24] integrated a dedicated 8-bit microcontroller into their RFID security processor to execute these integer arithmetic operations in a byte-wise fashion.

Array versus Tree. Array multipliers are known to provide high regularity as well as locality at the silicon level for the expense of a critical path that scales linearly with the word-size. On the other hand, tree multipliers have an irregular structure but can work at much higher frequencies due to a significantly shorter (i.e. logarithmic-length) critical path. While the speed characteristics of array and tree multipliers are obvious and well understood [37], the situation is not so clear regarding power consumption.

At a first glance, one is tempted to attribute array multipliers a rather high power consumption because of the long signal paths that may cause lots of gate-output transitions to propagate through the array. It was shown in [3] and [6] that array multipliers consume significantly more power than tree multipliers when the impact of wiring is completely ignored. However, since CMOS process technology improves and transistor geometries become smaller and smaller, the parasitic capacitances of interconnect wires dominate over the transistor capacitances. Tree multipliers suffer from long interconnect wires with high capacitive load, which is due to their highly irregular structure. In addition, signal paths of varying lengths lead to signal skew and an increased number of glitches and spurious transitions. Meier et al [26] demonstrated that wiring has a big impact on the power consumption of tree multipliers, and that the difference between array and tree multipliers is very small when wiring effects are considered.

An array-like architecture is definitely to prefer over a tree-like architecture when one employs a full-custom design methodology instead of logic synthesis with automatic place and route. It is, of course, also possible to produce a full-custom layout of a tree multiplier, but this is a laborious task because of the irregular structure. Array architectures feature a high degree of regularity and mainly local interconnect, which makes them easy to design and lay out. Taking these aspects into account, we opted for the array architecture.

Full-Custom Cell Library. Bisdounis et al [3] compared eight circuit techniques to evaluate their suitability for the implementation of low-power adders and multipliers. Complementary static CMOS logic showed good results in this comparison and belonged to the "Top 3" logic styles with respect to power consumption. Furthermore, static CMOS logic is fast, easy to design, immune to noise, and robust against voltage scaling and transistor down-sizing. All these desirable properties make static CMOS the circuit technique of choice for the implementation of a low-power/small-area multiplier [39].

We developed a small full-custom cell library containing all standard logic gates with two and three inputs [37]. Moreover, our cell library also includes a two-input XOR and XNOR designed on basis of the classical transmission-gate XOR/XNOR circuit [23, Figure 4] with a tailing inverter to increase the drive strength. Transmission gates were also employed for the implementation of an inverting 2:1 multiplexer. The XOR and XNOR gate consist of eight transistors each, while the multiplexer has a transistor count of six. We used Mentor Graphics' GDT tools (i.e. the Led editor, Lmask, and Lsim Power Analyst) for the design, parasitics extraction, and simulation of our full-custom cell library.

3 Unified Radix-2 Multiplier

A hardware multiplier computes the product of two integers (resp. polynomials) through generation and addition of partial products. In the following, we detail this process for a radix-2 (resp. radix-t) representation of operands.

3.1 Generation of Partial Products

Formally, a multiplication of two unsigned 16-bit integers A, B is carried out by generation and addition of partial products as follows.

$$Z = A \cdot B = A \cdot \sum_{i=0}^{15} b_i \cdot 2^i = \sum_{i=0}^{15} A \cdot b_i \cdot 2^i \tag{1}$$

In the binary case (i.e. radix-2 representation), the generation of a partial product $A \cdot b_i$ is simply a logical AND operation between the multiplier bit b_i and the 16 bits of the multiplicand A. These partial products have to be summed up according to their weight 2^i (i.e. in the appropriate relative position) to obtain the correct result $Z = A \cdot B$ [37]. The multiplication of binary polynomials can employ the same technique of generation and addition of partial products. For example, a multiplication of two binary polynomials $A(t), B(t)$ of degree 15 is performed as follows.

$$Z(t) = A(t) \cdot B(t) = A(t) \cdot \sum_{i=0}^{15} b_i \cdot t^i = \sum_{i=0}^{15} A(t) \cdot b_i \cdot t^i \tag{2}$$

All coefficients b_i of $B(t)$ are from $\mathrm{GF}(2) = \{0, 1\}$ when using a conventional (i.e. radix-t) representation; thus, the generation of a partial product $A(t) \cdot b_i$ is simply a logical AND operation between b_i and the coefficients of $A(t)$ [27]. The multiplication of a partial product $A(t) \cdot b_i$ by a power of the form t^i is nothing else than a left-shift by i bit positions ("Shift-and-XOR" technique). A partial product generator (PPG) for a unified radix-2/radix-t multiplier consists of a row of AND gates [9,33]. In other words, we can use exactly the same hardware (i.e. AND gates) to generate partial products for both integer and polynomial multiplication.

3.2 Addition of Partial Products

A conventional implementation of an array multiplier is built of rows of (3,2) counters (full adders) and (2,2) counters (half adders) to compress the partial products to a single sum and carry vector. The addition of these two vectors is the final step for completing the multiplication and calls for an efficient carry-propagation adder (generally referred to as "final adder"). In the radix-2 case, a typical $(w \times w)$-bit array multiplier consists of w^2 AND gates, $w - 1$ half and $(w - 1) \cdot (w - 2)$ full adder cells, respectively, and a fast w-bit carry-propagation adder for redundant-to-binary conversion of the upper part of the result [37].

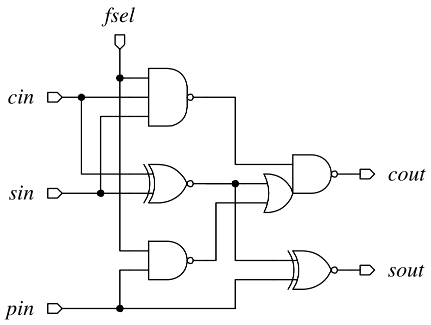


Fig. 2. Dual-field adder (DFA)

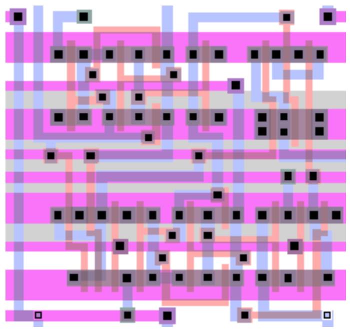


Fig. 3. DFA full-custom layout

As outlined in Section 2, there are three different approaches for combining integer and polynomial multiplication into a single datapath. The first method (i.e. the special wiring technique described in [13]) is applicable to tree multipliers, but would completely destroy the regularity of array multipliers. On the other hand, using a redundant binary representation—as proposed in [1]—has advantages for high-speed designs, but causes significant power consumption in polynomial mode [17]. Therefore, we decided to implement our multiplier using full adders with the ability to suppress the carry bit, as this approach complies best with our requirements for a regular layout and low power consumption.

The sum output of a conventional full adder represents the "modulo-2" sum (i.e. logical XOR) of the three input values. Thus, suppressing all carries of the adder cells enables a radix-2 integer multiplier to multiply binary polynomials [9]. A *Dual-Field Adder (DFA)* is, in essence, an ordinary full adder with some extra logic to set the carry output to zero [34]. Figure 2 shows the DFA design from [16], which was optimized for low power consumption. The control signal *fsel* allows for switching between integer and polynomial mode. In integer mode (i.e. $fsel = 1$), the DFA works like a conventional full adder, i.e. both the sum and the carry output are active. On the other hand, when the DFA operates in polynomial mode ($fsel = 0$), the NAND gates are 1, which forces the ORNAND gate (and consequently *cout*) to 0. An advantage of this design is that only the two XOR gates are active in polynomial mode; all other gates do not transition and, hence, they also do not consume power. Therefore, the DFA illustrated in Figure 2 has a substantially lower power dissipation in polynomial mode than in integer mode. Savaş et al's DFA design from [34] does not have this feature as it is prone to unnecessary gate transitions in polynomial mode.

We implemented the DFA using the cells of our full-custom library in such a way that all transistors have minimal size. The only exception are the PMOS transistors of the ORNAND gate, which were enlarged (as can be seen in the layout in Figure 3) to ensure that the sum and carry output have similar drive strengths and rise/fall times. A simulation of the DFA's netlist with extracted parasitics confirmed that the delay from the inputs *sin*, *cin* to the two outputs *sout* and *cout* is also almost identical. The usage of minimum-size transistors in combination with delay balancing helps to reduce the impact of spurious transitions (i.e. glitches) on the total power consumption. Other advantages of the DFA are its relatively small area and a short critical path. The five gates shown in Figure 2 amount to 32 transistors altogether (on basis of our full-custom cell library), which is only four transistors more than the classical 28-transistor full adder design given in [37, p. 517]. The delay of a full adder is generally determined by the delay of the XOR gates, which means that our DFA has the same critical-path delay as a conventional full adder. In summary, the DFA shown in Figure 2 is only slightly larger than a full adder and has the same propagation delay. The complete adder array of a unified (16×16)-bit multiplier has, in the radix-2 case, a critical path of 14 DFAs and one dual-field half adder.

3.3 Accumulator and Final Adder

The presented adder-array compresses the partial products to a single sum and carry vector. An accumulator allows for adding the result of the multiplication (i.e. the product) to a cumulative sum. The accumulator of our unified MAC unit consists of DFAs and has a length of 40 bits (i.e. we have 8 guard bits to prevent overflows). The output of the accumulator is exactly the result of the multiply/accumulate operation in a redundant representation, the carry-save form. Consequently, we have to convert the sum and carry vector into a normal binary number to obtain the final result.

The redundant-to-binary conversion ("final addition") calls for a fast carry-propagation adder [37]. An important aspect when designing a final adder is to consider the non-uniform signal arrival profile of the sum and carry vector as explained in [28]. Array multipliers have a typical "staircase-like" signal arrival profile, which means that the lower half of the result appears sequentially (in a bit-by-bit fashion), whereas the upper part arrives simultaneously after passing through the full adder array. In order to reduce the overall delay of the MAC unit, we designed the final adder to match this special arrival profile. Our final adder consists of a ripple-carry adder for redundant-to-binary conversion of the 16 sequentially-arriving bits, and a carry-select adder (CSA) for the upper bits of the result. We used ripple-carry adders of varying length for the sub-stages of the CSA to reduce the delay. A CSA composed of small ripple-carry adders features high regularity and is, therefore, suitable for full-custom design.

The multiplication of binary polynomials does not need a final adder, simply because all carry vectors are 0 in polynomial mode. Therefore, we forward the accumulator's sum output directly to the output of the MAC unit when a polynomial multiplication is performed. The sum input to the final adder is set to 0 in polynomial mode to ensure that the final adder is completely inactive and does not dissipate power.

4 Unified Radix-4 Multiplier

High-radix multiplication schemes reduce the number of partial products compared to the conventional (i.e. radix-2 or binary) multiplication scheme, which shortens the critical path of an array multiplier as fewer partial products have to be summed up. A radix-4 multiplication of two unsigned 16-bit numbers can be performed according to the following equation.

$$Z = A \cdot B = A \cdot \sum_{k=0}^{7} b_k \cdot 4^k = \sum_{k=0}^{7} A \cdot b_k \cdot 4^k \quad \text{with} \quad b_k \in \{0,1,2,3\} \tag{3}$$

The conventional radix-4 representation of integers is based on the digit-set $\{0,1,2,3\}$. Consequently, any partial product $P_k = A \cdot b_k$ is either 0, A, $2A$, or $3A$, whereby the latter is difficult to generate as the addition of $2A$ and A will typically imply a propagation of carries. Therefore, most radix-4 multiplication schemes employ other digit sets, e.g. the set $\{-2,-1,0,1,2\}$ as used in modified Booth recoding (resp. Booth-MacSorley recoding [25]).

The idea of high-radix multiplication is applicable to binary polynomials as well and was first described in [27]. Analogous to integer multiplication, the use of a high-radix representation reduces the overall number of partial products that need to be summed up. The radix-t representation of binary polynomials is based on the coefficient set $\{0,1\}$, and hence it corresponds to the conventional radix-2 representation of integers. On the other hand, the analogue of radix-4 integer representation is the radix-t^2 representation of binary polynomials. This representation uses the coefficient set $\{0,\ 1,\ t,\ t+1\}$ and allows for performing a multiplication of two binary polynomials $A(t)$, $B(t)$ of degree 15 as follows.

$$Z(t) = A(t) \cdot B(t) = A(t) \cdot \sum_{k=0}^{7} b_k(t) \cdot t^{2k} = \sum_{k=0}^{7} A(t) \cdot b_k(t) \cdot t^{2k} \tag{4}$$
$$\text{with} \quad b_k(t) \in \{0,\ 1,\ t,\ t+1\}$$

All coefficients $b_k(t)$ of a binary polynomial $B(t)$ in radix-t^2 representation are themselves binary polynomials, namely binary polynomials of degree up to one (i.e. 0, 1, t, or $t+1$). The corresponding partial product $P_k(t) = A(t) \cdot b_k(t)$ is either 0, $A(t)$, $t \cdot A(t)$, or $A(t) \oplus t \cdot A(t)$, all of which can be easily generated by 1-bit left-shift and XOR operations. Therefore, a radix-t^2 multiplication can be efficiently performed with the standard coefficient set $\{0,\ 1,\ t,\ t+1\}$.

4.1 Generation of Partial Products

Modified Booth recoding (also referred to as Booth-MacSorley recoding [25]) is commonly used in array multipliers as it reduces the number partial products by a factor of two. However, modified Booth recoding is not directly applicable to binary polynomials since it relies on a signed-digit representation with the digit set $\{-2,-1,0,1,2\}$, which does not exist in the context of polynomials as the principle of "carrying" from less to more significant positions is absent. It is nonetheless possible to "unify" the high-radix multiplication of integers and binary polynomials, as was demonstrated in [15]. In the following, we summarize how the unified radix-4/radix-t^2 multiplier from [17] generates partial products in integer mode and polynomial mode.

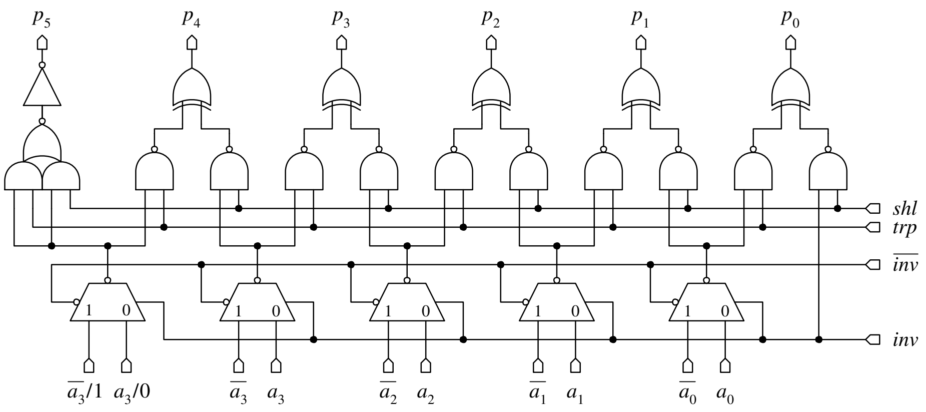


Fig. 4. Unified radix-4 PPG for integers and radix-t^2 PPG for binary polynomials

Integer Mode. Modified Booth recoding is, in general, performed within two steps: *Encoding* of the multiplier B and *Selection* (i.e. generation) of the partial products. The encoding step is simply a conversion in which a radix-2 number B with digits b_i in $\{0, 1\}$ is transformed into an equivalent radix-4 number $\widetilde{B}$ represented by digits $\widetilde{b}_k$ from the set $\{-2, -1, 0, 1, 2\}$. When assuming that B is an unsigned 16-bit integer, the conversion can be carried out as follows.

$$\widetilde{B} = \sum_{k=0}^{8} \widetilde{b}_k \cdot 4^k \text{ with } \widetilde{b}_k = -2 \cdot b_{2k+1} + b_{2k} + b_{2k-1} \text{ and } b_{17} = b_{16} = b_{-1} = 0 \quad (5)$$

The radix-4 digits $\widetilde{b}_k$ are obtained by partitioning the multiplier B into overlapping groups of three adjacent bits b_{2k+1}, b_{2k}, b_{2k-1} (for $k = 0, 1, \ldots, 8$) and calculating $-2 \cdot b_{2k+1} + b_{2k} + b_{2k-1}$ as shown in Equation (5). All digits $\widetilde{b}_k$ are available "simultaneously" because they can be calculated independently from each other and in parallel. A multiplication with the radix-4 number $\widetilde{B}$ instead of B cuts the number of partial products from 16 to 9 (or 8 when multiplying signed integers). The major advantage of using the digit set $\{-2, -1, 0, 1, 2\}$ is that the corresponding partial products $P_k \in \{-2A, -A, 0, A, 2A\}$ are easy to generate through shifts and bit-wise inversions. Booth multipliers generally use two's complement (TC) representation for negative numbers. Negative partial products, in TC form, are obtained via inversion of the corresponding positive partial product (i.e. producing the one's complement) and adding a "1" at the least significant bit position. This "correction" is performed together with the addition of partial products in the adder array.

Figure 4 depicts a unified radix-4/radix-t^2 partial product generator (PPG) for integers and binary polynomials. This design is based on [15] but optimized for implementation using our full-custom cell library. The PPG is controlled by the three signals *inv* (invert), *trp* (transport), and *shl* (shift left). These signals depend on the multiplier bits b_{2k+1}, b_{2k}, and b_{2k-1} according to Equation (6) to

Table 1. Radix-4 encoding of integers and radix-t^2 encoding of binary polynomials

Multiplier bits			Integer mode ($fsel = 1$)					Polynomial mode ($fsel = 0$)				
b_{2k+1}	b_{2k}	b_{2k-1}	P_k	*inv*	*trp*	*shl*	+1	$P_k(t)$	*inv*	*trp*	*shl*	+1
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	$+A$	0	1	0	0	0	0	0	0	0
0	1	0	$+A$	0	1	0	0	$A(t)$	0	1	0	0
0	1	1	$+2A$	0	0	1	0	$A(t)$	0	1	0	0
1	0	0	$-2A$	1	0	1	1	$t \cdot A(t)$	0	0	1	0
1	0	1	$-A$	1	1	0	1	$t \cdot A(t)$	0	0	1	0
1	1	0	$-A$	1	1	0	1	$t \cdot A(t) \oplus A(t)$	0	1	1	0
1	1	1	0	1	0	0	0	$t \cdot A(t) \oplus A(t)$	0	1	1	0

(8), which can be directly derived from Table 1. In integer mode ($fsel = 1$), the generation of partial products is done similar to the classical encoding scheme using *N/X1/X2* signals as specified in [37, p. 551]. Therefore, the unified PPG acts like an ordinary radix-4 Booth-PPG when operated in integer mode. The logical equation for the "+1" needed in the case of a negative partial product is also easily derived from Table 1.

$$inv = fsel \cdot b_{2k+1} \tag{6}$$

$$trp = fsel \cdot \left(\overline{b}_{2k} \cdot b_{2k-1} \ + \ b_{2k} \cdot \overline{b}_{2k-1}\right) \ + \ \overline{fsel} \cdot b_{2k} \tag{7}$$

$$shl = fsel \cdot \left(\overline{b}_{2k+1} \cdot b_{2k} \cdot b_{2k-1} + b_{2k+1} \cdot \overline{b}_{2k} \cdot \overline{b}_{2k-1}\right) + \overline{fsel} \cdot b_{2k+1} \tag{8}$$

The PPG shown in Figure 4 needs A (the multiplicand) and its inverse $\overline{A}$ as input, and the multiplexers select between A and $\overline{A}$, depending on the control signal *inv*. The AND/XOR combination performs a 1-bit left-shift operation when $shl = 1$, which is necessary for the generation of the partial products $2A$ and $-2A$. On the other hand, $trp = 1$ means that no left shift is performed and hence the resulting partial product is either A or $-A$.

Polynomial Mode. The radix-t^2 multiplication of binary polynomials corresponds to the radix-4 multiplication of integers. Given two binary polynomials $A(t)$, $B(t)$ in conventional (radix-t) representation, the radix-t^2 multiplication requires to consider two adjacent bits of $B(t)$ at a time in order to produce the corresponding partial product $P_k(t)$ as specified by Equation (5). However, two adjacent bits of $B(t)$ can also be interpreted as a binary polynomial of degree one, and depending on whether this polynomial is 0, 1, t, or $t + 1$, the corresponding partial product $P_k(t)$ is either 0, $A(t)$, $t \cdot A(t)$, or $t \cdot A(t) \oplus A(t)$. The multiplication of $A(t)$ by t is nothing else than a 1-bit left shift operation of the coefficients of $A(t)$, which means that the generation of partial products for a radix-t^2 multiplication is simply a matter of shift and XOR operations.

An important property of the *inv/trp/shl* scheme is that, in integer mode (i.e. $fsel = 1$), the two control signals *trp* and *shl* are never 1 at the same time (see Table 1), which allows us to use XOR or XNOR gates to select between

$\pm A$ and $\pm 2A$. Thanks to this property, the PPG depicted in Figure 4 provides the necessary functionality to generate partial products for radix-t^2 multiplication. In polynomial mode (i.e. $fsel = 0$), the control signal *inv* is always 0 and the PPG is directly controlled by the coefficients of $B(t)$, i.e. $trp = b_{2k}$ and $shl = b_{2k+1}$. Therefore, the PPG depicted in Figure 4 is a unified radix-4 PPG for integers and binary polynomials[2].

4.2 Addition of Partial Products

Applying radix-4 Booth recoding in a (16×16)-bit multiplier reduces the number of partial products from 16 to 9 (in the case of unsigned integers). On the other hand, radix-t^2 multiplication of polynomials halves the number of partial products compared to the radix-t architecture (i.e. 8 partial products instead of 16). The adder array of a unified radix-4 multiplier for 16-bit operands is, in general, dimensioned to sum up 9 partial products [17]. This means that the radix-4/radix-t^2 scheme reduces the overall number of DFAs by almost 50% in relation to the radix-2/radix-t datapath. However, the adder array of a unified radix-4 multiplier differs in the following aspects from the radix-2 version.

– The length of the partial products is 18 bits instead of 16 bits since they can be twice as large as the multiplicand A and may have a negative value. The MSB of the partial product is its sign bit.
– The rules of two's complement arithmetic demand a sign extension, which increases both area and power consumption. Therefore, it is important to minimize the effects of sign extension.
– The PPG shown in Figure 4 performs merely an inversion when the generation of a negative partial product is required. Therefore, the adder array has to add a "1" at the least significant position of the partial product in order to get the correct two's complement representation.
– The partial products must be summed up according to their weight (i.e. in the appropriate relative position) to get the correct result. For instance, the partial product P_k has four times the weight of P_{k-1}, which means that the offset between P_k and $P_{k-1}$is two bit positions.

Detailed information about the implementation of an adder array for unified radix-4/radix-t^2multiplication is given in [17]. Although the architecture from [17] was originally developed for a standard-cell implementation, it is also well suited for full-custom design since it features a high degree of regularity and mainly local interconnect. The adder array of a unified (16×16)-bit multiplier implemented on basis of [17] consists of 7 adder stages to sum up the 9 partial products. Every adder stage is composed of 18 DFAs, which amounts to 126 DFAs altogether. Note that the first three partial products, P_0, P_1, and P_2, can be summed up by one adder stage. All remaining partial products require an additional adder stage,

[2] We denote the combined radix-4/radix-t^2 PPG as a "unified radix-4 PPG for integers and binary polynomials" since the radix-t^2 multiplication of binary polynomials corresponds to the radix-4 multiplication of integers.

which yields the 7 adder stages mentioned before. The critical path of the radix-4 adder array (i.e. 7 DFA cells) is only one half of the radix-2 array.

4.3 Accumulator and Final Adder

The 40-bit accumulator of the unified radix-4 multiplier is identical to the one of the radix-2 version. On the other hand, the signal arrival profiles, and hence the final adders, differ slightly. The radix-4 multiplier also has a staircase-like profile, but the low-order bits (i.e. the 16 LSB) of the sum and carry vector arrive sequentially in two-bit blocks, and not one bit at a time as in the radix-2 version. Therefore, the final adder uses concatenated 2-bit ripple-carry adders for the redundant-to-binary conversion of these bits. The high-order bits of the sum and carry vector arrive simultaneously (as in the radix-2 multiplier), and are summed up by a fast carry-select adder.

5 Results and Discussion

We created a full-custom layout of both the unified radix-2 and radix-4 MAC unit using a standard 0.6 μm CMOS technology with two metal layers and a single polysilicon layer. The transistor width of gates with ordinary (1x) drive strength is always 1.5 μm; gates with 2x drive strength are realized with PMOS transistors of width 3 μm and NMOS transistors of 1.5 μm, respectively. Both the radix-2 and the radix-4 variant have a regular structure with mainly local interconnect (i.e. short wires), which allows using minimum-size transistors in gates and drivers. However, some gates demand an increased drive strength in order to achieve equal output signal strength or balance the delay and rise/fall times of signal slopes. Delay balancing ensures synchronously arriving signals at the input of logic gates, thereby eliminating, or substantially reducing, the power consumption caused by spurious transitions (glitches).

Table 2 summarizes the simulation results and main characteristics of the two unified $(16 \times 16 + 40)$-bit MAC units. Both the radix-2 and the radix-4 version consist of roughly 12,000 transistors, whereby the silicon area of the radix-2

Table 2. Simulation results for f = 10 MHz and Vdd = 3.3 V

Parameter	Radix-2 version	Radix-4 version
Transistor count	12,384	11,744
Delay (INT mode)	82.4 nsec	54.3 nsec
Avg. current (INT mode)	7.37 mA	5.75 mA
PDP (INT mode)	2.43 nJ	1.90 nJ
EDP (INT mode)	$200.2 \cdot 10^{-18}$ Js	$103.0 \cdot 10^{-18}$ Js
Delay (POLY mode)	66.8 nsec	38.0 nsec
Avg. current (POLY mode)	3.66 mA	3.49 mA
PDP (POLY mode)	1.21 nJ	1.15 nJ
EDP (POLY mode)	$80.7 \cdot 10^{-18}$ Js	$43.8 \cdot 10^{-18}$ Js

MAC unit is slightly larger. The radix-4 variant has a worst-case delay of 54.3 nsec (in integer mode at a supply voltage of 3.3 V), which corresponds to a maximum clock frequency of 18.4 MHz. As expected, the radix-2 MAC is slower, mainly due to the longer critical path in the adder array. Both versions are not very fast since we used an old 0.6 μm CMOS process and most of the transistors are of minimum size. However, the performance is still appropriate for common RFID applications. Each of our MAC units executes a polynomial multiplication much faster than an integer multiplication since a redundant-to-binary conversion of the result is not required in polynomial mode (i.e. the final adder is bypassed).

The average current drawn by the radix-2 and the radix-4 implementation is also given in Table 2. These results were obtained via simulation of netlists with extracted parasitics. In general, the current drawn during a multiplication depends heavily on the input values. Therefore, we simulated 10,000 multiply-accumulate operations using independent, pseudo-random input patterns and measured the power consumption. The simulation was done with a frequency of 10 MHz (i.e. new inputs were presented with a period of 100 nsec), which is a frequency that both the radix-2 and the radix-4 MAC unit could handle. Our results show that the radix-4 MAC draws, on average, 22% less power than the radix-2 version. This power saving is mainly due to the shorter adder array in the radix-4 multiplier, which reduces the number of glitches compared to the radix-2 version. In both cases, a multiplication of binary polynomials consumes significantly less power than an integer multiplication (e.g. roughly 39% in the radix-4 version). Polynomial multiplication needs no redundant representation and, therefore, causes less switching activities in the adder array (only the two XNORs of each DFA are active) and no switchings at all in the final adder.

Besides area, delay, and power consumption, also the Power-Delay Product (PDP) and the Energy-Delay Product (EDP) are widely accepted comparison metrics for MAC units. When leakage current is ignored, the PDP of a static CMOS multiplier can be interpreted as the average figure of energy consumed per multiplication. Thus, the PDP is an important metric for battery-operated devices as it determines the battery-lifetime. Our simulations showed that the average amount of energy needed to perform an integer multiplication is 2.43 nJ for the radix-2 MAC, but only 1.90 nJ for the radix-4 variant, which represents a saving of 22%. The EDP differs by more than 48% in integer mode. Both the radix-2 and the radix-4 version have significantly better energy characteristics in polynomial mode than in integer mode, which confirms that a multiplication of binary polynomials is more energy-efficient than integer multiplication.

6 Conclusions

In this paper, we analyzed and compared two implementations of a unified $(16 \times 16 + 40)$-bit MAC unit for integers and binary polynomials. Our first implementation utilizes a unified radix-2/radix-t multiplication scheme, whereas the second employs radix-4 Booth recoding for integers and radix-t^2 multiplication for binary polynomials. Both the radix-2 and radix-4 variant are based on a normal array architecture and perform multiplications and MAC operations in one cycle. The MAC unit can be integrated into low-cost ECC hardware to perform all arithmetic operations in GF(p) and GF(2^m).

Our simulation results show that the unified radix-4 MAC is superior to its radix-2 counterpart in terms of area, delay, and power consumption. While the difference in silicon area is only marginal, the radix-4 version achieves a 34% improvement in delay and a power advantage of 22% compared to the radix-2 version. Moreover, the radix-4 MAC exhibits a much better EDP. Taking the tight power budget of RFID tags into account, our unified radix-4 multiplier is a significant improvement over the radix-2 designs proposed in [34,33]. We also demonstrated that the multiplication of binary polynomials is more efficient in terms of power and energy than integer multiplication because the latter uses a redundant representation which causes more signal transitions.

Acknowledgements. A preliminary version of this paper was presented at the ECRYPT Workshop on RFID and Lightweight Crypto. The author is grateful to Dan Page for valuable comments and suggestions.

References

1. Au, L.S., Burgess, N.: A (4:2) adder for unified GF(p) and GF(2^n) Galois field multipliers. In: Conference Record of the 36th Asilomar Conference on Signals, Systems, and Computers, vol. 2, pp. 1619–1623. IEEE, Los Alamitos (November 2002)
2. Avoine, G.: Scalability issues in privacy-compliant RFID protocols. In: Kitsos, P., Zhang, Y. (eds.) RFID Security: Techniques, Protocols and System-On-Chip Design, ch. 9, pp. 191–228. Springer, Heidelberg (2008)
3. Bisdounis, L., Gouvetas, D., Koufopavlou, O.: Circuit techniques for reducing power consumption in adders and multipliers. In: Soudris, D., Piguet, C., Goutis, C. (eds.) Designing CMOS Circuits for Low Power, ch. 5, pp. 71–96. Kluwer Academic Publishers, Dordrecht (2002)
4. Bock, H., Braun, M., Dichtl, M., Heyszl, J., Hess, E., Kargl, W., Koroschetz, H., Meyer, B., Seuschek, H.: A milestone towards RFID products offering asymmetric authentication based on elliptic curve cryptography. In: Proceedings of the 4th Workshop on RFID Security (RFIDSec 2008), Budapest, Hungary (June 2008)
5. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.H.: PRESENT: An ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
6. Callaway, T.K., Swartzlander, E.E.: Power-delay characteristics of CMOS multipliers. In: Proceedings of the 13th IEEE Symposium on Computer Arithmetic (ARITH 1997), pp. 26–32. IEEE Computer Society Press, Los Alamitos (July 1997)
7. Chang, A., Dally, W.J.: Explaining the gap between ASIC and custom power: A custom perspective. In: Proceedings of the 42nd Design Automation Conference (DAC 2005), pp. 281–284. ACM Press, New York (June 2005)
8. Dally, W.J., Chang, A.: The role of custom design in ASIC chips. In: Proceedings of the 37th Design Automation Conference (DAC 2000), pp. 643–647. ACM Press, New York (June 2000)

9. Drescher, W., Bachmann, K., Fettweis, G.: VLSI architecture for datapath integration of arithmetic over GF(2^m) on digital signal processors. In: Proceedings of the 22nd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 1997), Munich, Germany, vol. 1, pp. 631–634 (April 1997)
10. Fan, J., Batina, L., Verbauwhede, I.: Implementation of hyperelliptic curve cryptography using a unified multiplier and inverter. Tech. rep., ESAT/COSIC, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium (July 2009), http://www.cosic.esat.kuleuven.be/publications/article-1293.pdf
11. Fan, J., Batina, L., Verbauwhede, I.: Light-weight implementation options for curve-based cryptography: HECC is also ready for RFID. In: Proceedings of the 4th International Conference for Internet Technology and Secured Transactions (ICITST 2009), pp. 845–850. IEEE, Los Alamitos (2009)
12. Fürbass, F., Wolkerstorfer, J.: ECC processor with low die size for RFID applications. In: Proceedings of the 40th IEEE International Symposium on Circuits and Systems (ISCAS 2007), pp. 1835–1838. IEEE, Los Alamitos (2007)
13. Garcia, J.E., Schulte, M.J.: A combined 16-bit binary and dual Galois field multiplier. In: Proceedings of the 16th IEEE Workshop on Signal Processing Systems (SIPS 2002), pp. 63–68. IEEE, New York (October 2002)
14. Großschädl, J.: A low-power bit-serial multiplier for finite fields GF(2^m). In: Proceedings of the 34th IEEE International Symposium on Circuits and Systems (ISCAS 2001), vol. IV, pp. 37–40. IEEE, Los Alamitos (May 2001)
15. Großschädl, J.: A unified radix-4 partial product generator for integers and binary polynomials. In: Proceedings of the 35th IEEE International Symposium on Circuits and Systems (ISCAS 2002), vol. III, pp. 567–570. IEEE, Los Alamitos (May 2002)
16. Großschädl, J., Kamendje, G.A.: Instruction set extension for fast elliptic curve cryptography over binary finite fields GF(2^m). In: Deprettere, E., Bhattacharyya, S., Cavallaro, J., Darte, A., Thiele, L. (eds.) Proceedings of the 14th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2003), pp. 455–468. IEEE Computer Society Press, Los Alamitos (June 2003)
17. Großschädl, J., Kamendje, G.A.: Low-power design of a functional unit for arithmetic in finite fields GF(p) and GF(2^m). In: Chae, K.-J., Yung, M. (eds.) WISA 2003. LNCS, vol. 2908, pp. 227–243. Springer, Heidelberg (2004)
18. Hankerson, D.R., Menezes, A.J., Vanstone, S.A.: Guide to Elliptic Curve Cryptography. Springer, Heidelberg (2004)
19. Hein, D., Wolkerstorfer, J., Felber, N.: ECC is ready for RFID — A proof in silicon. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 401–413. Springer, Heidelberg (2009)
20. Juels, A.: RFID security and privacy: A research survey. IEEE Journal on Selected Areas in Communications 24(2), 381–394 (2006)
21. Kaya, S.V., Savaş, E., Levi, A., Erçetin, Ö.: Privacy-aware multi-context RFID infrastructure using public key cryptography. In: Akyildiz, I.F., Sivakumar, R., Ekici, E., de Oliveira, J.C., McNair, J. (eds.) NETWORKING 2007. LNCS, vol. 4479, pp. 263–274. Springer, Heidelberg (2007)
22. Kobayashi, K., Takagi, N., Takagi, K.: An algorithm for inversion in GF(2^m) suitable for implementation using a polynomial multiply instruction on GF(2). In: Proceedings of 18th IEEE Symposium on Computer Arithmetic (ARITH 2007), pp. 105–112. IEEE Computer Society Press, Los Alamitos (June 2007)

23. Lee, H., Sobelman, G.E.: New low-voltage circuits for XOR and XNOR. In: Proceedings of IEEE SouthEastCon 1997, pp. 225–229. IEEE, Los Alamitos (April 1997)
24. Lee, Y.K., Batina, L., Sakiyama, K., Verbauwhede, I.: Elliptic curve based security processor for RFID. IEEE Transactions on Computers 57(11), 1514–1527 (2008)
25. MacSorley, O.L.: High-speed arithmetic in binary computers. Proceedings of the IRE 49(1), 67–91 (1961)
26. Meier, P.C., Rutenbar, R.A., Carley, L.R.: Exploring multiplier architecture and layout for low power. In: Proceedings of the 18th IEEE Custom Integrated Circuits Conference (CICC 1996), pp. 513–516. IEEE, Los Alamitos (May 1996)
27. Mekhallalati, M.C., Ashur, A.S., Ibrahim, M.K.: Novel radix finite field multiplier for GF(2^m). Journal of VLSI Signal Processing 15(3), 233–245 (1997)
28. Oklobdžija, V.G.: Design and analysis of fast carry-propagate adder under non-equal input signal arrival profile. In: Conference Record of the 28th Asilomar Conference on Signals, Systems, and Computers, vol. 2, pp. 1398–1401. IEEE, Los Alamitos (October 1994)
29. Oren, Y., Feldhofer, M.: A low-resource public-key identification scheme for RFID tags and sensor nodes. In: Basin, D.A., Capkun, S., Lee, W. (eds.) Proceedings of the 2nd ACM Conference on Wireless Network Security (WISEC 2009), pp. 59–68. ACM Press, New York (2009)
30. Poschmann, A., Robshaw, M.J., Vater, F., Paar, C.: Lightweight cryptography and RFID: Tackling the hidden overheads. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 129–145. Springer, Heidelberg (2010)
31. Rolfes, C., Poschmann, A., Leander, G., Paar, C.: Ultra-lightweight implementations for smart devices – Security for 1000 gate equivalents. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 89–103. Springer, Heidelberg (2008)
32. Sakiyama, K.: Secure Design Methodology and Implementation for Embedded Public-Key Cryptosystems. Ph.D. Thesis, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium (December 2007)
33. Satoh, A., Takano, K.: A scalable dual-field elliptic curve cryptographic processor. IEEE Transactions on Computers 52(4), 449–460 (2003)
34. Sava, E., Tenca, A.F., Koç, Ç.K.: A scalable and unified multiplier architecture for finite fields $GF(p)$ and $GF(2^m)$. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 277–292. Springer, Heidelberg (2000)
35. Vaudenay, S.: On privacy models for RFID. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 68–87. Springer, Heidelberg (2007)
36. Weis, S.A., Sarma, S.E., Rivest, R.L., Engels, D.W.: Security and privacy aspects of low-cost radio frequency identification systems. In: Hutter, D., Müller, G., Stephan, W., Ullmann, M. (eds.) SPC 2003. LNCS, vol. 2802, pp. 201–212. Springer, Heidelberg (2004)
37. Weste, N.H., Eshraghian, K.: Principles of CMOS VLSI Design: A Systems Perspective, 2nd edn. Addison-Wesley, Reading (1993)
38. Wolkerstorfer, J.: Is elliptic-curve cryptography suitable to secure RFID tags? In: Proceedings of the ECRYPT Workshop RFID and Lightweight Crypto, Graz, Austria, pp. 78–91 (July 2005)
39. Zimmermann, R., Fichtner, W.: Low-power logic styles: CMOS versus pass-transistor logic. IEEE Journal of Solid-State Circuits 32(7), 1079–1090 (1997)

Weaknesses in Two Recent Lightweight RFID Authentication Protocols

Pedro Peris-Lopez[1], Julio C. Hernandez-Castro[2], Juan M.E. Tapiador[3], Tieyan Li[4], and Jan C.A. van der Lubbe[1]

[1] Department of Information and Communication, Delft University of Technology
[2] School of Computing, University of Portsmouth
[3] Department of Computer Science, University of York
[4] Institute for Infocomm Research, A*STAR Singapore

Abstract. The design of secure authentication solutions for low-cost RFID tags is still an open and quite challenging problem, though many algorithms have been published lately. In this paper, we analyze two recent proposals in this research area. First, Mitra's scheme is scrutinized, revealing its vulnerability to cloning and traceability attacks, which are among the security objectives pursued in the protocol definition [1]. Later, we show how the protocol is vulnerable against a full disclosure attack after eavesdropping a small number of sessions. Then, we analyze a new EPC-friendly scheme conforming to EPC Class-1 Generation-2 specification (ISO/IEC 180006-C), introduced by Qingling and Yiju [2]. This proposal attempts to correct many of the well known security shortcomings of the standard, and even includes a BAN logic based formal security *proof*. However, notwithstanding this formal security analysis, we show that Qingling et al.'s protocol offers roughly the same security as the standard they try to improve, is vulnerable to tag and reader impersonation attacks, and allows tag traceability.

Keywords: RFID, EPC, Cloning, Traceability, Impersonation, Cryptanalysis, BAN, Proof.

1 Introduction

Many authors have recently focused their attention on low-cost RFID tags, because designing secure solutions within their restricted capabilities is a great challenge. The solutions proposed can be categorized with respect to three main criteria. First, some proposals are based on the learning parity with noise (LPN) problem, initially examined by Hopper and Blum [3] and introduced by Juels in the context of RFID systems [4]. Secondly, other authors severely restrict the assumed set of operations supported by tags to very simple and efficient operations: The SASI [5] and Gossamer [6] protocols are two proposals in this direction, where tag capabilities are limited to bitwise operations and rotations. These protocols have been christened ultra-lightweight protocols, in Chien's classification [5]. When we add to the described tags the requirement of supporting

Pseudo Random Number Generators (PRNGs), we then call these proposals lightweight. For example, EPC Class-1 Generation-2 (Gen-2 in short) compliant tags support a 16-bit PRNG and a 16-bit Cyclic Redundancy Check (CRC) [7].

In this paper, we successfully cryptanalyse two recent lightweight authentication protocols. First, Mitra's scheme [1] is explored, discovering that its two main objectives (anti-cloning and untraceability) are not guaranteed. After that, we present a full disclosure attack that points out the protocol fails short of the security level required for the intended applications. Then, a new scheme under the EPC Gen-2 framework is scrutinized. As is already well-known, the Gen-2 specification has some important security drawbacks. In [2], Qingling et al. made an attempt to correct many of them in their proposed scheme, but we show that the authors failed, just as those of many other previous protocols [8,9,10,11], despite providing a formal security *proof*.

2 Mitra's Protocol

In 2008, Mitra proposed a new scheme (in the following called TC-RFID for short) that attempts to protect tags against traceability and cloning [1]. The author assumes that tags support an on-chip PRNG. Tags are also able to compute simple operations, particulary multiplication and addition. Operations in readers are limited to the computation of a modulo. Regarding communication channels, both the forward (reader-to-tag) and the backward (tag-to-reader) can be eavesdropped by an adversary. As for memory requirements, each tag stores a static identifier $\{EPC_i\}$ and a key $\{K_i\}$. This key is shared between the tag and legitimate readers (and the back-end database) registered in the system.

The author proposed this simple protocol, in which only tags ($\mathcal{T}$) are authenticated by readers ($\mathcal{R}$):

Step 1: $\mathcal{R} \rightarrow \mathcal{T}_i$ The reader sends a request message to tag i.

Step 2: $\mathcal{T}_i$ (Tag i) computes an *encrypted* and/or *anonymized* version of its static identifier:

$$E_i(n) = RND(n) * K_i + EPC_i \tag{1}$$

where RND is a random number, the output after the n^{th}-call to the on-chip PRNG, and K_i is the shared key between $\mathcal{T}_i$ and $\mathcal{R}$.

Step 3: $\mathcal{T}_i \rightarrow \mathcal{R}$ The tag sends the reader $E_i(n)$, which serves as an authentication token.

3 Vulnerabilities of Mitra's Protocol

In this section we analyze the most relevant weaknesses of the TC-RFID protocol.

3.1 Cloning Attack

As mentioned in the last section, nowadays tags often respond to reader's queries without requiring any authentication at all. Tags can even transmit their static identifier $\{EPC_i\}$ over the channel in plaintext. In it case, an adversary can snoop this publicly available information and transfer it to a clone device (i.e. another tag or a more sophisticated emulator).

Symmetric-key cryptography can be used to avoid tag cloning attacks. Specifically, a challenge-response mechanism like the following can be employed. We assume the tag ($\mathcal{T}_i$) shares a secret key $\{K_i\}$ with the reader ($\mathcal{R}$). Afterwards, the following messages are exchanged:

Step 1: $\mathcal{R} \rightarrow \mathcal{T}_i$ The reader generates a fresh random number ($RND(n)$, a nonce challenge) and transmits it to the tag.

Step 2: $\mathcal{T}_i \rightarrow \mathcal{R}$ The tag computes $H_i(n) = g(K_i, RND(n))$ and sends it back to the reader.

Step 3: $\mathcal{R}$ The reader locally computes $H_i'(n) = g(K_i, RND(n))$ and checks if its value is identical to tag's answer H_i.

The g function can be implemented by using any hash or encryption algorithm. Note that the protocol security is highly dependent on that of the g function. As low-cost RFID tags have severe resource limitations, the use of standard cryptographic primitives is not possible. Particularly, Mitra proposed the use of multiplication and summation operands for the g function ($H_i(n) = E_i(n) = RND(n) * K_i + EPC_i$), which is vulnerable to cloning attacks. An attacker can collect a number of encrypted messages, and compute their difference:

$$\Delta = E_i(n) - E_i(n+1) = (RND(n) - RND(n+1)) * K_i \tag{2}$$

Then, she will compute the greatest common divisor of these differences. The attacker concludes this value is the secret key $\{K_i\}$ of the target tag.

Additionally, an attacker that eavesdrops on two (non necessarily consecutive) authentication sessions ($\{E_i(n), E_i(n+p)\}$) between the target tag ($\mathcal{T}_i$) and a legitimate reader ($\mathcal{R}$) is able to supplant a tag indefinitely by sending $E_i' = E(n) + RND_r(q) * \Delta$ as its authentication token, where $\Delta = E_i(n) - E_i(n+p) = (RND(n) - R(n+p)) * K_i$.

3.2 Traceability Attack

The traceability problem has been studied by many researchers lately. In [12], Juels and Weis give a formal definition of the untraceability model. The same definition, though in a style more commonly used to formally define the properties of security protocols, is described by Phan in his recent attack against the SASI protocol [13], and used in the following.

In RFID systems, tags ($\mathcal{T}$) and readers ($\mathcal{R}$) interact in protocol sessions. In general terms, the adversary ($\mathcal{A}$) controls the communications between all the participants and interacts passively or actively with them. In our case, we can succeed in the traceability attack by only using passive means. Specifically, $\mathcal{A}$ can run the following queries:

- Execute($\mathcal{R}$, $\mathcal{T}$, i) query. This models a passive attacker. $\mathcal{A}$ eavesdrops on the channel, and gets read access to the exchanged messages between $\mathcal{R}$ and $\mathcal{T}$ in session i of a genuine protocol execution.
- Test(i, $\mathcal{T}_0$, $\mathcal{T}_1$) query. This does not model any ability of $\mathcal{A}$, but it is necessary to define the untraceability test. When this query is invoked in session i, a random bit is generated $b \in \{0,1\}$. Then, $E_b(n)$, from the set $\{E_0(n), E_1(n)\}$ corresponding to tags $\{\mathcal{T}_0, \mathcal{T}_1\}$ is given to $\mathcal{A}$.

Upon definition of the adversary's abilities, the untraceability problem can be defined as a game $\mathcal{G}$. We now show why the TC-RFID scheme does not achieve untraceability. Specifically, the TC-RFID protocol, in an RFID system (S= $\{R_i$, $\mathcal{T}_0$, $\mathcal{T}_1$, ...$\}$ in which an adversary $\mathcal{A}$ can invoke two Execute($\mathcal{R}$, $\mathcal{T}$, i) queries and one Test(i, $\mathcal{T}_0$, $\mathcal{T}_1$) query, is vulnerable to traceability attacks, since the advantage for an adversary is significant: $Adv_{\mathcal{A}}^{UNT}(t, r = 2) = 0.5 \gg \varepsilon(t, 2)$, t being a security parameter (i.e. the bit length of the secret key) and $\varepsilon(.)$ some negligible function.

Specifically, an adversary $\mathcal{A}$ performs the following steps:

Phase 1 (Learning): $\mathcal{A}$ sends two Execute queries to $\mathcal{T}_0$. Consecutiveness in the queries is not necessary, which is handy because the target tag may have been read by a legitimate reader in between. $\mathcal{A}$ acquires the following messages:

$$E_0(n) = RND(n) * K_0 + EPC_0 \tag{3}$$
$$E_0(n+m) = RND(n+m) * K_0 + EPC_0 \tag{4}$$

Phase 2 (Challenge): $\mathcal{A}$ chooses two fresh tags whose associated identifiers are EPC_0 and EPC_1. Then he sends a Test(q, $\mathcal{T}_0$, $\mathcal{T}_1$) query. As a result, $\mathcal{A}$ is given a challenge cipher text $E_b(q)$ from the set $\{E_0(q), E_1(q)\}$, which depends on a chosen random bit $b \in \{0,1\}$:

$$E_i(q) = \begin{cases} RND(q) * K_0 + EPC_0 \text{ if } b = 0 \\ RND(q) * K_1 + EPC_1 \text{ if } b = 1 \end{cases} \tag{5}$$

Phase 3 (Guessing) $\mathcal{A}$ finishes $\mathcal{G}$ and outputs a bit d ($d \in \{0,1\}$) as his guess of the value b. In particular, we propose the following procedure to obtain value d:

1. $\mathcal{A}$ computes the difference between Equations 3, 4:

$$\begin{aligned} \Delta_1 &= |E_0(n) - E_0(n+m)| \\ &= |RND(n) * K_0 + EPC_0 - RND(n+m) * K_0 - EPC_0| \\ &= |(RND(n) - RND(n+m)) * K_0| \end{aligned} \tag{6}$$

2. $\mathcal{A}$ computes the difference between Equations 3, 5:

$$\Delta_2 = \begin{cases} |(RND(n) - RND(q)) * K_0| & \text{if } b = 0 \\ |RND(n) * K_0 - RND(q) * K_1 + EPC_0 - EPC_1| & \text{if } b = 1 \end{cases} \tag{7}$$

3. $\mathcal{A}$ uses the following decision rule:

$$d = \begin{cases} 0 & \text{if } gcd(\Delta_1, \Delta_2) \geq 2^{L/2} \\ 1 & \text{if } gcd(\Delta_1, \Delta_2) < 2^{L/2} \end{cases} \tag{8}$$

where gcd(.) symbolizes the greatest common divisor and L represents the length of the variables used. To be compatible with the common encodings schemes defined by EPCGlobal [14], we can fix L to 96 bits.

We have run 1000 experiments, with 2^{20} executions of the above game ($\mathcal{G}$) in each experiment, in order to obtain an approximation of the success probability of the adversary. From this experimentation, a probability of 1 was obtained. So, the $Adv_{\mathcal{A}}^{UNT}(t, r)$ is not only not negligible, but maximal. $\mathcal{A}$ wins $\mathcal{G}$, allowing him the traceability of tags: $Adv_{\mathcal{A}}^{UNT}(t, 2) = |1 - \frac{1}{2}| = 0.5$. So the use of random numbers does not prevent the attacker from associating the tags's answers with its holder, with complete certainty.

3.3 Full Disclosure Attack

In this section, we show a much more harmful attack in which the attacker disclosures the secret key $\{K_i\}$ of the target tag. Once $\{K_i\}$ is made public, the static identifier $\{EPC_i = E_i(n + c) \mod K_i\}$ is compromised too. That is, all the private information of the tag is exposed by the adversary.

Specifically, in the TC-RFID protocol, an adversary that eavesdrops on t (non necessarily consecutive) authentication sessions $\{E_i(n+c_1), E_i(n+c_2), ..., E_i(n+c_t)\}$ between the target tag ($\mathcal{T}_i$) and a legitimate reader ($\mathcal{R}$) is able to disclosure the secret key $\{K_i\}$ by computing the greatest common divisor of the t–1 independent differences $\{|E_i(n + c_1) - E_i(n + c_2)|, ..., |E_i(n + c_1) - E_i(n + c_t)|\}$. The probability of success –from basic Number Theory [15]– is quite accurately given by the equation bellow:

$$Pr[\mathcal{A}\ reveals\ K_i] \approx \frac{1}{\zeta(t-1)} \tag{9}$$

where ζ is the Riemann zeta function.

We ran 10,000 experiments in order to estimate the Adversary's probability of success. We conducted this experiment for several numbers of eavesdropped sessions. In Figure 1, the good fitting between the theoretical and experimental results is depicted. We observe that after a low number [2 : 11] of eavesdropped sessions the attack is quite successful (60% – 100%). The attack just presented here is the most powerful and implies all those described in previous sections (i.e. privacy exposure, cloning, traceability, etc.).

4 Qingling et al.'s Protocol

In 2008, Quingling et al. proposed a minimalist mutual authentication protocol conforming with Gen-2 specification (QYY-Gen2 in short) [2]. A formal security

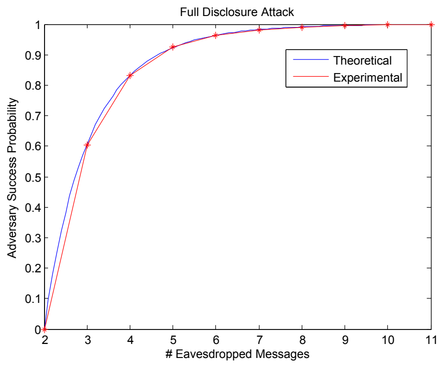


Fig. 1. Full Disclosure Attack

analysis was included in their proposal, which generated some hopes that it could be the first solution to really increase the security of the Standard. We have analyzed the scheme, and realized these hopes were overoptimistic, because the protocol has the same security weaknesses as the Gen-2 Standard it is intended to improve, as shown in Section 5.

The authors proposed a challenge-response protocol in which both tags ($\mathcal{T}_i$) and reader ($\mathcal{R}$) are authenticated. Each tag and reader (back-end database) share a 32-bit EPC unique identifier $\{TID^{Tag_i}\}$ and a 32-bit access password $\{APWD^{Tag_i}\}$. For simplicity, we condense readers and back-end database into a single entity as the communication channel between both entities is assumed to be secure. The subindex M and L are used to represent the 16 most and least significant bits of a variable (i.e. $X = X_L||X_M$). We outline the messages exchanged bellow:

Step 1: $\mathcal{R} \rightarrow \mathcal{T}_i$**:** $Query$**,** RND^{Rdr}

The reader first generates a random number RND^{Rdr} and sends $\{Query, RND^{Rdr}\}$ to the target tag.

Step 2: $\mathcal{T}_i \rightarrow \mathcal{R}$**:** M^{Tag_i}**,** RND^{Tag_i}

Upon receiving reader's query, the tag also generates a nonce RND^{Tag_i} and computes: $M^{Tag_i} = M_L^{Tag_i}||M_M^{Tag_i}$

$$M_L^{Tag_i} = CRC(TID_L^{Tag_i} \oplus RND^{Rdr} \oplus RND^{Tag_i}) \oplus APDW_L^{Tag_i}$$
$$M_M^{Tag_i} = CRC(TID_M^{Tag_i} \oplus RND^{Rdr} \oplus RND^{Tag_i}) \oplus APDW_M^{Tag_i} \quad (10)$$

Finally, the tag sends $\{M^{Tag_i}, RND^{Tag_i}\}$ to the reader.

Step 3: The reader verifies whether the equation $M^{Tag_i} \oplus APDW^{Tag_i} = CRC(TID_L^{Tag_i} \oplus RND^{Rdr} \oplus RND^{Tag_i})||CRC(TID_M^{Tag_i} \oplus RND^{Rdr} \oplus RND^{Tag})$ holds for any tag registered in the back-end database. If it can find a match, the tag is authenticated and the process continues; otherwise it stops the process, which ends in failure.

Step 4: $\mathcal{R} \rightarrow \mathcal{T}_i$**:** $M^{Rdr} = M_L^{Rdr}||M_M^{Rdr}$

The reader computes an authentication message and sends it to the tag.

$$\begin{aligned} M_L^{Rdr} &= CRC(TID_L^{Tag_i} \oplus RND^{Tag_i}) \oplus APDW_L^{Tag_i} \\ M_M^{Rdr} &= CRC(TID_M^{Tag_i} \oplus RND^{Tag_i}) \oplus APDW_M^{Tag_i} \end{aligned} \quad (11)$$

Step 5: On receiving M^{Rdr}, the tag verifies the equation $M^{Rdr} \oplus APDW^{Tag_i} = CRC(TID_L^{Tag_i} \oplus RND^{Tag_i})||CRC(TID_M^{Tag_i} \oplus RND^{Tag_i})$. If it holds, the reader is successful authenticated; otherwise it stops the process, which ends in failure.

5 Vulnerabilities of Qingling's Protocol

In this section we analyze the most relevant weaknesses of the QYY-Gen2 protocol. The authors use a Cyclic Redundancy Check (CRC) function as if it were a "strong" primitive despite its well-known security problems due to its linearity [16,17]. Basically, the security of the proposed scheme is incorrectly based on the assumption that CRC functions are one-way functions.

CRC functions possess some properties that are undesirable from a security point of view. We show one of them, which will be enough to prove that the QYY-Gen2 protocol fails short of its security objectives, being as insecure as the original Gen-2 specification it is intended to improve on. For any CRC (independently of its generator polynomial) and for any values A, B: $CRC(A \oplus B) = CRC(A) \oplus CRC(B)$.

5.1 Tag/Reader Impersonation Attack

Tags are authenticated by checking the equation: $M^{Tag_i} \oplus APDW^{Tag_i} \stackrel{?}{=} CRC(TID_L^{Tag_i} \oplus RND^{Rdr} \oplus RND^{Tag_i})||CRC(TID_M^{Tag_i} \oplus RND^{Rdr} \oplus RND^{Tag_i})$. The authors claim that only genuine tags can compute correct answers to the readers's queries because private information shared between these two entities is employed in message generation. However, we show how an attacker can supplant a tag without needing to know any private information. A passive attacker, after eavesdropping one authentication session between an authentic tag ($\mathcal{T}_i$) and a genuine reader ($\mathcal{R}$), can impersonate the target tag by sending message $M_L^{Tag_i} \oplus CRC(\alpha)||M_M^{Tag_i} \oplus CRC(\alpha), RND_{new}^{Tag_i}$, where $\delta = RND_{new}^{Tag_i} \oplus RND^{Tag_i}$, $\gamma = RND_{new}^{Rdr} \oplus RND^{Rdr}$, and $\alpha = \delta \oplus \gamma$. The adversary is thus authenticated as a legitimate tag, without knowing any of the tag's private information.

Similarly, the authors point out that an attacker cannot respond to tag queries due to their ignorance of the private information and the use of fresh random nonces in each authentication session. However, we show an attacker can impersonate a reader using similar arguments as in the tag impersonation attack. On capturing messages transmitted in a valid authentication session between an authentic tag ($\mathcal{T}_i$) and a genuine reader ($\mathcal{R}$), an adversary can supplant a reader by sending message $M_L^{Rdr} \oplus CRC(\delta)||M_M^{Tag_i} \oplus CRC(\delta)$, where

$\delta = RND_{new}^{Tag_i} \oplus RND^{Tag_i}$. The tag cannot, therefore, detect the ploy and authenticates the adversary as a genuine reader.

We must emphasize that these attacks are very efficient, since passive capture of messages in just one legitimate authentication session is all that is required for their success.

5.2 Traceability Attack

The authors argue that the proposed protocol guarantees untraceability due to the use of new nonces in each session. However, in this section we show how an attacker is able to track a tag by making some simple computations over the tag's response. To do this, we use the traceability game defined in Section 3.2.

Specifically, QYY-Gen2 protocol, in an RFID system (S= $\{R_i, \mathcal{T}_0, \mathcal{T}_1, ...\}$ in which an adversary $\mathcal{A}$ can invoke one Execute($\mathcal{R}$, $\mathcal{T}$, i) and Test(i, $\mathcal{T}_0$, $\mathcal{T}_1$) query in an untraceability game $\mathcal{G}$, is vulnerable to traceability attacks, since the advantage for an adversary to win $\mathcal{G}$ is not negligible: $Adv_{\mathcal{A}}^{UNT}(t, r = 1) \simeq 0.499999 \gg \varepsilon(t, 1)$, t being a security parameter (i.e. the bit length of the access and key password) and $\varepsilon(.)$ some negligible function.

An adversary $\mathcal{A}$ performs the following steps to track tags in the QYY-Gen2 protocol:

Phase 1 (Learning): $\mathcal{A}$ sends an Execute query to $\mathcal{T}_0$. $\mathcal{A}$ acquires the random numbers used in the session and the tag's authentication message: RND^{Rdr}, $RND_q^{Tag_0}$, $M^{Tag_0} = M_L^{Tag_0} || M_M^{Tag_0}$.

$$\begin{aligned} M_L^{Tag_0} &= CRC(TID_L^{Tag_0} \oplus RND^{Rdr} \oplus RND_q^{Tag_0}) \oplus APDW_L^{Tag_0} \\ M_M^{Tag_0} &= CRC(TID_M^{Tag_0} \oplus RND^{Rdr} \oplus RND_q^{Tag_0}) \oplus APDW_M^{Tag_0} \end{aligned} \tag{12}$$

Phase 2 (Challenge): $\mathcal{A}$ chooses two fresh tags whose associated identifiers are TID^{Tag_0} and TID^{Tag_1}. Then he sends query Test(q, $\mathcal{T}_0$, $\mathcal{T}_1$). As a result, $\mathcal{A}$ is given two random number messages RND_{new}^{Rdr}, $RND_{q+1}^{Tag_i}$, and an authentication message M^{Tag_i} from the set $\{M^{Tag_0}, M^{Tag_1}\}$, which depends on a chosen random bit $b \in \{0, 1\}$:

$$M^{Tag_i} = \begin{cases} CRC(TID_L^{Tag_0} \oplus RND_{new}^{Rdr} \oplus RND_{q+1}^{Tag_0}) \oplus APDW_L^{Tag_0} || \text{ if } b{=}0 \\ \ CRC(TID_M^{Tag_0} \oplus RND_{new}^{Rdr} \oplus RND_{q+1}^{Tag_0}) \oplus APDW_M^{Tag_0} \\ CRC(TID_L^{Tag_1} \oplus RND_{new}^{Rdr} \oplus RND_{q+1}^{Tag_1}) \oplus APDW_L^{Tag_1} || \text{ if } b{=}1 \\ \ CRC(TID_M^{Tag_1} \oplus RND_{new}^{Rdr} \oplus RND_{q+1}^{Tag_1}) \oplus APDW_M^{Tag_1} \end{cases} \tag{13}$$

Phase 3 (Guessing) $\mathcal{A}$ finishes $\mathcal{G}$ and outputs a bit d ($d \in \{0, 1\}$) as its conjecture of value b. Specifically, we propose the following procedure to obtain d:

1. From Equation 12 and CRC linearity, the following constant value univocally associated with $\mathcal{T}_0$ is obtained by the adversary: $X = X_L||X_M$

$$\begin{aligned} X_L &= M_L^{Tag_0} \oplus CRC(RND^{Rdr}) \oplus CRC(RND_q^{Tag_0}) = \\ &\qquad = CRC(TID_L^{Tag_0}) \oplus APDW_L^{Tag_0} \\ X_M &= M_M^{Tag_0} \oplus CRC(RND^{Rdr}) \oplus CRC(RND_q^{Tag_0}) = \\ &\qquad = CRC(TID_M^{Tag_0}) \oplus APDW_M^{Tag_0} \end{aligned} \tag{14}$$

2. By the same mathematical reasoning, $\mathcal{A}$ calculates the constant value associated with Equation 13:

$$Y = \begin{cases} CRC(TID_L^{Tag_0}) \oplus APDW_L^{Tag_0}|| \text{ if } b=0 \\ \ CRC(TID_M^{Tag_0}) \oplus APDW_L^{Tag_0} \\ CRC(TID_L^{Tag_1}) \oplus APDW_L^{Tag_1}|| \text{ if } b=1 \\ \ CRC(TID_L^{Tag_1}) \oplus APDW_L^{Tag_1} \end{cases} \tag{15}$$

3. $\mathcal{A}$ utilizes the following simple decision rule:

$$d = \begin{cases} 0 & \text{if } X = Y \\ 1 & \text{if } X \neq Y \end{cases} \tag{16}$$

As tags are randomly initialized, there is a negligible probability for the value $CRC(TID_L^{Tag_i}) \oplus APDW_L^{Tag_i}||CRC(TID_M^{Tag_i}) \oplus APDW_L^{Tag_i}$ to be equal for different tags (i.e. $\mathcal{T}_0$ and $\mathcal{T}_1$). Specifically, for the parameters proposed by the authors, with variables set to a 32-bit length and a CRC function of 16 bits, we have a $1/2^{32}$ probability of collision, assumed independence and uniformity in the secret values $\{TID^{Tag_i}, APWD^{Tag_i}\}$ linked to each tag. Therefore the advantage of $\mathcal{A}$ in distinguishing whether the adversary interacts with $\mathcal{T}_0$ or $\mathcal{T}_1$ is: $Adv_{\mathcal{A}}^{UNT}(t,1) = |Pr[d=b] - \frac{1}{2}| = \frac{1}{2} - \frac{1}{2^{32}}$.

6 Conclusions

In this paper the cryptanalysis of two recent lightweight protocols is proposed. Both of these protocols assume that tags support an on-chip PRNG and simple operations. In Mitra's protocol, operations are limited to multiplication and addition, paving the way towards a simple differential analysis of exchanged messages, and showing its lack of resistance to cloning and traceability attacks. Additionally, a full disclosure attack, which involves all the aforemention attacks, can be implemented with a high probability of success by applying some simple results from Number Theory. Qingling et al.'s protocol is on the other hand based on a CRC function and bitwise operations as dictated by the Gen-2 specification. The security of this scheme resides in the strength provided by CRC functions. However, the authors are not aware that CRCs are linear. These functions should be confined only to detect random transmission errors and must not be used for security purposes. Qingling et al. even included a formal BAN logic analysis in their paper. The whole analysis presented is incorrect because it assumes that a secure encryption algorithm is used in the protocol. However, the use of the combination $CRC() \oplus password$ is not at all secure, as shown in Section 5.

References

1. Mitra, M.: Privacy for RFID systems to prevent tracking and cloning. International Journal of Computer Science and Network Security 8(1), 1–5 (2008)
2. Qingling, C., Yiju, Z., Yonghua, W.: A minimalist mutual authentication protocol for RFID system & BAN logic analysis. In: Proc. of CCCM 2008, pp. 449–453. IEEE Computer Society, Los Alamitos (2008)
3. Hopper, N.J., Blum, M.: Secure human identification protocols. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 52–66. Springer, Heidelberg (2001)
4. Juels, A., Weis, S.: Authenticating pervasive devices with human protocols. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 293–308. Springer, Heidelberg (2005)
5. Chien, H.Y.: SASI: A new ultralightweight RFID authentication protocol providing strong authentication and strong integrity. IEEE Trans. Dependable Secur. Comput. 4(4), 337–340 (2007)
6. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: Advances in ultralightweight cryptography for low-cost RFID tags: Gossamer protocol. In: Chung, K.-I., Sohn, K., Yung, M. (eds.) WISA 2008. LNCS, vol. 5379, pp. 56–68. Springer, Heidelberg (2009)
7. EPCglobal: Class-1 generation 2 UHF air interface protocol standard version 1.2.0: Gen 2 (2008), http://www.epcglobalinc.org/standards/
8. Chien, H., Chen, C.: Mutual authentication protocol for RFID conforming to EPC class-1 generation-2 standards. Computer Standards and Interfaces 29(2), 254–259 (2007)
9. Han, D., Kwon, D.: Vulnerability of an RFID authentication protocol conforming to EPC Class 1 Generation 2 Standards. Computer Standards and Interfaces 31(4), 648–652 (2009)
10. Lim, T., Li, T.: Addressing the weakness in a lightweight RFID tag-reader mutual authentication scheme. In: Proc. of the IEEE Int'l Global Telecommunications Conference - GLOBECOM 2007, pp. 59–63. IEEE Computer Society Press, Los Alamitos (2007)
11. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: Cryptanalysis of a novel authentication protocol conforming to EPC-C1G2 standard. Computer Standards and Interfaces 31(2), 372–380 (2009)
12. Juels, A., Weis, S.: Defining strong privacy for RFID. In: Proc. of PerCom 2007, pp. 342–347. IEEE Computer Society Press, Los Alamitos (2007)
13. Phan, R.: Cryptanalysis of a new ultralightweight RFID authentication protocol - SASI. IEEE Transactions on Dependable and Secure Computing (2008), doi:10.1109/TDSC.2008.33
14. EPCglobal: EPC Tag data standard version 1.4. (2008), http://www.epcglobalinc.org/standards/
15. Hardy, G.H., Wright, E.M.: An Introduction to the Theory of Numbers, 6th edn. Oxford University Press, Oxford (2008)
16. Anarchriz: CRC and how to reverse it (1999), http://www.woodmann.com/fravia/crctut1.htm
17. Ranasinghe, D.C.: Lightweight Cryptography for Low Cost RFID. In: Networked RFID Systems and Lightweight Cryptography, pp. 311–346. Springer, Heidelberg (2007)

Algebraic Side-Channel Attacks

Mathieu Renauld[⋆] and François-Xavier Standaert[⋆⋆]

UCL Crypto Group, Université catholique de Louvain, B-1348 Louvain-la-Neuve
{mathieu.renauld,fstandae}@uclouvain.be

Abstract. In 2002, algebraic attacks using overdefined systems of equations have been proposed as a potentially very powerful cryptanalysis technique against block ciphers. However, although a number of convincing experiments have been performed against certain reduced algorithms, it is not clear whether these attacks can be successfully applied in general and to a large class of ciphers. In this paper, we show that algebraic techniques can be combined with side-channel attacks in a very effective and natural fashion. As an illustration, we apply them to the block cipher PRESENT that is a stimulating first target, due to its simple algebraic structure. The proposed attacks have a number of interesting features: (1) they exploit the information leakages of all the cipher rounds, (2) in common implementation contexts (*e.g.* assuming a Hamming weight leakage model), they recover the block cipher keys after the observation of a single encryption, (3) these attacks can succeed in an unknown-plaintext/ciphertext adversarial scenario and (4) they directly defeat countermeasures such as boolean masking. Eventually, we argue that algebraic side-channel attacks can take advantage of any kind of physical leakage, leading to a new tradeoff between the robustness and informativeness of the side-channel information extraction.

1 Introduction

In classical cryptanalysis against block ciphers, an adversary is usually provided with the inputs/outputs of a target algorithm. Side-channel attacks additionally provide him with some partial information on the cipher intermediate values, leaked by a device performing a cryptographic computation. Such attacks are therefore much less general - since they are specific to a given implementation - but often much more powerful than classical cryptanalysis. Hence they are considered very seriously by cryptographic devices (*e.g.* smart cards) manufacturers. Following the publication of the first Differential Power Analysis (DPA) in the late nineties [14], various types of side-channel attacks have been proposed in order to carry out effective key recoveries (see, *e.g.* [17] for a survey). Most of these techniques share a divide-and-conquer strategy in which different parts of a target key (*e.g.* physical bytes, typically) are recovered separately. They also generally exploit the leakages corresponding to the first (or last) rounds of a block cipher, where the diffusion is sufficiently weak for some parts of the intermediate key-dependent computations to be easily enumerated and predicted.

[⋆] Work supported in part by the Walloon Region research project SCEPTIC.
[⋆⋆] Associate researcher of the Belgian Fund for Scientific Research (FNRS - F.R.S.).

As a matter of fact, these side-channel attacks are quite demanding in leaked information since they use physical measurements to identify key bytes exactly. Also, they usually do not exploit particular weaknesses of the block ciphers. Therefore, an intriguing question is to know if an adversary could use side-channels to recover simple targets rather than exact key byte values and then use this partial information in a more elaborated offline cryptanalysis step. In other words, can we stop measuring earlier and still have the complexity of a key recovery that does not grow exponentially with the key size?

In this paper, we answer this question positively and show that combining powerful (template-like) side-channel attacks with algebraic cryptanalysis allows performing key recoveries with extremely restricted access to the target devices. Still, the attack is general and can work in a flexible manner that includes the two following phases. First, the adversary selects as many intermediate computations in the target algorithm as possible and measures their physical leakage. For each of these intermediate computations, he recovers some partial information. This partial information can be represented by a surjective function of which the output has been recovered thanks to a side-channel attack. As a typical example, if a device leaks an information that is strongly correlated with the Hamming weight of the target intermediate computations results, this function could be the Hamming weight function. But any other type of function (and hence leakage model) could be considered. It is eventually the adversary's choice to select a target that is both informative and robust. At the extremes, a bijective function is the most informative but recovering its output by a side-channel attack may require several measurements - and a surjective function with only one possible output value yields no information at all. Then, during a second (offline) phase, the adversary exploits this partial information on the algorithm intermediate values with an algebraic attack. That is, he writes the block cipher as a system of quadratic (or cubic, ...) equations and adds the previously defined functions with known outputs to the system. In practice, the approach we follow in this paper is to convert the system of equations representing the block cipher into a SAT problem and to use an automated solver to perform the key recoveries. It turns out that this solution yielded very good results. However, it remains an open question to determine better techniques for this purpose.

The proposed attacks differ from most previously known side-channel attacks in a number of interesting aspects. First, they potentially exploit the leakage of all the cipher rounds (classical DPA generally exploits the first or last rounds only). Second, they can succeed in an unknown plaintext/ciphertext adversarial context (classical DPA usually requires the knowledge of either the plaintexts or the ciphertexts). Third, they require much less observations to succeed. In certain reasonable implementation contexts, we show that the leakage trace of a single encryption can be sufficient to perform a complete key recovery. This implies that constructions based on re-keying strategies such as [20,21] can sometimes be broken in practice. Eventually, they can deal with block ciphers protected with a masking countermeasure, *e.g.* [12]. In particular, we show experiments that break such masked designs with a single trace, nearly as easily as unprotected ones.

In summary, classical side-channel attacks can be viewed as a combination of two sub-problems: (1) "how to efficiently recover partial information on certain parts of a cipher state?" and (2) "how to efficiently exploit this partial information?". Algebraic side-channel attacks raise a new question, namely: "which partial information should we try to recover?". It relates both to the previously mentioned tradeoff between robustness and informativeness and to the selection of the best target key classes mentioned as an open question in [28].

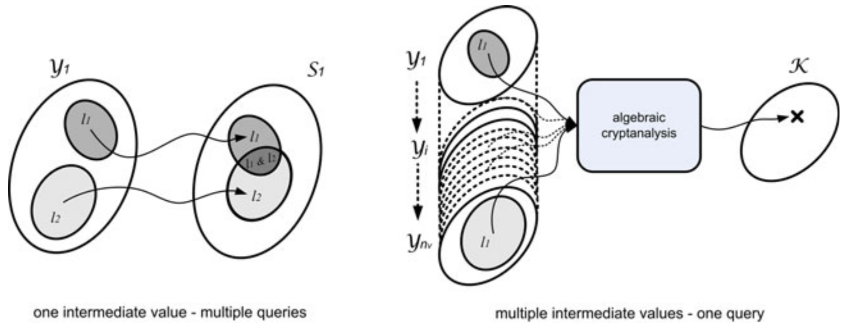


Fig. 1. Standard DPA versus algebraic side-channel attacks

More precisely, the difference between algebraic side-channel attacks and standard DPA attacks is illustrated in Figure 1. As already mentioned, standard DPA attacks exploits a divide-and-conquer strategy and recover several pieces of a secret key independently. For example, in the left part of the figure, the set $\mathcal{S}_1$ typically contains the 256 candidates for a key byte. In order to recover the correct one, the adversary targets a single intermediate value (in the set $\mathcal{Y}_1$). Typically, it could be the output of an S-box in the first cipher round. Each leakage trace l_i provides him with information about this intermediate value that is then "translated" into subkey information. By combining the leakage corresponding to several plaintexts (*i.e.* by increasing the data complexity q), he finally identifies the key byte exactly. By contrast, an algebraic side-channel attack aims to limit the data complexity to $q = 1$ and exploits several (n_v) intermediate values within a single leakage trace. This information is then combined in an offline cryptanalysis step in order to recover the master key at once. Note that the data complexity is not always equivalent to the number of measurements since the same leakage trace can be measured several (n_r) times. But the data complexity is the most relevant quantity to compare from a cryptanalytic point of view, in particular regarding constructions such as [20,21].

Related works. The following results can be related to three different lines of research. First, they aim to recover partial information from a leaking device in the most efficient way. They consequently exploit techniques such as template attacks (*e.g.* [9,29]) and stochastic models [25]. Second, they take advantage of algebraic cryptanalysis in the black box setting, introduced by Courtois and

Pieprzyk in [10]. In particular, we exploit solutions based on SAT solvers as described in [1,11]. Eventually, several other papers suggested to combine side-channel attacks with classical cryptanalysis. The most studied problem is probably the one of collision-based side-channel attacks, detailed *e.g.* in [15,26,27]. Techniques borrowed from square attacks [7] and differential cryptanalysis [13] against block ciphers have also been proposed in 2005 and 2006, respectively. More recently, impossible and multiset collision attacks were presented at CHES 2007 [3]. All these attacks have objectives similar to ours. They usually try to exploit the information leakages for more than the first block cipher rounds with advanced cryptanalysis. The goal is to break implementations for which only those rounds would be protected against side-channel attacks or to reduce the number of measurements required to perform a key recovery. We finally mention the recent and very efficient collision-based attacks of [6] that also use algebraic techniques and therefore closely connect to the present paper. In fact, our proposed cryptanalysis can be viewed as a generalization of such collision-based attacks. We similarly aim to reduce the data complexity ([6] found $4 \leq q \leq 20$, we claim $q = 1$). The main difference is that we are not restricted to one particular type of information (*i.e.* collisions) and are not limited to the exploitation of the first/last rounds of a block cipher. In principle, our algebraic attacks can take advantage of any information leakage, from any part of a cryptographic computation. A consequence is that they can be easily extended to protected implementations (*e.g.* masked), contrary to collision-based ones [5].

2 Target Cipher

PRESENT is a Substitution-Permutation Network with a block size of 64 bits [4]. The recommended key size is 80 bits, which should be sufficient for the expected applications of the cipher. However a 128-bit key-scheduling is also proposed. The encryption is composed of 31 rounds. Each of the 31 rounds consists of a XOR operation to introduce a round key K_i for $1 \leq i \leq 32$, where K_{32} is used for post-whitening, a linear bitwise permutation and a non-linear substitution layer. The non-linear layer uses a single 4-bit S-box which is applied 16 times in parallel in each round. The cipher is described in pseudo-code in Appendix A.

3 Offline Phase: Algebraic Attack

3.1 Deriving the System of Equations

The first step in an algebraic cryptanalysis is to describe the target cryptosystem with a set of polynomial equations involving the key bits as variables. Given such a representation, exposing the secret key is equivalent to solving the system of equations. For this purpose, we will denote the bits of the plaintext, ciphertext and key as P_i, C_i and K_i and look for a set of equations involving these variables and describing the cryptosystem PRESENT. The most obvious solution would be to build a system of equations of the form:

$$C_1 = f_1(P_1, ..., P_{64}, K_1, ..., K_{80})$$
$$C_2 = f_2(P_1, ..., P_{64}, K_1, ..., K_{80})$$

$$C_{64} = f_{64}(P_1, ..., P_{64}, K_1, ..., K_{80})$$

However, this kind of representation is quite useless for practical attacks. Due to the diffusion in the cryptosystem, each equation would involve every single bit of the plaintext and the key. Due to the 31 successive rounds of non-linear substitutions, these equations would also include a lot of high degree monomials. In order to avoid such limitations, the idea developed by Courtois and Pieprzyk in [10] is to introduce new internal variables in order to work with a large number of small, low degree polynomial equations instead of a small number of huge, high degree equations. For PRESENT, we decided to add three groups of variables:

– one variable x_i for each input bit of each S-box in the cryptosystem,
– one variable y_i for each output bit of each S-box in the cryptosystem,
– one variable k_i for each bit of each sub-key.

In practice, the substitutions are the only non-linear elements of PRESENT and are therefore the most challenging parts of the cipher to reduce to low degree equations. Fortunately, it has been shown in [2] that for small S-boxes, such equations can be constructed in a simple and systematic manner. As an illustration, the construction of a system of low degree equations for a 3-bit S-box is described in Appendix B. Extending this technique to the complete 31-round PRESENT, we can build a system of approximately 40 000 equations in 7000 variables (50 000 monomials). The most interesting characteristic of this system is its sparsity. If we represent such a system like a matrix, a line being an equation, a column being a monomial, the proportion of non-null elements is very low (approximately 0.0155%). Using a compact representation of the system matrix consequently improves the attack performances considerably.

3.2 Conversion to a SAT Problem

Bard *et. al* showed in [1] how to reduce a system of equations to a SAT problem. Most SAT solvers take a formula in *conjunctive normal form* (CNF) as input. A problem in CNF is a conjunction (AND) of clauses, each clause being a disjunction (OR) of literals. The literals are variables (x) or variable negations ($\bar{x}$).

In order to reduce our problem, there are two main steps. First, we need to translate every monomial of degree higher than 1. If we consider a monomial $x_1x_2x_3x_4$, we can turn it into a dummy variable a and a set of clauses:

$$(x_1 \vee \bar{a}) \wedge (x_2 \vee \bar{a}) \wedge (x_3 \vee \bar{a}) \wedge (x_4 \vee \bar{a}) \wedge (a \vee \bar{x_1} \vee \bar{x_2} \vee \bar{x_3} \vee \bar{x_4}), \qquad (1)$$

that is equivalent to $a = x_1x_2x_3x_4$. Hence we can transform each occurrence of the monomial $x_1x_2x_3x_4$ into an occurrence of the dummy variable a in the system of equations and include the previous set of clauses in a CNF. So, for each monomial of degree $d > 1$, we introduce one dummy variable and $d + 1$ clauses. Secondly, we need to translate the exclusive disjunctions (XOR) of our original equations into conjunctions and disjunctions. Translating long XOR-equations in conjunctive normal form is problematic because the number of new clauses is exponential in the number of terms in the equation. Hence, we use again dummy variables in order to bound the number of new clauses in the formula. We transform each equation $x_1 \oplus x_2 \oplus x_3 \oplus ... \oplus x_n = 0$ into:

$$\begin{aligned} x_1 &\oplus x_2 \oplus x_3 \oplus b_1 = 0 \\ b_1 &\oplus x_4 \oplus x_5 \oplus b_2 = 0 \\ &... \\ b_m &\oplus x_{n-1} \oplus x_n = 0 \end{aligned}$$

This way, we separate each n-term equation into an equivalent set of $m = \lceil n/2 \rceil - 1$ (for $n > 2$) 4-term equations[1], via the addition of m dummy variables. After that, each 4-term equation of the form $a \oplus b \oplus c \oplus d$ is turned into an equivalent set of 8 clauses that we add to the previously initiated CNF:

$$\begin{aligned} (\bar{a} \vee b \vee c \vee d) &\wedge (a \vee \bar{b} \vee c \vee d) \wedge (a \vee b \vee \bar{c} \vee d) \wedge (a \vee b \vee c \vee \bar{d}) \wedge \\ (\bar{a} \vee \bar{b} \vee \bar{c} \vee d) &\wedge (\bar{a} \vee \bar{b} \vee c \vee \bar{d}) \wedge (\bar{a} \vee b \vee \bar{c} \vee \bar{d}) \wedge (a \vee \bar{b} \vee \bar{c} \vee \bar{d}) \end{aligned}$$

Combining these two steps, we can build a CNF formula from our system of equations. We can estimate the number of different literals in the formula: $n_{literal} \simeq n_{mon} + n_{equ} * (\lceil n_{term}/2 \rceil - 1)$, were n_{mon} and n_{equ} are respectively the number of monomials and equations in the system and n_{term} is the average number of terms in an equation (typically quite low, because of the sparsity of the system). We can similarly estimate the number of clauses: $n_{clause} \simeq n_{mon} * (d + 1) + n_{equ} * (\lceil n_{term}/2 \rceil - 1) * 8$, where d is the average degree of the monomials appearing in the system. Interestingly, the size of the CNF and the number of literals are linearly dependent in most of the cryptosystem parameters (block and key size, number of rounds). In fact, only the size of the S-boxes has more impact, because it not only modifies n_{mon} and n_{equ} but also d and n_{term}.

3.3 Solving the System

We selected zChaff, a SAT solver that was developed by Princeton University [8] and won the 2004 SAT competition [24]. This solver is not the best SAT solver available anymore, but it works fine for our experiments. zChaff uses the Chaff algorithm, which is a refinement of the DPLL algorithm [19]. These algorithms use a recursive backtracking procedure to find an adequate solution [18]. In summary, at each step s the procedure assigns a random value (say FALSE) to a literal x_{i_s} and simplifies the formula. If no conflict (empty clause) is detected, the procedure is repeated for the next step $s+1$. If one or more conflicts are detected, the procedure backtracks: the formula is restored as it was before assigning x_{i_s}.

1 Using a cutting number of 4 is arbitrary but yielded satisfactory results in our context. 5-term, 6-term, ... equations could be similarly investigated.

Then it assigns the opposite value (TRUE) to x_{i_s} and continues as previously. If both values (TRUE and FALSE) were already tried for x_{i_s}, the procedure backtracks to $x_{i_{s-1}}$, and so on. If the procedure assigns a value to all the literals and finds no conflicts, the problem is declared satisfiable. If the procedure must but cannot backtrack ($s = 0$), the problem is declared unsatisfiable. In practice, solving the system of equations described in Sections 3.1, 3.2 with the previous SAT solver is generally hard. Therefore, the idea we propose in this paper is to take advantage of the additional information provided by side-channel leakages on the intermediate values during the execution of PRESENT. The next section describes the online part of this attack. We explain how a leakage model that can be efficiently exploited in an algebraic cryptanalysis was selected and constructed for a given device. In addition, we discuss the generalization to other leakage models and the resulting information *vs.* robustness tradeoff in Section 6.

4 Online Phase: Side-Channel Attacks

Our experiments target an implementation of PRESENT with an 80-bit key in a PIC 16F877 8-bit RISC-based micro-controller and exploit the measurement setup described in [29]. This target device is particularly convenient for a first investigation since it typically leaks a power consumption that is strongly correlated with the Hamming weight of the data it manipulates. For example, the left part of Figure 2 illustrates the power consumption corresponding to different 8-bit values commuting on the PIC bus, having different Hamming weights: the bold traces represent the mean power consumption for a given Hamming weight between 0 and 8; the grey traces represent single measurements. This picture visually suggests that the Hamming weight of the data commuting on the bus can be recovered with very high confidence in a single trace. In practice, we used a Bayesian template attack such as described in [9] to perform a partial key recovery for which the target subkey is the Hamming weight of a data commuting on the bus. That is, we have $|\mathcal{S}| = 9$ (rather than $|\mathcal{S}| = 256$ in standard DPA attacks). In this setting, we experimented a single-byte success rate (defined in Appendix C) of 99.3%. That is, given a leakage sample corresponding to some 8-bit byte x, we can recover $W_H(x)$ with probability 0.993[2].

Importantly and contrary to most other side-channel attacks, algebraic side-channel attacks do not only exploit the leakage corresponding to one byte of the intermediate cipher state at once. On the contrary, they aim to exploit as much partial information about these intermediate values as possible. Hence, our attacks exploit a powerful profiling step such that the adversary recovers the leakages corresponding to the computation of $2 \times 8 \times 31 = 496$ bytes during the encryption of a single plaintext. Those bytes relate to the computation of the key additions (8 bytes per round) and the S-boxes (8 bytes per round) for all the 31 cipher rounds. As for any profiling step, this requires that the adversary can manipulate a device with a known key prior to the attack. But once this profiling is performed, it can be re-used for as many online attacks as possible.

[2] Improved techniques such as [29] could be used in more critical contexts.

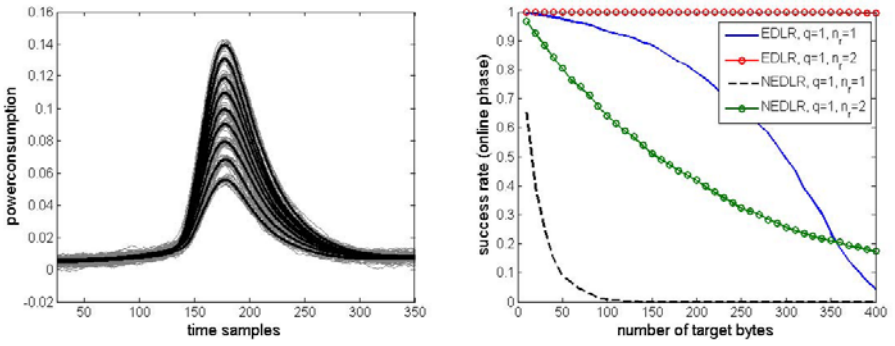


Fig. 2. Leakage traces, mean leakage traces and multiple byte success rate

Because a SAT solver can hardly deal with errors in the information it is provided with, the important quantity in our attacks is not the single-byte success rate but the multiple-byte success rate. In practice, it can be improved by using simple error detection and likelihood rating techniques such as:

– Detection of impossibilities: we systematically rejected the leakages samples that give rise to incoherent inputs and outputs for the S-boxes.
– Selection of most likely Hamming weights: when using only a subset of the 496 available leakages, we used the ones with highest probabilities first.

In addition and when necessary, the success rate of our attacks can be increased by repeating a measurement for the same input plaintext, therefore keeping a constant data complexity $q = 1$. The right part of Figure 2 represents the multiple-byte success rate as a function of the number of target bytes in the implementation of PRESENT, with or without error detection and likelihood rating (EDLR), and for $q = 1$ and $n_r = 1, 2$. As a matter of fact, the complexity of this online phase depends on how many bytes of information are required to solve the system of equations given in Section 3.1, 3.2. Yet, it remains that for our target device, it is easier to recover the Hamming weight of a byte than the exact value of this byte, which is the main motivation behind our attack.

We mention that we did not consider side-channel leakages from the key scheduling of PRESENT (*i.e.* we assumed implementations with securely precomputed round keys). This allows avoiding the algebraic counterparts of simple power analyzes such as [16] that would trivially break the implementation. In other words, we focused our attention on the more challenging scenarios where only the cipher rounds are leaking. Note that although the leakage of the key scheduling algorithm is not exploited in our attacks, its algebraic description is included in the system of equations representing PRESENT.

5 Combining Algebraic and Side-Channel Attacks

Following the previous section, the partial information provided by side-channel leakages can be represented by a surjective function of which the output is known.

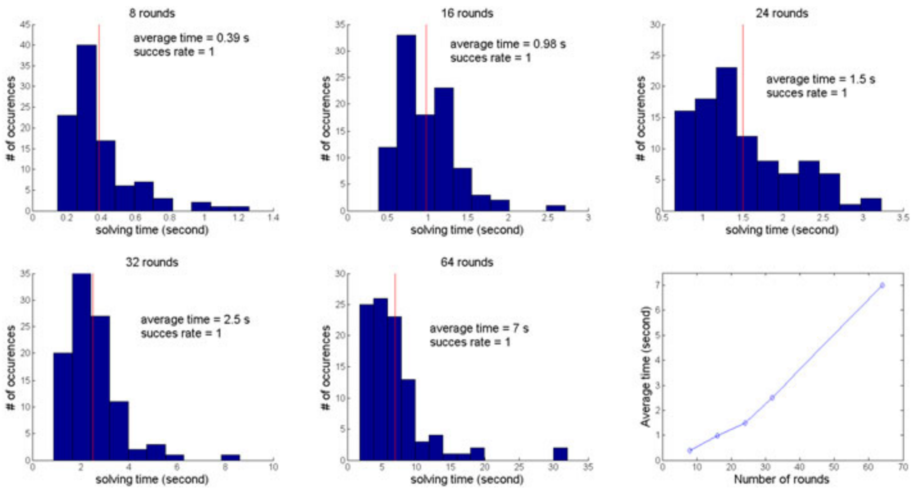


Fig. 3. 8, 16, 24, 32 and 64-round PRESENT, complete W_H leakages

Given a leakage model and a number of target bytes, an adversary can directly inject this additional information into the system of Section 3.1, or in its CNF representation (since the leakage information can be converted into a set of clauses). In practice, we exploited the Hamming weights of the key addition and S-box outputs in PRESENT. As mentioned in Section 2, a maximum of 496 bytes can be extracted. It corresponds to a SAT problem for a 31-round PRESENT that includes approximately 130 000 variables and 1 100 000 clauses.

5.1 First Experimental Results

We first applied algebraic side-channel attacks assuming that all the Hamming weights of the S-box inputs and outputs in PRESENT are correctly extracted. For comparison purposes, we attacked different reduced-round and extended-round versions of the algorithm. The results of these attacks are summarized in Figure 3 from which we obtain the following observations:

1. The success rate of these attacks equals one for all versions of PRESENT.
2. Hence, the important quantity to analyze is the resolution time which seems to follow an exponential probability distribution.
3. The average resolution time is approximately linear in the number of rounds.

5.2 Advanced Scenarios

The experiments in the previous section are quite disconnected from practical attacks since we assume that all Hamming weights are recovered. In practice, the subkey extraction may suffer from errors and consequently, only a subset of the 496 bytes Hamming weights in a 31-round PRESENT can be exploited. In this section, we consider advanced scenarios where only parts of the 496 bytes are targeted. Specifically, we consider two types of contexts: (1) consecutive weights, *i.e.* the adversary recovers a number of Hamming weights that correspond to consecutive rounds in the block cipher, starting from the middle ones and (2) random weights, *i.e.* the adversary recovers Hamming weights that correspond to random bytes in the cipher rounds. We measure the amount of information extracted in "number of rounds of W_H information", one round corresponding to the Hamming weights of 16 intermediate bytes. And we consider that an attack has failed whenever the solver has not found a solution within 3600 seconds.

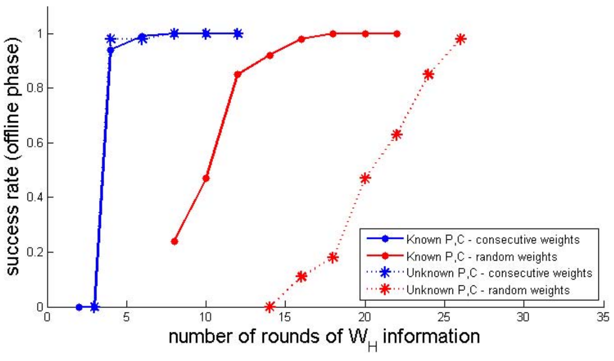


Fig. 4. 31-round PRESENT, partial W_H leakages and unknown P,C (the average solving times for these advanced scenarios are available in Appendix D)

The plain curves in Figure 4 correspond to this more realistic scenario. They illustrate that reaching high success rates is significantly more difficult with random weights. It also shows that recovering the Hamming weights of 4 consecutive rounds is generally sufficient to solve the system of equations. It is interesting to trade the effectiveness of this offline algebraic phase with the one of the online phase in Section 4. Indeed, taking advantage of the error detection and likelihood rating techniques implies a non consecutive selection of weights.

Unknown plaintexts-ciphertexts. In addition, the same figure shows the success rates when considering attacks in an unknown plainext and ciphertext adversarial scenario. Quite naturally, it does not affect the results when consecutive rounds are considered. But unknown plaintexts/ciphertexts imply the need of larger amounts of information when this information is randomly distributed over the block cipher intermediate values. It is worth noting that most side-channel attacks (*e.g.* Kocher's original DPA [14]) would fail in a similar context.

Masked implementations. We also performed experiments against an implementation protected with a masking countermeasure. In short, masking a software implementation aims to trade the efficiency and cost of this implementation for an improved security against side-channel attacks. In general, the most challenging part to mask in a block cipher is the non-linear S-box. That is, assuming a masked plaintext $p \oplus m$, we need to generate an output mask q for the S-box $\mathsf{S}(p \oplus k)$. For example, [17] describes a masked software implementation of the AES in Section 9.2.1. In order to limit the cost overheads, the same pair of masks (m, q) is used for all the AES S-boxes and in all the AES rounds (4 other masks are used in the MixColumn operation). Applying this solution to PRESENT would yield very little additional variables for our algebraic attacks. Hence, we considered a more robust countermeasure (algebraically speaking) denoted as duplication in [12] or GLUT in [22] in which a different mask is used for all the S-boxes and is propagated through the rounds as in Figure 5: an S-box S' is then used to generate the output masks q. Since PRESENT has 4-bit S-boxes, this is feasible at a reasonable cost (the memory requirements of S' equals $2^8 \times 4$).

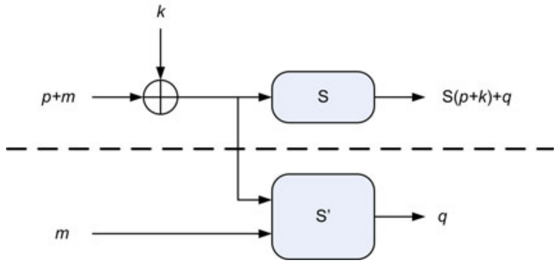


Fig. 5. Masked implementation of PRESENT

Interestingly, two different strategies can be considered to break this masked implementation. A simple one is to just neglect the masks and solve the system as if it had unknown plaintext and ciphertext (*i.e.* to consider only the upper part of Figure 5). A more elaborated solution is to build a system of equations including the new S-boxes S' and to solve it with known plaintext and ciphertext. We performed both solutions and the results are summarized in Figure 6. These attacks show that even a masked implementation can be broken with the observation of a single encrypted plaintext in our experimental setting. They also confirm the intuition of the previous section that unknown plaintext/ciphertext only increases the difficulty of recovering the block cipher key if random weights are provided to the adversary. In this context, building a more complex system including the mask propagation leads to better results. It is worth noting that when building a complete system, more Hamming weights are also extracted per round. Eventually, we mention that refreshing the masks in every round would not improve the security either, since the combined information on $p \oplus m \oplus k$ and $p \oplus m \oplus m' \oplus k$ would allow to easily break the masking too.

Global success rate. Eventually, the previous success rates have been computed for the offline phase of our attacks only. But we can similarly compute the global success rate, combining the ones of the online side-channel attack and the

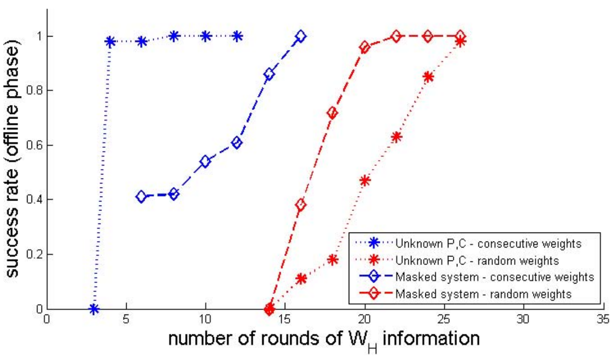


Fig. 6. Masked 31-round PRESENT, partial W_H leakages and unknown P,C

algebraic cryptanalysis. For example, Figure 7 presents this global success rate for an unprotected implementation, in a known plaintext/ciphertext scenario and with randomly distributed leakages. It illustrates the tradeoff between the two phases of the attack. Recovering more Hamming weights increases the probability of mistake for one of them - but if all correct, they significantly facilitate the offline computations. We mention that in this figure, we assume that an incorrect Hamming weight leads to a failure. But advanced strategies could be considered, by better dealing with incorrect information (*e.g.* extracting sets of Hamming weights including the correct one rather than exact Hamming weights).

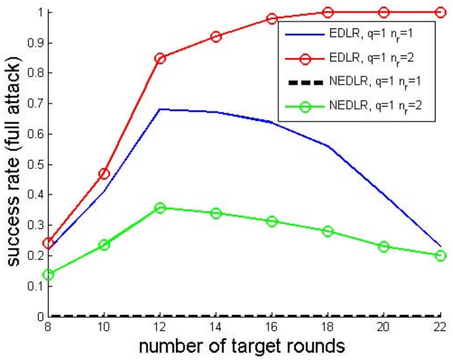


Fig. 7. Global success rate, randomly distributed W_H and known P,C

6 Information *vs.* Robustness Tradeoff

Although the construction of a leakage model and the selection of a target subkey in Section 4 appears natural for our running device, they are in fact arbitrary to a certain extent. Indeed, an adversary can generally choose any leakage model as soon as this model fits reasonably well to the actual measurements. And given a leakage model, he still has to decide which subkey to recover. In the previous section and given a byte x commuting on the PIC bus, we decided to guess the value of the Hamming weight of this byte (which is motivated by reasonable physical assumptions). But another adversary could decide to guess the exact value of x (a far more challenging goal) or one bit of x. The selection of a target subkey with respect to a leakage model gives rise to a tradeoff between the robustness and informativeness of the partial key recovery that we now discuss.

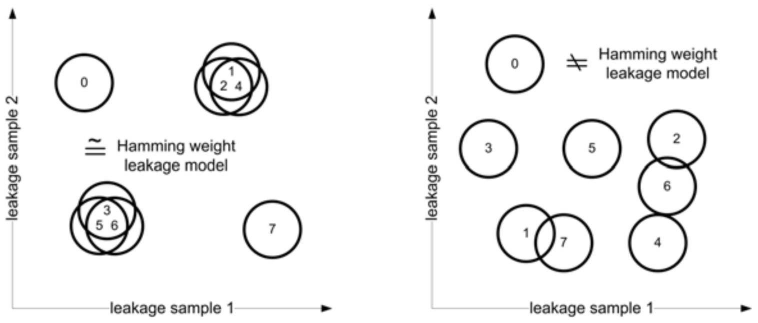

leakage sample 2
leakage sample 1
≅ Hamming weight leakage model
≇ Hamming weight leakage model
0
1
2
3
4
5
6
7

Fig. 8. Two illustrative leakage probability density functions

As an illustration, let us imagine two leaking device with a 3-bit ALU and bus. Let us also assume that these devices leak two normally distributed samples as intuitively represented in Figure 8. That is, each of the 8 possible values that can appear on the device bus gives rise to a leakage represented by a two-dimensional Gaussian curve (*i.e.* a circle on the figure). From this example, it quite directly appears that a good hypothetical leakage model for the left leakage function would be a Hamming weight-like function. By contrast, the right leakage function does not show such Hamming weight dependencies (since values with identical Hamming weights do not give rise to similar leakages). In both cases, it is the adversary's choice to select a target subkey that is easy to recover and informative. Recovering Hamming weights is usually relevant for standard CMOS devices. But in general, one can decide to extract any type of information from the leakages. And the better the target subkey actually corresponds to the actual leakages, the more robust and efficient the subkey recoveries. Of course and on the contrary, more informative subkeys generally give rise to an easier solving for the system of equations described in Section 3.

In summary, classical DPA attacks usually directly target key bytes in block ciphers and this requires to combine the leakages corresponding to several encrypted plaintexts. Algebraic side-channel attacks allow focusing on easier targets and can exploit any information on the intermediate values of a block cipher that physical leakages may effectively determine. This information can be easily represented by a surjective function from the target intermediate value space (*i.e.* $\{0,1\}^8$ in our example) to an hypothetical leakage space (*i.e.* $\{0,1,\ldots 9\}$ in our example). It is then the goal of the SAT solver to find if the extracted information is sufficient to perform a complete key recovery from it. The answer depends on the quality of the solver, the specifications of the block cipher and the quality of its implementation. Hence, algebraic side-channel attacks raise a new research problem, namely: “what is the smallest amount of information that a side-channel attack has to provide to break a block cipher?” In Section 4, we show that finding the Hamming weights of the intermediate bytes in PRESENT is sufficient. But is is an open question to determine if other types of information (*e.g.* obtained from for dual-rail circuits) can be sufficient.

7 Conclusions and Open Problems

This paper introduces algebraic side-channel attacks allowing to successfully recover a block cipher key with the observation of a single side-channel trace. While most side-channel attacks can be generically applied independently of the algorithms, the proposed technique takes advantage of cryptanalytic tools that are dependent both on the amount of physical information leakage and the structure of the target cipher. It trades some of the flexibility of standard DPA for a reduced data complexity. Still, it remains very flexible, since given a system of equations describing a block cipher (or a masked block cipher), any type of physical information can in principle be exploited in a straightforward manner. These results raise two main open questions. A first one is to prevent algebraic side-channel attacks. Intuitively, this would require to increase the algebraic complexity of both the target algorithms and the information leakages. Experiments performed against the AES Rijndael (see [23]) suggest that moving to more elaborated ciphers than PRESENT is not sufficient. Hence, additionally moving from 8-bit platforms where the Hamming weight leakages are very informative to larger devices is an interesting direction for increasing the security of cryptographic devices. The impact of former countermeasures against DPA in this new context (*e.g.* time randomizations) should also be investigated. A second question is to improve and optimize the different parts of an algebraic side-channel attack. It implies to study both the offline cryptanalytic aspects and the selection of good leakage models. Properly combining these two phases would allow determining the best tradeoff between the robustness and informativeness of a side-channel attack that is discussed in Appendix 6.

References

1. Bard, G., Courtois, N., Jefferson, C.: Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers. In: Cryptology ePrint Archive, Report 2007/024 (2007)
2. Biryukov, A., De Cannière, C.: Block Ciphers and Systems of Quadratic Equations. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 274–289. Springer, Heidelberg (2003)

3. Biryukov, A., Khovratovich, D.: Two New Techniques of Side-Channel Cryptanalysis. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 195–208. Springer, Heidelberg (2007)
4. Bogdanov, A., Knudsen, L., Leander, G., Paar, C., Poschmann, A., Robshaw, M., 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)
5. Bogdanov, A.: Improved Side-Channel Collision Attacks on AES. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 84–95. Springer, Heidelberg (2007)
6. Bogdanov, A., Kizhvatov, I., Pyshkin, A.: Algebraic Methods in Side-Channel Collision Attacks and Practical Collision Detection. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 251–265. Springer, Heidelberg (2008)
7. Carlier, V., Chabanne, H., Dottax, E., Pelletier, H.: Generalizing Square Attack using Side-Channels of an AES Implementation on an FPGA. In: The Proceedings of FPL 2005, Tampere, Finland, pp. 433–437 (August 2005)
8. http://www.princeton.edu/~chaff/
9. Chari, S., Rao, J., Rohatgi, P.: Template Attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
10. Courtois, N., Pieprzyk, J.: Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)
11. Courtois, N., Bard, G.: Algebraic Cryptanalysis of the Data Encryption Standard. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007. LNCS, vol. 4887, pp. 152–169. Springer, Heidelberg (2007)
12. Goubin, L., Patarin, J.: DES and Differential Power Analysis. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
13. Handschuh, H., Preneel, B.: Blind Differential Cryptanalysis for Enhanced Power Attacks. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 163–173. Springer, Heidelberg (2007)
14. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
15. Ledig, H., Muller, F., Valette, F.: Enhancing Collision Attacks. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 176–190. Springer, Heidelberg (2004)
16. Mangard, S.: A Simple Power Analysis (SPA) Attack on Implementations of the AES Key Expansion. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 343–358. Springer, Heidelberg (2003)
17. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks. Springer, Heidelberg (2007)
18. Mitchell, D.: A SAT Solver Primer. In: The Proceedings of EATCS Bulletin, The Logic in Computer Science Column, vol. 85, pp. 112–133 (2005)
19. Moskewicz, M., Madigan, C., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an Efficient SAT Solver. In: The Proceedings of DAC 2001, Las Vegas (June 2001)
20. Petit, C., Standaert, F.-X., Pereira, O., Malkin, T.G., Yung, M.: A Block Cipher based PRNG Secure Against Side-Channel Key Recovery. In: The Proceedings of ASIACCS 2008, Tokyo, Japan, pp. 56–65 (March 2008)
21. Pietrzak, K.: A Leakage-Resilient Mode of Operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2010)

22. Prouff, E., Rivain, M.: A Generic Method for Secure S-box Implementation. In: Kim, S., Yung, M., Lee, H.-W. (eds.) WISA 2007. LNCS, vol. 4867, pp. 227–244. Springer, Heidelberg (2008)
23. Renauld, M., Standaert, F.-X., Veyrat-Charvillon, N.: Algerbaic Side-Channel Attacks on the AES: Why Time also Matters in DPA. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 97–111. Springer, Heidelberg (2009)
24. SAT 2004 competition main page, http://www.lri.fr/~simon/contest04/results/
25. Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side-Channel Cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005)
26. Schramm, K., Wollinger, T.J., Paar, C.: A New Class of Collision Attacks and Its Application to DES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 206–222. Springer, Heidelberg (2003)
27. Schramm, K., Leander, G., Felke, P., Paar, C.: A Collision-Attack on AES: Combining Side Channel and Differential Attack. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 163–175. Springer, Heidelberg (2004)
28. 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 (2010)
29. Standaert, F.-X., Archambeau, C.: Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 411–425. Springer, Heidelberg (2008)

A Pseudo-Code of PRESENT

```
P' ← P
Round key 1 ← K
for $i = 1$ to 31 do
P' ← P' ⊕ Round key $i$
P' ← Substitution(P')
P' ← Permutation(P')
Round key $i+1$ ← Update(Round key $i$)
end for
C ← P' ⊕ Round key 32
```

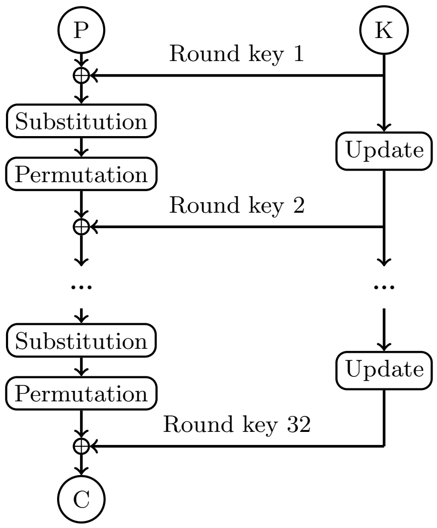


Fig. 9. A top-level algorithmic description of PRESENT

B Building the System of Equations for a 3-Bit S-Box

Let us consider the 3-bit to 3-bit substitution defined by the lookup table : $\mathsf{S} = \{5, 3, 0, 4, 7, 2, 6, 1\}$, with input bits x_1, x_2, x_3 and output bits y_1, y_2, y_3 (the most significant bits being x_1 and y_1). First, one has to decide the monomials that will appear in the equations, which significantly impacts their complexity. In our example, we choose the monomials x_i, y_i, and x_iy_j ($\forall\ 1 \leq i, j \leq 3$) so that we eventually have 16 monomials (including the independent term). Then a 16×2^3 matrix is built in which each row represents a monomial and each column represents a possible input for the substitution. The matrix is filled as follows: the element (i, j) is the value of the monomial i if the substitution input is j. Finally, we perform a Gaussian elimination on the matrix : we add and swap the rows until the matrix becomes upper triangular (*i.e.* all the elements below the diagonal are null). In our example, we obtain 8 linearly independent combinations of monomials which are null for every possible input of the substitution, as illustrated by the following matrices:

$$\begin{array}{c|cccccccc|}
1 & 1&1&1&1&1&1&1&1\\
x_1 & 0&0&0&0&1&1&1&1\\
x_2 & 0&0&1&1&0&0&1&1\\
x_3 & 0&1&0&1&0&1&0&1\\
y_1 & 1&0&0&1&1&0&1&0\\
y_2 & 0&1&0&0&1&1&1&0\\
y_3 & 1&1&0&0&1&0&0&1\\
x_1y_1 & 0&0&0&0&1&0&1&0\\
x_1y_2 & 0&0&0&0&1&1&1&0\\
x_1y_3 & 0&0&0&0&1&0&0&1\\
x_2y_1 & 0&0&0&1&0&0&1&0\\
x_2y_2 & 0&0&0&0&0&0&1&0\\
x_2y_3 & 0&0&0&0&0&0&0&1\\
x_3y_1 & 0&0&0&1&0&0&0&0\\
x_3y_2 & 0&1&0&0&0&1&0&0\\
x_3y_3 & 0&1&0&0&0&0&0&1
\end{array}
\Longrightarrow
\begin{array}{c|cccccccc|}
1 & 1&1&1&1&1&1&1&1\\
x_3 & 0&1&0&1&0&1&0&1\\
x_2 & 0&0&1&1&0&0&1&1\\
x_3+y_2 & 0&0&0&1&1&0&1&1\\
x_1 & 0&0&0&0&1&1&1&1\\
1+x_2+y_3 & 0&0&0&0&0&1&0&1\\
1+x_2+x_3+y_1 & 0&0&0&0&0&0&1&1\\
x_1+x_1y_2 & 0&0&0&0&0&0&0&1\\
1+x_1+x_2+y_3+x_1y_1 & 0&0&0&0&0&0&0&0\\
x_1+x_3+y_1+y_3+x_1y_3 & 0&0&0&0&0&0&0&0\\
x_1+y_1+y_2+y_3+x_2y_1 & 0&0&0&0&0&0&0&0\\
1+x_1+x_2+x_3+y_1+x_1y_2+x_2y_2 & 0&0&0&0&0&0&0&0\\
x_1+x_1y_2+x_2y_3 & 0&0&0&0&0&0&0&0\\
1+x_2+x_3+y_2+y_3+x_1y_2+x_3y_1 & 0&0&0&0&0&0&0&0\\
1+x_1+x_2+y_2+y_3+x_3y_2 & 0&0&0&0&0&0&0&0\\
x_1+y_2+x_3y_3 & 0&0&0&0&0&0&0&0
\end{array}$$

Hence, the equations we are looking for in order to describe our substitution are:

$$\begin{aligned}
0 &= 1 + x_1 + x_2 + y_3 + x_1y_1\\
0 &= x_1 + x_3 + y_1 + y_3 + x_1y_3\\
0 &= x_1 + y_1 + y_2 + y_3 + x_2y_1\\
0 &= 1 + x_1 + x_2 + x_3 + y_1 + x_1y_2 + x_2y_2\\
0 &= x_1 + x_1y_2 + x_2y_3\\
0 &= 1 + x_2 + x_3 + y_2 + y_3 + x_1y_2 + x_3y_1\\
0 &= 1 + x_1 + x_2 + y_2 + y_3 + x_3y_2\\
0 &= x_1 + y_2 + x_3y_3
\end{aligned}$$

Note that in general, large S-boxes are quite unfavorable for an algebraic representation. The number of linearly independent equations produced is equal to $n_{equ} = n_{mon} - 2^n$, where n_{mon} is the number of monomials used and n is the input size of the substitution. This relation implies that the number of monomials (and thus the complexity of the equations) needs to grow exponentially to match the size of the substitution input. This explains why PRESENT, with its small 4-bit to 4-bit S-boxes is an interesting first target for our attack.

C Definition of a Subkey Recovery Success Rate

Let the adversary $\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}$ be an algorithm with limited time complexity τ, memory complexity m and queries q to the target implementation $(\mathsf{E}_K, \mathsf{L})$. Its goal is to guess a subley $s = \gamma(x, k)$ with non negligible probability. For this purpose, we assume that the adversary $\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}$ outputs a guess vector $\mathbf{g} = [g_1, g_2, \ldots, g_{|\mathcal{S}|}]$ with the different key candidates sorted according to the attack result: the most likely candidate being g_1. A success rate of order 1 (*resp.* 2, ...) relates to the probability that the correct subkey is sorted first (*resp.* among the two first ones, ...) by the adversary. More formally, we define the experiment:

Experiment $\mathbf{Exp}^{\text{sc-kr-}o}_{\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}}$
$[k \xleftarrow{R} \mathcal{K};\ s = \gamma(x, k);\ \mathbf{g} \leftarrow \mathsf{A}_{\mathsf{E}_k,\mathsf{L}};];$
if $s \in [g_1, \ldots, g_o]$, **then** return 1, **else** return 0;

The o^{th}-order success rate of the side-channel key recovery adversary $\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}$ against a subkey variable S is straightforwardly defined as:

$$\mathbf{Succ}^{\text{sc-kr-}o,S}_{\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}}(\tau, m, q) = \Pr\ [\mathbf{Exp}^{\text{sc-kr-}o}_{\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}} = 1] \tag{2}$$

When not specified otherwise, a first-order success rate is assumed. [28] also defines an alternative security metric (the guessing entropy) that measures the effectiveness of a side-channel adversary in a more flexible fashion: it corresponds to the average position of the correct key candidate in the guess vector $\mathbf{g}$.

D Average Solving Times in Advanced Scenarios

Table 1. Solving times in advanced scenarios, when a 100% success rate is reached. Experiments are performed on an Intel-based server with a Xeon E5420 processor cadenced at 2.5GHz running a linux 32-bit 2.6 Kernel.

Scenario	Nbr. of W_H for 100% success rate	Average solving time (s)
Known P,C - consecutive weights	8 rounds	79,69
Known P,C - random weights	18 rounds	117,1
Unknown P,C - consecutive weights	8 rounds	45,59
Unknown P,C - random weights	26 rounds	214,12
Masked system - consecutive weights	16 rounds	393,86
Masked system - random weights	22 rounds	154,32

CAPTCHA Phishing: A Practical Attack on Human Interaction Proofing[⋆]

Le Kang and Ji Xiang

State Key Laboratory of Information Security, Chinese Academy of Science,
Yuquan road, 19A. Beijing, China
{kangle,jixiang}@is.ac.cn

Abstract. CAPTCHAs are widespread security measures on the World Wide Web that prevent automated programs from massive access. To overcome this obstacle attackers generally utilize artificial intelligence technology, which is not only complicated but also not adaptive enough. This paper addresses on the issue of how to defeat complex CAPTCHAs with a social engineering method named **CAPTCHA Phishing** instead of AI techniques. We investigated each step of this attack in detail and proposed the most effective way to attack. Then we did experiment with real Internet web sites and obtained a positive results. The countermeasures to prevent this attack are also discussed.

Keywords: CAPTCHA, phishing.

1 Introduction

CAPTCHA [2] is a verification technique used on the World Wide Web to determine whether a user is a human or a computer. It exists as a challenge response test for users. This test is designed to be a task which a computer can not perform but a human can easily do. A typical CAPTCHA may be an image containing distorted characters on the web registration form. The user can complete the registration task only if he entered the correct characters, and since a computer can not read distorted characters, CAPTCHA works.

CAPTCHAs have become one of the most common web site security modules. With this technique web servers can effectively screen out those users who employ automated programs for abusing the service. A CAPTCHA located on the comment board of blogs can prevent automated programs from publishing a mass of advertisements or other junk messages. Thus, it is beneficial to break CAPTCHAs for the attackers. AI technologies such as OCR have been developed to recognize distorted characters. It forces people to develop more complex CAPTCHA to confuse intelligent computers, and then upgrading both the attack and defence.

⋆ This work is supported by 863 Foundation No.2006AA01Z454, and NSF No.70890084/G021102.

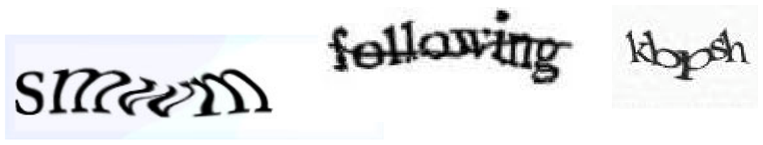
Another simple but potent method is based on social engineering. It is ingenious but underestimated by previous researches. In this paper we focus on how to organize a scalable and low-cost social engineering method to break CAPTCHA. We propose a quasi-phishing-attack scheme named **CAPTCHA phishing**, which can be implemented as an automated architecture to help attackers smoothly solve CAPTCHAs. Similar to traditional web phishing attacks, CAPTCHA phishing takes advantage of both technical and social vulnerabilities. The CAPTCHAs to be solved are passed on to online people by the attacker, and then there are some tactics to induce these users to contribute their answers. In this way, a CAPTCHA problem is transferred to a quasi-phishing problem. We have developed a toolkit and harvested a successful result to prove the feasibility of our idea. We also discuss how to prevent this attack.

In the remainder of this paper we explain our work in detail, which is interesting. Section 2 introduces related works about CAPTCHA and anti-CAPTCHA techniques, in Section 3 we expound on our work, and section 4 we give the results of the experiment. Section 5 discusses some possible ways to prevent our attack, and in Section 6, we make a conclusion.

2 Related Works

2.1 CAPTCHAs

CAPTCHA was firstly proposed by Luis von Ahn of CMU in 2002 [2] to help the free email system to screen out the 'bots' that would sign up for thousands of accounts and send out masses of junk email. Figure 1 shows some typical CAPTCHAs. Figure 1(a) is an early one generated by EZ-GIMPY: An image with randomly distorted characters and gradient color background. 1(b) is a more complicated CAPTCHA that adds a perturbing curve to prevent computer from segmenting the string into single characters; 1(c) makes the characters crowded and difficult to separate, some even human might not recognize.



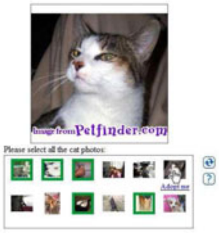
(a) (b) (c)

Fig. 1. Typical text-based CAPTCHAs

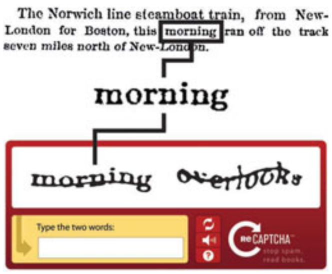
In recent years some complex CAPTCHAs were designed to confront developing OCR techniques. Asirra and reCAPTCHA are representative ones. Asirra(Animal Species Image Recognition for Restricting Access) [1] is the representative approach of those which are not text-based. A typical Asirra in

Figure 2a provides a group of pictures of animals, and demands users to point out all in a species.

reCAPTCHA [15] is a great approach that hits the shortage of OCR techniques. As Figure 2b shows, a reCAPTCHA appearing with two imaged words demands user to input both. Scanned text is subjected to analysis by two different OCR programs; in cases where the programs disagree, the questionable word is converted into a CAPTCHA. The word is displayed along with a control word already known. The system assumes that if the human types the control word correctly, the questionable word is also correct. This scheme guarantees that the challenge is not vulnerable to OCR software.



(a) Asirra


The Norwich line steamboat train, from New-London for Boston, this morning ran off the track seven miles north of New-London.
morning
morning overlooks
Type the two words:
reCAPTCHA
stop spam. read books.

(b) reCAPTCHA

Fig. 2. Complex CAPTCHAs

There are some other interesting CAPTCHAs such as audio-based or question-based, but they are not as widely used as the above two. It is troubling to treat those CAPTCHAs. Finally, the administrators of web sites should consider user experience.

2.2 AI-Based anti-CAPTCHAs

For each CAPTCHA scheme there are corresponding countermeasures already. [17,11,12,6,16] These approaches remain essentially the same despite all apparent changes. Optical character recognition(OCR) is a significant method. Early in 2003 Mori and Malik[11] had already broken the EZ-Gimpy[5] (92% success) and the Gimpy (33% success) CAPTCHAs with sophisticated object recognition algorithms. Moy et al [12] developed distortion estimation techniques to break EZ-Gimpy with a success rate of 99% and 4- letter Gimpy-r with a success rate of 78%.

In these anti-CAPTCHA techniques, machine learning is a popular tool. Take text-based CAPTCHAs for example, it is supposed that there exists some statistical features in all distorted instances of the same English character, and the features can be extracted from enough samples and used to recognize new instance. With this method, even those non-text-based CAPTCHAs are proven to be unsafe. Golle[8] has presented a successful machine learning attack against Assira.

Although the AI-based anti-CAPTCHA techniques show their excellences, they always fall behind their adversary. A simple but smart CAPTCHA scheme may require a complicated algorithm to break. Moreover, there is not a universal solution to break various CAPTCHAs.

2.3 Social-Engineering-Based anti-CAPTCHAs

To break CAPCTHA with social engineering is not a new concept. Early in 2003, [4,13] had revealed this ingenious crack, to offer a free porn site which requires that visitors key in the solution to a captcha before they can gain access. Each solution entered by an innocent user is relayed by the attacker to complete the malicious task.The process is summarized in Figure 3:

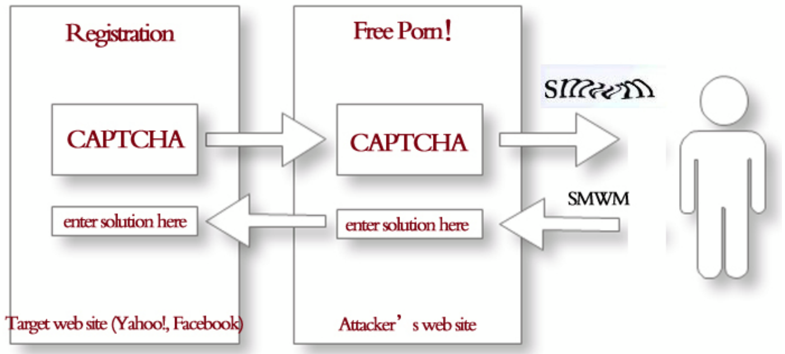


Fig. 3. To hire human to solve CAPTCHAs

In spite of the fact, as far as we know, this attack has not been widely used, and definitely has some weaknesses. For example, it is illegal to provide porn content, it is costly to maintain a whole web site, and it should be doubted that how the attacker attracted a large amount of users. Nonetheless, if the social-engineering-based CAPTCHA crack can be easily and safely deployed on a large scale, it is believed to work better than AI solutions. CAPTCHAs are so common components on web pages that it is easy to make someone contribute their solutions unconsciously. Moreover, the precision of the human is expected to be higher than any AI recognition. Some researches proposed their defence [10,9], but they are actually easy to break in practice.

Another human-based crack is, as [14] describes, the attackers pay humans for their solutions. In this paper we do not concern ourselves with this issue. To stop attackers from hiring humans is impossible, so it's none-sense to make a discussion about this. We focus on the attack that is robot driven and easily deployed by a single attacker.

3 CAPTCHAs Phishing

3.1 Principle

Suppose we are attackers who plan to widely spam on public blogs and decide to use the social-engineering-based method to crack CAPTCHAs, we can build a web site which is similar to the previous work introduced in [4], but it can not be expected that people will actively provide help, so we should induce them to come over to our site.

Phishing is a form of social engineering in which an attacker attempts to fraudulently acquire sensitive information from a victim by impersonating a trustworthy third party [7]. It employs generalized "lures". For instance, a phisher misrepresenting himself as a large banking corporation or popular on-line auction site will have a reasonable yield, despite knowing little to nothing about the recipient. Inspired by this attack, the idea of CAPTCHA phishing is to publish phishing messages on those web sites with many users to attract people.

Although CAPTCHA phishing employs the same principle as normal phishing attacks, it is only a quasi-phishing method because its target is not the lured people. Neither the lured people nor the web site which convey phishing messages suffers from any loss. In this paper when we refer to "victims", it denotes those target web sites that generate CAPTCHAs. The people who give their solutions are called "fools" because they do not lose anything but can still be cheated. We will discuss the details.

3.2 Phishing Strategy

A phishing strategy is the logic process of CAPTCHA fetching, transmission, and phishing. We proposed three strategies.

Asynchronous fetch-and-phish (AFP). strategy: Whenever a CAPTCHA is to be solved, it is fetched and immediately deployed to phishing web site, and will not be changed until a fool enters an answer.

AFP strategy is the simplest of all. The shortcoming is, the session of CAPTCHA may expire. Default HTTP session expiring time is 20 minutes. If it can not be ensured that the visits to the phishing page are frequent enough, AFP may become an inefficient policy.

Synchronous fetch-and-phish (SFP). strategy: The robot refreshes encountered CAPTCHA at a regular interval and replaces the old CAPTCHA with a new one. When someone enters a solution, the solution is adopted as the current CAPTCHA's answer.

Most web sites employing CAPTCHA provide a refreshment user interface. If the imaged text is so indistinct that even a human can not definitely recognize it, the user can choose to try another challenge. SFP guarantees that the CAPTCHAs will not expire if the interval of refreshment is small enough, such as 30 seconds, however, continuing refreshment is easy to detect.

Phish-and-fetch (PF). strategy: In a phish-and-fetch strategy the robot waits until someone visits the phishing page, then immediately gets a CAPTCHA, and sends this temporarily retrieved image to the fool.

PF strategy overcomes the defect of AFP and SFP, but has a network delay problem. It can not be exactly estimated how long it will take from when a user visits the phishing page to when the fake CAPTCHA is completely sent to him. If this time span is too long, the page may be closed before the CAPTCHA is displayed.

3.3 Phishing Carrier

We implemented a CAPTCHA phishing based on a web page. It should neither be expensive nor bring about risk to us. We deployed CAPTCHA phishing interface on a page(This page is the called **CAPTCHA Carrier**)and then chose some web sites as phishing areas to publish phishing messages. Obviously, the best phishing areas are those web sites that have a large amount of traffic, such as popular web forums. These sites often gather masses of users who always browse page by page.

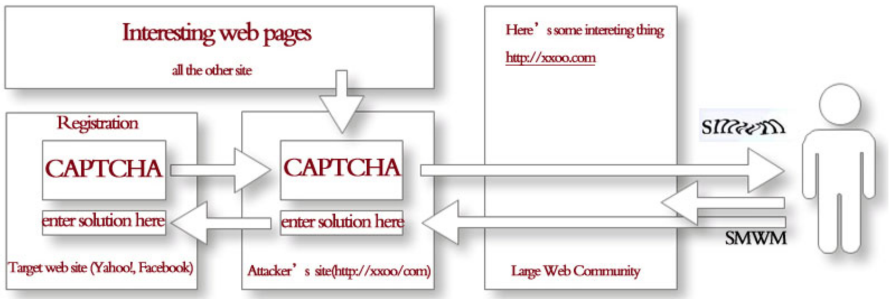
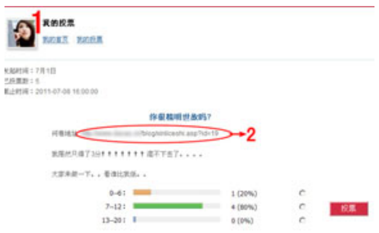


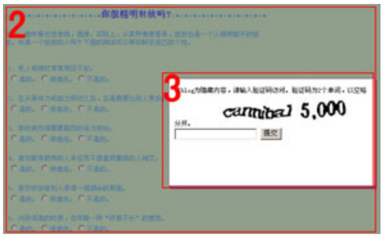
Fig. 4. complete architecture of CAPTCHA phishing

Figure 5 shows our page-based phishing instance. Figure 6(a) is a piece of phishing message that sets up a vote and a hyperlink pointing to a psychological test page. Viewers can do the test firstly and submit their scores. The psychological test page shown in Figure 6(b) is our phishing page. It demands viewers to enter a solution before they do the test. The page in 6(b) only contains the user interface for CAPTCHA displaying and solution input, and an <iFrame>tag, which includes a psychological test page of another site. In other words, the content in phishing page is easy to change, so the cost for attacking is reduced. The improved architecture is in Figure 4.

Although this CAPTCHA phishing scheme in Figure 4 seems plausible, its effect should be doubted in practice. A simple but non-ignorable reason is most



(a) Phishing message in SNS



(b) Phishing page

Fig. 5. Instance of page-based CAPTCHA phishing: (a) is a piece of phishing message which contains a hyperlink to phishing page; (b) labeled 2 is the phishing page which displays a psychological test, 3 is the CAPTCHA user interface which shows: the content will be visible after the above word is decoded

online people are not willing to click a hyperlink which points to an outer address. In our experiment we can demonstrate online people are lazy and inpatient. They may feel it's trouble or risky to click a unknown hyperlink. It is the bottleneck of the attack.

So a better way is to integrate the phishing carrier into the phishing area. If the attacker can paste the fake CAPTCHAs and input interface directly on the page of web communities, it can deceive more users. However, most web sites do not allow users to code script with data interaction. Without considering regular cross-site scripting attack based on javascripts, we found a legal solution: Adobe flash files can implements internet communication interface. For example, a flash object displayed on the page of domain A may actually be stored in domain B, and can call remote HTTP services of domain C. When users visit web page of domain A, the content they browse in the flash is loaded from domain C. When a user interacts with this flash, the client data can also be sent to any domain. In other words, it allows the attacker to combine the phishing area and the phishing carrier together, it is completely allowed, flash files are usually used to republish media and advertising animus. Although Adobe has realized the potential threat and has restricted the cross-domain calling in the new version, it is easy to bypass by using old version.

Figure 6 shows the framework of flash-based phishing. The flash is only a shell which presents a CAPTCHA user interface, and remotely loads CAPTCHA images from the attacker's server. It also relays some flash file such as game or video. The visitors who want to play the game or enjoy the video will be required to solve a fake CAPTCHA. Figure 7 shows an instance. The flash is displayed on a page of a web Forum, and shows a flash game(We make the game's content blurry). A popup panel contains a CAPTCHA image and a user input interface is displayed.

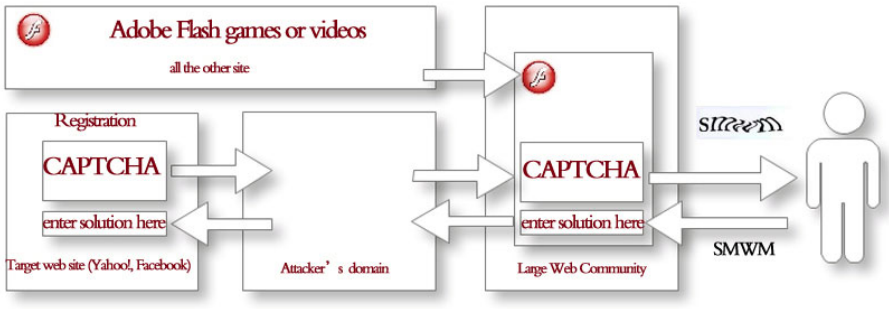


Fig. 6. Architecture of flash-based CAPTCHA phishing

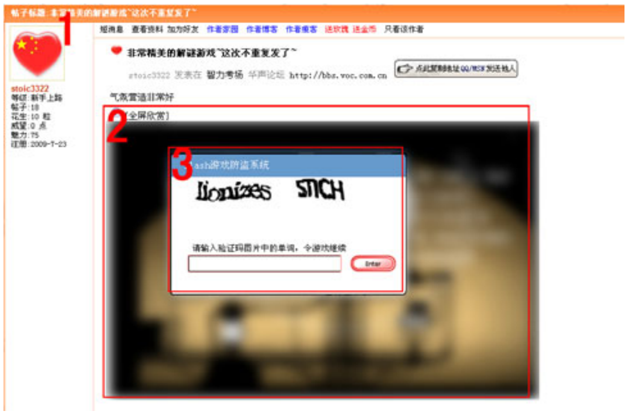


Fig. 7. Instance of flash-based CAPTCHA phishing: 1 is a post in web forum; 2 is the phishing flash which exhibit a game; 3 is the CAPTCHA input panel which shows: please enter the two words to make game continuing

3.4 Verification to Solutions

Each fool who created the solution is waiting for a confirmation on the phishing page. We can choose to give him a real or fake verification.

A real verification denotes that whether the solution is correct or not, it will be fed back to the fool. One benefit is to prevent fools from discovering that the CAPTCHA is a fake. On the other hand, if a fool gives a wrong solution, we can demand him to contribute another. However, Real verification increases the complexity of data transmission and time delay. Fake verification does not need redundant communication. The simplest method is to make the user get passed no matter what he entered, but it is improper. If a user accidentally found that the CAPTCHA is invalid, he may enter nothing or freely fill out some characters next time when he meets another fraud. A better fake verification scheme is as follows:

1. For each phished solution, make it passed with p probability. and with 1-p probability, the user is notified that his answer is incorrect, then another CAPTCHA is displayed to call for a new answer.
2. Apply a basic format check using regular expressions to solutions. For example, in a reCAPTCHA, the string is required to be in a format which is made up of two words and one blank split as regular expression ".* .*". So randomly filled answers are likely to be screened out.

4 Experimentation

The experiment is organized as follows. Two categories of web communities are chosen as phishing area: web forum and social network service(SNS). The former is a typical form of traditional web application. On web forums users create and read a great amount of posts every day. The latter is a representative one in web 2.0 time. Each SNS user shares his individual information with his friends. In each category we chose large sites that gathers millions of users (the site names are not revealed), and deploy both page-based phishing and flash-based phishing. The results will be compared.

To evaluate the results we measuresd the **efficiency**, which shows the relationship between the number of phished solutions and our cost, the **precision**, which measures the correct rate of phished solutions, and the **time lag**, which reveals the time required by different kinds of phishing strategies.

4.1 Efficiency

An efficient phishing attack should be able to obtain as many solutions as possible with few phishing messages. We tried page-based and flash-based phishing respectively in different web communities, and measured the results. In page-based phishing each phishing message contains a hyperlink with a unique identification number, so do the flashes. The identification number will be sent back to our server together with solutions, so we can get to know where each solution comes from. Both flash-based phsihing and page-based were tested in both web forum and SNS. In each situation, 10 phishing messages were published and traced for 10 days.

Table 1. Efficiency of CAPTCHA phishing

	flash-based		page-based	
	Web Forum	SNS	Web Forum	SNS
#visits	1890	942	322	409
# solutions	1905	905	139	126

As table 1 shows, flash-based phishing presented more attraction. In web forum it attracted 1,890 users, and obtained 1,905 solutions in total (some fools have contributed several), in SNS it also harvested 942 visits and 905 solutions. In comparison, page-based phishing performed rather weaker. In web forum, it only successfully induces 322 users, 139 of whom gave their solutions, in SNS the corresponding number is 409 and 126. Moreover, we found that 30% of the phishing messages in page-based phishing were deleted by administrators of the web forum in 10 days, while all the phishing flashes were safe. It seems web phishing is so notorious that web sites do not welcome suspicious outer links.

Duration: A good CAPTCHA phishing should have continuous effect. In our experiment each phishing message has been traced from when it was posted. Because the posts on the page of web communities are usually ordered by their publication time and popularity, they may be 'sunk' and lose focus in several days. Only those posts that draw a lot of attraction can get a high rank. We certainly hope our posts which convey phishing messages will attract more users.

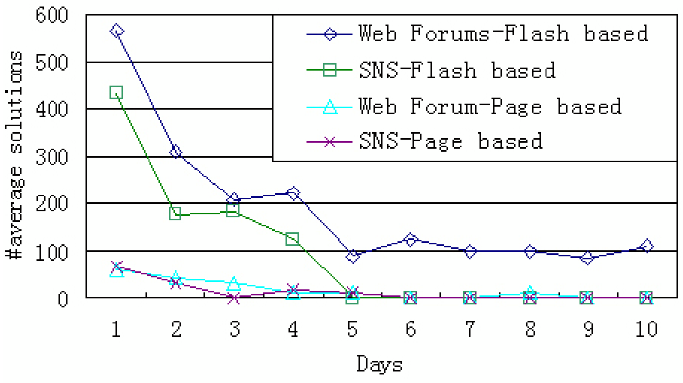


Fig. 8. Duration of different CAPTCHA phishing

Figure 8 shows the results. The duration of flash-based phishing in web forums is much longer than the others. In the first 5 days the phishing flashes performed in web forums as well as in SNS, then it lost focus rapidly in SNS, and worked for longer time in web forums. It can harvest ten of solution even if the phishing messages have been posted for 1 month in web forums. The results of the other two are worse. Page-based phishing performs weakly, few people show their interest in it after the first 5 days.

This result illustrates two facts: First it proves that people indeed present their lack of patience. A hyperlink pointing to an outer address is not very attractive even if there is a brief introduction to the content. It is reasonable for us to blame the effect of the page-based CAPTCHA phishing. A flash which facilitates users performs much better. Second, the traditional web forums is a better phishing area than SNS. The difference is mainly in duration, phishing messages in web forums get longer term results, because the information updating in SNS is more frequent, a majority of contents will be 'sunk' more quickly after they were published.

Scalability We increased the attack size and scope. 10 popular web forums that have similar visits number were chosen to paste the phishing flashes. The experiment was deployed for 3 times, in each of which the number of flashes increased from 10 to 50. Table 2 shows the results of different attack size in the first day.

Table 2. Scalability of CAPTCHA phishing

attack size	10	30	50
#solutions	578	1121	2146

The number of solutions that each flash harvested are not equal. On average there are about 40 to 50 solutions per flash in the first day. The effect of the phishing flashes mainly depends on the visits of their carrier sites, it can be be asserted how much solutions a single phishing flash can obtain, so the attacker can expand the attack scope and size to achieve an overall effect.

4.2 Precision

In theory the CAPTCHA solutions contributed by humans are more precise than any computers. We chose two different kinds of CAPTCHA to attack, EZ-GIMPY used by Yahoo! and reCAPTCHA used by Facebook. For each we tested 1,000 CAPTCHA images, whose answers were already known. In every 1,000, 500 utilized real verification and 500 utilized fakes. The success rate p of fake verification is set to 0.6.

Table 3. Precision of CAPTCHA phishing

	EZ-GIMPY		reCAPTCHA	
verification	real	fake	real	fake
#solutions	500	500	500	500
# wrongly recognized	32	30	139	176
#deleberately wrongly filled	19	17	21	32

The result is given in Table 3. The precision of human recognizing EZ-GIMPY is as high as 92%. Apart from this, 6% of the fools have wrongly recognized, and approximately 2% intentionally gave incorrect solutions whether the verification was real or not. Those deliberately wrongly filled solutions are easy to identify, because people prefer to type several adjacent keys on keyboard. The precision of reCAPTCHA recognition is lower, reaching only about 72%, because the imaged text in reCAPTCHA is often distorted so much that even a human can not definitely recognize it correctly. Moreover, it is interesting that more people chose to freely fill characters when they encountered reCAPTCHAs. It seems that complex challenges are better at making users to lose their patience.

This result demonstrates that the precision of CAPTCHA phishing is considerable. By comparison we concluded that whether the feed back to the fools is real or not is not very important. Fake verification will result in a few more wrong solutions, and real verification also can not prevent some curious people from trying to freely fill the information.

4.3 Time Lag

We made a statistical work to measure how long the whole process of a CAPTCHA phishing task will take. The process flow is similar to a communication protocol which contains a 4-way handshake. In Step 1 the fool visits the phishing carrier. In step 2 fake CAPTCHA is transmitted and displayed. In step 3 the transmission of CAPTCHA solution is done. Finally, the fool is given a verification result for his answer.

The time spans between each two adjacent steps are recorded for 100 times. Different phishing strategies have different time lag during step 1 and 2. In SFP and AFP, the CAPTCHA image is already prepared when a fool visits, but in PF, the image should be fetched instantly. The verification scheme determines the time lag during step 3 and 4, it requires some time to verify the solutions.

Table 4. Time lag of CAPTCHA phishing

	Time lag (in seconds)		
Step	1-2	2-3	3-4
APF & SPF	0.6	11.5	-
FP	6.2	10.8	-
real verification	-	-	3.5
fake verification	-	-	0.4

Table 4 shows the mean values. Taking all factors into consideration, In can be concluded that PF strategy is the best, not only because it can overcome the defect of the other two, but also because its time lag is trivial. On average the fake CAPTCHA will be displayed in the browser in 6.2 seconds after the page is opened.

The time span from step 2 to 3 reflects the reaction of the fools. It takes less than 12 seconds for most people to enter their solutions, meaning that people do not suspect our trick. They consider CAPTCHA so common a thing that they will solve it wherever it is necessary. Once they encounter one, they solve it.

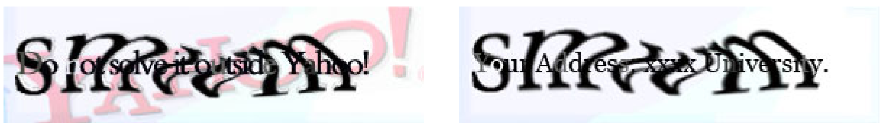
It took 3.5 seconds on real verification on average, which is acceptable. Considering the result in section 4.2, real verification will not perform worse than the fake one.

5 Possible Countermeasure for CAPTCHA Phishing

CAPTCHA crack based on social engineering is difficult to prevent. In our attack, the content of CAPTCHA phishing did not contain any illegal or malicious information. The usual phishing tactics such as imitating HTTP address and imitating web page visual views were not necessary at all. Just because of this, everything in CAPTCHA phishing looks so normal that the swindle in CAPTCHA phishing is invulnerable to current anti-phishing methods[3].

Considering the whole process flow of CAPTCHA phishing, all the steps are transparent to external people except for two phases: phishing interface and original CAPTCHA extraction. People have to try to expose the fraud of phishing, or prevent the attacker from retrieving CAPTCHAs from the victim site, but to completely achieve the two goals is scarcely possible. In the former one, take our flash-based method as an example, people can not find any malicious or suspect code even if they decompile and analyze our phishing flash file. In the latter one, it is impossible to prevent attackers from extracting anything that is already shown in browser.

A considerable method is to stop the attacker from relaying CAPTCHAs. Golle [9] has proposed a scheme named **CAPTCHA token** that requires soft keyboard to input. But flash can implement powerful scripting, the attackers can take a screenshot to extract complete information, and easily reproduce the user interface such as soft keyboard in flash. So special user interface can not stop relay attack. A good countermeasures we propose is **identified CAPTCHA**, which is embedded into a watermark which contains identification of this CAPTCHA images' original domain or user client and some cautionary information. The watermark should be properly designed to make it difficult to be cleaned by image processing. It is supposed that the attackers who employ social engineering do not have enough ability using complicated AI techniques.


SMRAWMY
Do not solve it outside Yahoo!
YAHOO!
SMRAWMY
Your Address: xxxx University.

(a) (b)

Fig. 9. Two demo identified CAPTCHAs to prevent social engineering. (a) warns: Do not solve it outside Yahoo!. (b) indicate the address of client: xxxx university.

We roughly improved an EZ-GIMPY CAPTCHA as Figure 9 shows, 9(a) contains a Yahoo! logo as background and presents a line of caution to warn people. 9(b) shows a line that labels the address of visitor(the address can be looked up according to visitor's IP). The caution text should be located, rotated or distorted randomly. While confronting this CAPTCHA, attackers have to deploy phishing attack inside Yahoo!, but if they do this, the risk of being exposed will increase. Moreover, the language of the caution must be the same as the CAPTCHA text, otherwise, the attacker may employ a cross-language attack, where they can relay the CAPTCHA with English caution to Chinese people, or do the opposite.

We have tested identified CAPTCHAs. Images were embedded into watermark and delivered to people. As we expected, this time people did not play the roles of fool. In 261 visitors only 35 keyed in their solutions. The identified CAPTCHAs successfully reduced the success rate to 13.4%. The scheme in Figure 9(b) performed a little better than that in Figure 9(a).

Another countermeasure is to increase attackers' cost for retrieving CAPTCHAs. Halprin [10] has proposed a scheme named **dependent CAPTCHA**, whose solution requires data that exists on the page in which that CAPTCHA is given, but outside the CAPTCHA image itself. It is useless for the attacker to extract a single image, and it indeed increases the difficulty of retrieving. However, the attacker can also extract complete information by partly screen shot to the browser.

We proposed a scheme named **hidden CAPTCHAs**. The Attacker usually find and extract CAPTCHAs according to their HTML element attributes such as ID, name, image Url, position and element index, they will be bothered by following settings.

1. To embed the CAPTCHA in a random image on the page.
2. Not to assign the images any attributes which can identify it such as ID.
3. To make all the images on the page have similar Url format,
4. To embed tiny random noise into all the images on the page.
5. To fix the position of CAPTCHA input interface.

HTML parser will be confused by hidden CAPTCHAs. It can not easily extract the correct image because the right one has no obvious feature. All the images are similar in HTML attributes. If there are 20 images on the page, the crack rate will be reduce to 1/20 when the attacker employ OCR to attack, and it is inconvenient for the attackers to relay all these images to phish a solution. However, the same applies to dependent CAPTCHA, hidden CAPTCHAs also sacrifices user experience. The distorted imaged text is not located near the solution input interface, the users will have to find it, and all the images are always regenerated when the page is requested for, it is a waste of cache.

6 Conclusion

This paper shows how easy it is to break CAPTCHAs with a quasi-phishing scheme. The attack achieves its best result when attacker uses flash-based phishing in web forums, both real and fake verification do work. PF strategy is the most practicable of the proposed three. This attack is easy to deploy but quite difficult to penetrate, and can get a high precision. It is quite possible for some attackers to use it in large scale web spamming or other robot-driven attacks. We think that so far there's hardly a technical solution that can stop this social-engineering-based attack, but people can bring problems to the attackers with especially-designed CAPTCHAs if they find methods to achieve equilibrium between security and user experience.

References

1. Asirra: a captcha that exploits interest-aligned manual image categorization. In: 14th ACM Conference on Computer and Communications Security, pp. 366–374. ACM Press, New York (2007)
2. Ahn, L.V., Blum, M., Langford, J.: Telling humans and computers apart automatically. Commun. 47(2), 56–60 (2004)
3. Badra, M., El-Sawda, S., Hajjeh, I.: Phishing attacks and solutions. In: 3rd International Conference on Mobile Multimedia Communications, ICST, Brussels, Belgium, pp. 1–6 (2007)
4. Caine, A., Hengartner, U.: The ai hardness of captchas does not imply robust network security, pp. 367–382 (2007)
5. captcha site.: http://www.captcha.net/
6. Chellapilla, K., Simard, P.Y.: Using machine learning to break visual human interaction proofs (hips). In: NIPS (2004)
7. Dhamija, R., Tygar, J.D., Hearst, M.: Why phishing works. In: SIGCHI Conference on Human Factors in Computing Systems, pp. 581–590. ACM Press, New York (2006)
8. Golle, P.: Machine learning attacks against the asirra captcha. In: 15th ACM Conference on Computer and Communications Security, pp. 535–542. ACM Press, New York (2008)
9. Golle, P., Ducheneaut, N.: Keeping bots out of online games. In: 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, pp. 262–265. ACM Press, New York (2005)
10. Halprin, R.: Dependent captchas: Preventing the relay attack (2009)
11. Mori, G., Malik, J.: Recognizing objects in adversarial clutter: breaking a visual captcha. In: Proceedings of 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. I-134–I-141. IEEE Press, Los Alamitos (2003)
12. Moy, G., Jones, N., Harkless, C., Potter, R.: Distortion estimation techniques in solving visual captchas. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004, vol. 2, pp. II-23–II-28 (2004)
13. Boing Boing: Solving and creating captchas with free porn (2004), http://boingboing.net/2004/01/27/solvingandcreating.html
14. Inside India's CAPTCHA solving economy describes (2008), http://blogs.zdnet.com/security/p=1835
15. Ahn, L.V., Maurer, B., Mcmillen, C., Abraham, D., Blum, M.: Recaptcha: Human-based character recognition via web security measures. Science, 1160379 (2008)
16. Yan, J., Ahmad, A.S.: Breaking visual captchas with naive pattern recognition algorithms. In: 23th Annual Computer Security Applications Conference, pp. 279–291 (2007)
17. Yan, J., Ahmad, A.S.: A low-cost attack on a microsoft captcha. In: 15th ACM Conference on Computer and Communications Security, pp. 543–554. ACM, New York (2008)

An Attack and Repair of Secure Web Transaction Protocol for Anonymous Mobile Agents

Saba Jalal and Brian King

Purdue School of Engineering and Technology
Indiana University Purdue University Indianapolis
briking@iupui.edu

Abstract. The security problems that impact mobile agents include authentication and privacy. A challenging task is to implement secure mobile agents that provide anonymity for the mobile agent. Wang, Zhang, and Wang constructed a scheme that provides secure web transactions with anonymous mobile agents over the internet. Our paper will discuss the Wang, Zhang and Wang protocol(WZW) thoroughly, our work demonstrates the weaknesses of the WZW protocol, showing that it is not as secure as it is claimed to be. In this paper, we will construct attacks on the WZW protocol, provide several repairs and construct a secure scheme for conducting E-commerce using anonymous mobile agents.

Keywords: Anonymity, Mobile Agent Security, Mobile Agent Authentication, E-commerce.

1 Introduction

Mobile agents are autonomous or semi-autonomous software that migrate from one host to another in the network. Mobile agent reduce bandwidth and network traffic. Moreover, mobile agents are good for efficiency and space savings. Mobile agents are used in many applications, like E-commerce, Intrusion Detection, and Social Networking. In the context of this paper, a motivating application for mobile agents will primarily be E-commerce. However our work has implication of the use of mobile agents well beyond this setting.

Secure anonymous routing protocols have been developed [4,10,13] to provide routing in ad hoc networks. Wireless hosts in an ad hoc network transmit and receive packets, malicious nodes will affect the security of the network if secure transmission is not handled properly. In such scenarios, utilizing mobile agents to provide secure communications between nodes will improve efficiency.

A mobile agent can be utilized to solve many problems and adds value to any system as long as the mobile agent system satisfies the security and integrity concerns. What are some of the security concerns that might arise? A mobile agent may be a threat to a host or website on the network, also mobile agents can tamper with other mobile agents [3]. We will discuss many of these problems in the next section.

In a court of law, the motto is : “Innocent until proven guilty”. When using mobile agents we can’t afford such a philosophy. In order to implement a secure system, we need to consider any potential malicious behavior that can occur within the network/system. There are a number of attacks that the system can encounter in which the integrity of the system is compromised. A mobile agent consists of state, code and data. We need to maintain the integrity of state, code and data during the lifetime of the agent. If the agent is tampered by a malicious host, then the user that initiated that agent is damaged, but so is the next host.

In [12], a model constructed by Wang, Zhang and Wang was introduced. This model discusses the anonymity of a mobile agent system; this model also allows the mobile agent to sign a message without revealing the private key or the identity of the user. In this paper we will demonstrate a number of weaknesses of the Wang, Zhang and Wang (WZW) protocol. We construct several attacks. We then construct a secure anonymous authentication scheme for mobile agents.

2 Background and Prior Work

Security in Mobile Agents
Security is defined by three important factors: *Confidentiality* or *Privacy*, *Integrity* and *Availability*. *Privacy* of the data and information for each party is critical, for example one host might not want other hosts to view its products or prices. This could be because these two hosts are competitors. Further, anonymity may be preferred, that is, users may favor to remain anonymous, not wanting other hosts to deduce their purchasing patterns. In a mobile agent system which supports anonymity, the host needs to be able to authenticate the transaction without identifying the user. Wang, Zhang and Wang constructed such a protocol [12]. However, we will show that there are problems with their protocol. *Integrity* is an important security property that has to be maintained. Integrity assures that the code of the agent is not tampered with. If the agent code is modified in any way by an unauthorized party it should be detected. Only the owner of the code can change it and in some cases authorization is given to a trusted party. *Availability* on the other hand is related to the ability to use the resources of the host. Each agent should be assigned certain resources for its execution duration on the host. If there is a malicious agent, this agent might try to use as much resources as it can and deny the other agents from the resources, this is a denial of service attack. Consequently there are three primary security aims: *securing hosts from attacks by malicious agents*, *securing agents from attacks by malicious hosts*, and *protecting agents from attacks by other agents* [3,7].

Cryptography
One cryptographic tool that is used throughout our work will be elliptic curve cryptography ECC [9]. There is a natural “arithmetic" that can be defined on the elliptic curve E, such that E can be interpreted as an additive abelian group,

where $\mathcal{O}$ is the additive identity. For cryptographic applications, the curve E is selected so that it has a subgroup of large prime order. The order of the subgroup of E will be denoted by q. An essential calculation is to compute the scalar multiple of a point $kP = P + P + \cdots + P$ where P belongs to the subgroup of E of large prime order and k is the secret key. Thus the secret key k belongs to the field $\mathbb{F}_q$. In the context of cryptographic purposes the secret may be the scalar and the public value will be the scalar multiple. The *elliptic curve discrete log problem (ECDLOG)* is the problem that given an elliptic curve E, q (the order of the subgroup of E of prime order), the point P belonging to this subgroup, and the scalar multiple kP, then it is "computationally hard" to determine k.

We will also use pairing based cryptography. Let E be an elliptic curve defined over $\mathbb{F}_p$ and $\mathbb{G}_1$ is an additive subgroup of E of order q, $\mathbb{G}_2$ is a multiplicative group of order q.

Definition 1. *A map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is said to be bilinear pairing if it satisfies the following properties:*

Bilinearity $e(aP, bQ) = e(P, Q)^{ab}$ *for all $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p$,*

Non-degeneracy $e(G, G) \neq 1$ *where G is a generator of $\mathbb{G}_1$, and*

Computability *there exists and efficient algorithm that computes $e(P, Q)$ for all $P, Q \in \mathbb{G}_1$.*

3 Wang, Zhang and Wang's Secure Web Transaction with Anonymous Mobile Agent Protocol

In 2003, Wang, Zhang and Wang [12] constructed a protocol, which we will call the WZW protocol. Their goal was to provide secure e-commerce using anonymous mobile agents. Essentially their goal was to construct a mobile agent protocol that provides user anonymity at the same time providing security for the hosts against potential malicious agents and malware. A trivial solution to the problem of providing anonymity, is to have each user register with the trusted third party. The trusted third party will have the private information of each user and they are trusted so they will not reveal anything to the hosts. Every time a user wants to release a mobile agent they will approach the trusted third party. The trusted third party will create pseudonym credentials for the agent. The trusted third party will send the agent and user's request to the host, and the agent will migrate in the network as specified. This approach however will create a bottleneck on the trusted third party, as every mobile agent released within the mobile agent system will require active processing (user-centric) by the trusted third party. A primary goal is to create a scheme so that the trusted third party provides at most off-line support. Thus there is no active role (user-centric) played by the trusted third party when a mobile agent is released. The model in the WZW protocol consists of three types of parties:

User Agent Home (UAH) The UAH, also denoted by *user*, is the party that generates, constructs and releases the mobile agent.

Agent Management Center (AMC) The AMC, also called the *trustee*, is the trusted party that all users who wish to release a mobile agent must register to.

Host The host is the party in the network that the mobile agent migrates to and who executes the mobile agent instructions.

We now describe the WZW protocol as described in [12]. The WZW protocol utilizes an elliptic curve E defined over $\mathbb{F}_p$. The elliptic curve possesses a prime subgroup of order q, let the elliptic curve point G denote a generator of the prime subgroup of order q. If P is a point of the elliptic curve we will use $R_x(P)$ to denote the x-coordinate of P and $R_y(P)$ to denote the y-coordinate of P. Let X_{AMC} and P_{AMC} denote the private and public key pair of the AMC. Thus $P_{AMC} = X_{AMC}G$. For brevity, we will denote $H(R_x(P_{AMC})||R_y(P_{AMC}))$ in the following by H_0. This value is known by all parties. Each party who would like to participate in the system will be required to register with the AMC. This is a one-time registration. Before the user registers, they must generate a long-term private key/public-key pair. This is done once by the user.

3.1 User Key Generation

The User C selects a private key $X_c \in_R \mathbb{F}_q^*$. [1] The user then computes the public key $Y_c = X_cG$. Here X_C and Y_C are user C's private key/public key pair. Y_c and X_c are the long term public and private keys of the user C.

3.2 Registering a User

In order to utilize the mobile agent network, user C must register with the trusted party AMC. To register user C sends its identity ID_C and public key Y_C to the AMC. The AMC chooses $\omega_c \in_R \mathbb{F}_q^*$ and computes A_C where $A_c = \omega_c G - Y_c$. The AMC also computes $\gamma_c = H_0\omega_c + R_x(A_c)X_{AMC} \mod q$. The AMC then stores $(A_c, \gamma_c, Y_c, ID_c, \omega_c)$ in its database and returns A_c and γ_c to the user, this represents the long term certificate of the user in the sense of its use within the mobile agent system. The user C verifies that the certificate from the AMC by checking that γ_C satisfies the following congruence relation.

$$\gamma_c G = H_0(A_c + Y_c) + R_x(A_c)P_{AMC}. \tag{1}$$

Observe that $\gamma_c G$ represents the ability of the user to authenticate to other members of the system because it binds their (AMC generated) A_C, their long-term public key Y_C, and the public key P_{AMC} of the AMC. More important, because the user possesses γ_C they can generate multiples of the right hand side of equation (1). Observe that the AMC keeps ω_C private from the user C, that is by sending A_C rather than ω_C the user knows A_C but not ω_C, this of course protects the AMC's private key X_{AMC}, which is included as one of the terms in γ_C.

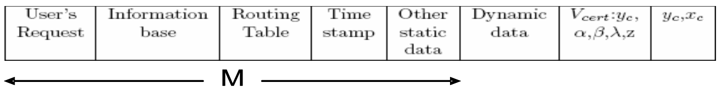
[1] We use $\in_R$ to denote that we select the element uniformly random from the targeted set.

3.3 User Releases Mobile Agent

When the user would like to utilize the mobile agent system and release a mobile agent, the user's mobile agent will need to provide a signature to demonstrate that they were released by a user who is authorized (i.e. has registered). There are a number of potential problems with this process because the user's anonymity needs to be preserved. Later, the mobile agent will need to sign the output that the host generates. There are a number of problems with this as well. One, any signing key will have to be provided among the data and instructions that the agent carries, thus a signature with the user's long term private key cannot be generated at the moment the agent arrives on the host because all instructions will be executed on the host and so the host would know the key. Secondly, the user's identity needs to be protected from the host, remember that anonymity is a goal, so one cannot use any long-term information that a host could use to identify users from their mobile agent data. Wang, Zhang and Wang [12], solved this problem by requiring the user to generate what they termed as "one-off keys", these are keys that would be generated "fresh" each time the user releases a mobile agent.

Creating the One-Off keys
The user creates what the authors called *one-off keys*. The user C selects $x_c \in_R \mathbb{F}_q^*$ and then computes $y_c = x_c G$.

Mobile Agent Data Structure
The entire data structure that is included with the mobile agent is illustrated in Figure 1. This data structure includes M, which contains the user's request (instructions that need to be executed), information base, routing table, timestamp, and other static data. The data structure also includes dynamic data, the Virtual certificate V_{cert} and the one-off keys $\{x_C, y_C\}$.



User's Request	Information base	Routing Table	Time stamp	Other static data	Dynamic data	V_{cert}:y_c, α,β,λ,z	y_c,x_c

Fig. 1. Mobile Agent Data Structure

Creating the Virtual Certificate
The user cannot send γ with the agent since this would providing tracing material for any host to identify the user who released the agent. Consequently the user must mask their information, at the same time the mobile agent needs to provide information to the host that proves that the Mobile Agent belongs to an authorized user. Let M denote the static data that the user will place on the mobile agent. Thus M contains the user's request (instructions that need to be executed), information base, routing table, timestamp, and other static data. The user will need to generate a virtual certificate called V_{cert} by completing

Algorithm 1. The V_{cert} is used by Wang, et. al. in the agent authentication part of the WZW protocol.

Algorithm 1. AGENT$_C$'s Virtual Certificate Generation V_{cert} for static agent data M

```
Input: users $A_C, X_C$, one-off keys $x_C, y_C$ and static agent data $M$
Output: $V_{cert} = (\alpha, \beta, z, \lambda, y_C)$
 1: User selects $\psi \in_R \mathbb{F}_q^*$
 2: User computes the elliptic curve point $T = \psi G$
 3: Let $t = R_x(T) \mod q$
 4: if $t = 0$ then
 5:     Goto 1 and select a new $\psi \in_R \mathbb{F}_q^*$
 6: Let $z = tR_x(A_c) \mod q$
 7: Let $\alpha = A_C + T$
 8: Let $\beta = t(A_C + T)$
 9: Let $\lambda = \gamma_c t + (\psi t - X_c t + x_c H(M))H_0) \mod q$
10: Output the $V_{cert} = \{z, \alpha, \beta, \lambda, y_C\}$
```

3.4 Agent Authentication to Host and Validity Verification

The mobile agent migrates through the network from host to host. In the WZW protocol, the mobile agent authenticates to a host, as follows. The host checks the validity of the agent by checking that λ satisfies the following congruence relation, where λ is included in the virtual certificate V_{cert}.

$$\lambda G \overset{?}{=} H_0(\beta + y_c H(M)) + zP_{AMC}. \tag{2}$$

In the WZW protocol, if the congruence is satisfied then the agent will be authenticated and the agent's instructions will be executed by the host. Due to the assumption that the ECDLOG problem is hard, and the fact that the agent can provide the discrete log term of zP_{AMC} (which is z) implies that the agent used their γ_C in their construction of λ. However as we will see in Section 4, that this (authentication) does not imply that the λ is formed in the manner that Wang et. al. defined.

3.5 Signature and Verification

After the host W has executed the agent's instructions an output/bid W_{bid} is created by W. The agent needs to sign this output/bid W_{bid}. In the WZW protocol, the agent's one-off keys will be used to generate the signature on the bid W_{bid}. That is, the host uses the one-off keys provided by the agent in the WZW protocol, to generate a signature on the W_{bid}. Thus the agent can sign without revealing the private key of the user. Let $m = H(ID_w, W_{bid})$. The agent generates signature Sig on m by the following algorithm.

The host verifies the signature of the agent by checking that r satisfies the congruence $rG = H(m)K + R_x(K)y_c$. The host W then signs it bid W_{bid} using its digital signature key and constructs $Sig_W(ID_W, W_{bid})$. Note anyone with m, Sig, V_{cert} can verify the agent's signature. By defining Sig in this manner, the authors imply that Sig represents a binding of the agent with V_{cert} to the output m.

Algorithm 2. AGENT$_C$'s signature on host's W_{bid}

```
Input: $m = H(ID_w, W_{bid})$, one-off keys $x_C$
Output: $Sig = \{K, r\}$
1: Select $k \in_R \mathbb{F}_q^*$
2: Compute the elliptic curve point $K = kG$
3: if $R_x(K) = 0 \mod q$ then
4:     Go To 1 and select a new $k$
5: Let $r = H(m)k + R_x(K)x_c \mod q$
6: Output $Sig = \{K, r\}$
```

3.6 AMC Tracing of Malicious Agents

In the WZW protocol [12], the authors recognized the problem *what often appears to be an honest user is really dishonest and such a user may release a malicious agent.* In order to discourage malevolent behavior, the host upon the realization of malevolent agent activity by the AGENT, can submit the AGENT's V_{cert} to the AMC so that AMC can determine the identity ID_{user} of the user who constructed the agent AGENT, and then penalize the user in a appropriate manner.

Recall that during the registration process the AMC stored ID_C, Y_C, A_C and ω_C into its database, which we denote by $\mathcal{D}$, for each registered user C. To determine the identity ID_{user} of the user who constructed the malicious agent, the AMC performs the following algorithm. Observe that if the host followed the "authenticate agent procedure", then they checked the congruence relationship found in step 1 of Algorithm 3, thus the conditional in step 1 should always be false. This tracing algorithm is inefficient since the trusted party AMC must look at each entry in the database until it finds the user's identity.

Algorithm 3. AMC Trace the User

```
Input: AGENT's $V_{cert} = (z, \alpha, \beta, \lambda)$
Output: $ID_C$, $Y_C$, $A_C$ and $\omega_C$
1: if $\lambda G \neq H_0(\beta + y_c H(M)) + zP_{AMC}$ then
2:     return Invalid $V_{cert}$
3: for each $A_C \in$ AMC database $\mathcal{D}$ do
4:     Compute $\frac{z}{R_x(A_c)}\alpha$
5:     if $\frac{z}{R_x(A_c)}\alpha = \beta$ then
6:         return The user who released AGENT is $ID_C$, $Y_C$, $A_C$ and $\omega_C$
7: return The user who released the AGENT is not in the AMC database
```

4 Attacking the WZW Protocol

Wang et. al. in [12] relied heavily on the argument that to defeat their scheme one had to break the ECDLOG problem. However, as we will now establish this is certainly not true. Recall the data structure of an AGENT is described in Figure 1. We now provide a number of attacks and observations concerning the WZW protocol.

4.1 Masquerading Attack–Malicious User on Honest User Attack

Whenever Alice wants to purchase a product, there are a number of hosts that her agent will be migrate to. All of them will know the contents of her agent data structure (the hosts know this information but they do not know Alice's identity). Further, adopting the Dolev-Yao security model [8], all content on the network is visible to all parties. Consequently there are many parties that will know the content of the agent's data structure, all of theses parties possess M, V_{cert}, x_C and y_C.

Suppose Eve is a malicious host who is also a user of the mobile agent system. Further, suppose Eve has acquired the data structure of a mobile agent that was release by another party, which is denoted by M_C, $V_{cert,C}$, x_C and y_C. Eve now constructs a mobile agent, she selects her agent's instructions, routing table, etc. which we denote by M_{eve}. Eve now proceeds to form the rest of the data structure for her mobile agent AGENT_{eve}, in particular $V_{cert,eve}$, x_{eve} and y_{eve}. Eve sets $x_{eve} = x_C$ and $y_{eve} = y_C$, she sets $z_{eve} = z_C$, $\beta_{eve} = \beta_C$ and $\alpha_{eve} = \alpha_C$. She then must compute λ_{eve} which must satisfy equation (2). Clearly Eve is not using her registered public key Y_{eve} and her AMC generated A_{eve}. Observe that Eve knows λ_C where $\lambda_C = \gamma_c t + (\psi t - X_c t + x_c H(M))H_0 \mod q$, but Eve does not know t, X_C, and ψ, because these parameters belong to user C. In order to satisfy equation (2), the value λ_{eve} will need to satisfy this equation $\lambda_{eve} = \gamma_c t + (\psi t - X_c t + x_c H(M_{eve}))H_0 \mod q$. Then

$$\lambda_{eve} - \lambda_C = x_c H_0(H(M_{eve}) - H(M)) \mod q \tag{3}$$

and observe that Eve knows each term on the right-hand side of equation (3). Therefore Eve can compute $\lambda_{eve} - \lambda_C$ and so she computes $\lambda_{eve} = \lambda_C + (\lambda_{eve} - \lambda_C)$. Now Eve's data structure for her AGENT_{eve} is $\{M_{eve}$, Dynamic data, $V_{cert,eve}$ $\{y_{eve}$, α_{eve},β_{eve},λ_{eve},$z_{eve}\}$, $\{y_{eve}$,$x_{eve}\}\}$

Thus Eve's data structure will satisfy the authentication step as described in equation (2). Also note that Eve has modified the agent's instructions from the original instructions (user C's original instructions were given in M where M was defined by user C). Further, Eve has modified the lists of hosts to visit (she can modify the routing table). Consequently Eve can place malicious instructions/malware in M_{eve}. Now when a host complains to the AMC about the AGENT_{eve}, they will send to AMC the certificate $V_{cert,eve}$. When the AMC uses Algorithm 3, the AMC will compute for each $A_i \in$ Database $\mathcal{D}$ the point $\frac{z_{eve}}{R_x(A_i)}\alpha_{eve}$ and compare it to β_{eve}. Of course $z_{eve} = z_C$, $\alpha_{eve} = \alpha_C$ and $\beta_{eve} = \beta_C$. Consequently the AMC will determine that user C has released the malicious AGENT_{eve} and so Eve has successfully masqueraded as user C and this has gone undetected. This attack is devastating to the WZW protocol, because it shows that the WZW protocol lacks any identity integrity.

It is clear that Wang et. al. [12] rely on the assumption that the only way to form a λ that will be authenticated requires that λ must be well-formed and follow the definition given in Algorithm 1, which is false. Another illustration of the weakness of the original WZW protocol is as follows. Here we describe a simple modification of a V_{cert} that would go undetected. Consider a $V_{cert} = \{z, \alpha, \beta, \lambda, y_C\}$, and the one-off keys x_C and y_C. The parameter β need not be of the form $t(A_C + T)$, in order for λ to satisfy the authentication procedure. We will illustrate this with an example.

Example 1. Consider a $V_{cert} = \{z, \alpha, \beta, \lambda, y_C\}$. Let $\lambda' = \lambda + b$ where $b \in \mathbb{F}_q^*$. Let $\beta' = \beta + H_0^{-1} \cdot bG$, $V'_{cert} = \{z, \alpha, \beta', \lambda', y_C\}$ then $\lambda' G = (\gamma_c t + (\psi t - X_c t + x_c H(M_{eve}))H_0)G + bG$. Consequently
$\lambda' G = H_0(\beta + H_0^{-1} \cdot bG + y_c H(M)) + zP_{AMC} = H_0(\beta' + y_c H(M)) + zP_{AMC}$. So it is easy to reuse another user's authentication parameter λ.

4.2 Violating the Trustee AMC's Tracing

If Alice wants to purchase a product using a mobile agent then Alice issues mobile agent AGENT$_{alice}$. Now if Alice is malicious and would like to construct her AGENT$_{alice}$ to include malware to tamper with a host then the potential repercussions that Alice will bear is that the host will complain to the AMC, which would then trace Alice by using Algorithm 3, and then Alice would be punished in some manner. It is this psychology that discourages a dishonest party from behaving maliciously. However if Alice could somehow avoid tracing, then Alice may issue malicious agent AGENT$_{alice}$. Let M_{alice} denote the set of instructions, routing table, etc. Alice then computes her V_{cert}, she selects $\psi \in_R \mathbb{F}_q^*$, computes $T = \psi G$, sets $t = R_x(T) \mod q$, sets $z_{alice} = tR_x(A_{alice})$, selects $x_{alice} \in_R \mathbb{F}_q^*$, compute $y_{alice} = x_{alice} G$, computes $\lambda = \gamma_{alice} t + (\psi t - X_{alice} t + x_{alice} H(M_{alice}))H_0 \mod q$, and computes $\beta = tA_{alice} + tT$. However Alice then selects $\alpha \in_R$ *prime subgroup of order q of E*. Now that the mobile agent data structure is complete, Alice releases the mobile agent AGENT$_{alice}$.

Clearly AGENT$_{alice}$ will satisfy equation (2). Thus AGENT$_{alice}$ will be authorized by any host it was sent to. However if AGENT$_{alice}$ included any malware, then when the host complains to the AMC, the AMC will be unsuccessful in tracing Alice because $\beta \neq t\alpha$ (since α is a random EC point) which is necessary for the WZW protocol to provide tracing. This situation can occur because no host can check the condition $\beta \overset{?}{=} t\alpha$, because if t was ever provided to a host then $R_x(A_{alice})$ would be revealed which violates the anonymity of the user. This attack will be much more difficult to repair than the other attacks we have discussed because some type of proof concerning the relationship between β and α must be conducted, while maintaining the anonymity of the user.

4.3 Attack – Agent's Signature of W_{bid}

The following discussion concerns the signature Sig generated on the host's W_{bid}. Here we list a series of observations concerning potential vulnerabilities and inadequacies. Depending on the application, the actual agent instructions (that the host executed), and the W_{bid}, one may be able to take advantage of the vulnerabilities we raise to construct an attack. First, the Sig as defined in the WZW protocol only takes as input the m which is the hash $H(ID_w, W_{bid})$, thus there is no binding of the agent's original set of instructions (i.e. its data structure M) to this signature. Now the agent must provide the host all parameters to produce the agent's signature, but the only parameter used is x_C, which is a weakly authenticated parameter. In the agent-to-host authentication $\lambda G = H_0(\beta + y_c H(M)) + zP_{AMC}$, the term $x_C H(M)$ can easily be replaced with another term, thus the binding of the agent to the W_{Bid} via Sig (by its use of x_C) is weak at its best. That is, m (host's bid/output) and M agent's instructions (which generated the host bid m) should be "bound together more securely".

A signature is important, no user should be able to deny a signature, and no one should be able to forge a signature. However, in the WZW protocol nothing in the signature binds the V_{cert} and the message M to the signature, the user can deny the signature. Moreover, any host (even those that are not in the routing table) can generate a K and an r randomly and forge a signature for the user. Also observe that the routing table (if it is fixed in the message M) is not bound in the signature.

5 Constructing a Secure Mobile Agent System That Supports Anonymity

The structure of our mobile agent system will be similar to the structure that Wang et. al. introduced in [12]. There will be three types of parties: users, the trusted party $\mathcal{TTP}$ and servers. The goal is to develop a protocol that allows the user to release mobile agents to execute on servers, such that the agents can authenticate to the server, establishing that they are owned by users that have registered to the trusted party. Central to the theme of the protocol is that there is NO active/on-line requirements for the $\mathcal{TTP}$ to intervene during the release of a mobile agent by a single user. Another central theme should be that the anonymity of the user should be protected throughout the protocol. We first discuss modifications that need to be made to the WZW protocol notation, and state our security assumptions needed throughout the protocol.

We will denote elliptic curve points using uppercase characters, scalars will be denoted by lowercase letters or Greek letters. If $P = (x_0, y_0)$ is an elliptic curve point where $P \neq \mathcal{O}$, we will let $R_x(P)$ denote the projection of the point P onto its x-coordinate and $R_y(P)$ denote the projection of the point P onto its y-coordinate. Thus $R_x(P) = x_0$ and $R_y(P) = y_0$. The user will be denoted by C, the trusted third party by $\mathcal{TTP}$ instead of AMC and the server/host by $\mathcal{S}$.

5.1 Security Assumptions

We will assume that the *elliptic curve discrete log problem* (ECDLOG) is hard. The ECDLOG problem is such that given an elliptic curve E, with a prime subgroup of order q, point P belonging to this prime subgroup with $P \neq \mathcal{O}$, and the scalar multiple kP, then it is "computationally infeasible" to determine k. We will be utilizing a multiplicative group $\mathbb{G}_2$. We will assume that the discrete log problem is "hard" in $\mathbb{G}_2$. Several arguments in this paper will deal with the knowledge or the calculation of the *discrete log* and/or the *elliptic curve discrete log*. The discrete log of aP base P is a, which we will denote by $DLOG_P(aP) = a$.

In addition, we will be using a bilinear map e, we assume that the discrete log is "hard" even with the presence of the bilinear map, that is given generator G, a scalar multiple aG and $e(G, aG)$ it is "hard" to compute the scalar a. We will assume that *Computational Diffie-Hellman (CDH)* problem is a hard problem. Moreover that in both $\mathbb{G}_1$ and $\mathbb{G}_2$, CDH is infeasible. Implying that a bilinear map isn't useful for solving the CDH problem. The CDH problem is such that given aG, bG, and $e(aG, bG)$ it is hard to compute ab. The *Decision Diffie-Hellman (DDH)* problem is the problem that: concerning whether one can distinguish between G, aG, bG, cG and G, aG, bG, abG. The *Decision Diffie-Hellman (DDH)* problem is "easy" in $\mathbb{G}_1$ due to the existence of the bilinear map e. We will be working in an algebraic setting described by Boneh et. al, [2] as the *Gap Diffie-Hellman (GDH)* group. The additive group $\mathbb{G}_1$ is called a GDH group if DDH is easy in $\mathbb{G}_1$ but CDH is hard. We will assume that $\mathbb{G}_1$ is a GDH group.

5.2 Setup

The $\mathcal{TTP}$ does the following:

An elliptic curve E is generated. $\mathbb{G}_1$ is a subgroup of E of order q, where q is prime. G is a generator for the subgroup $\mathbb{G}_1$. $\mathbb{G}_2$ is a multiplicative group of order q and $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a bilinear map. Let H denote secure cryptographic hash function. The $\mathcal{TTP}$ publishes E, $\mathbb{G}_1$, $\mathbb{G}_2$, G and H. The $\mathcal{TTP}$ selects $s \in_R \mathbb{Z}_q^*$ randomly. The $\mathcal{TTP}$ publishes sG. Each entity $\mathcal{C}$ in the system establishes a private key $k_{\mathcal{C}}$ and public key $Y_{\mathcal{C}} = k_{\mathcal{C}}G$. This includes the $\mathcal{TTP}$. Thus the private key for the $\mathcal{TTP}$ is $k_{\mathcal{TTP}}$ and the public key is $Y_{\mathcal{TTP}}$. A common term that we will employ throughout our protocol will be $H_0 = H(R_x(Y_{\mathcal{TTP}})||R_y(Y_{\mathcal{TTP}}))$, recall H is a hash function, $R_x(Y_{\mathcal{TTP}})$ and $R_y(Y_{\mathcal{TTP}})$ are the x and y coordinates of $Y_{\mathcal{TTP}}$ and $||$ will denote concatenation. Each entity $\mathcal{C}$ possesses an identity $ID_{\mathcal{C}}$, we assume that this identity has been bound to their public key $Y_{\mathcal{C}}$ by a certificate generated by a trusted party *Certificate Authority (CA)*.

5.3 Our Mobile Agent Data Structure

The first item is the formation of λ as described in step 9 of Algorithm 1. Recall the masquerade attack described in Section 4.1. This attack allowed a user Eve to utilize another user C's $V_{cert,C}$. Moreover Eve was able to construct her own set of instructions and modify the routing table. If Eve was merely performing a replay attack on user C then this is solved by the timestamp that is included in M. So the problem is that Eve can construct her own M_{eve} yet reuse much of user C's virtual certificate $V_{cert,C}$. The reason this attack can be performed is the definition of $\lambda = \gamma_C t + (\psi t - k_C t + xH(M))H_0 \mod q$, each factor of the term $xH(M)H_0$ is known by Eve. Hence she can replace this term by the term $xH(M_{eve}))H_0$ to use and help her form λ_{eve}. Thus in our protocol we will make the parameter x private and not part of the mobile agent data structure. Of course this decision will have implications in other areas of the protocol.

To address the attack on $\mathcal{TTP}$ tracing as described in Section 4.2 we will need to provide a mechanism that will prove that the mobile agent belongs to an authenticated user, and that the agent is using the parameters of this user, the authentic parameters that the $\mathcal{TTP}$ has generated for them. Further, the agent needs to prove to the server that the $\mathcal{TTP}$ can trace the owner of the mobile agent, in cases where the agent acts “poorly” or “maliciously” the owner/user needs to be able to be traced by the $\mathcal{TTP}$, this will be future work and will be considered in future work.

5.4 User Key Generation and Registering with the $\mathcal{TTP}$

This aspect of the protocol is identical to what Wang et. al. described in [12]. Each user C will generate a private key k_C and compute the corresponding public key $Y_C = k_C G$. To register with the $\mathcal{TTP}$ so that C becomes an authentic member of the mobile agent system, user C sends ID_C and Y_C. The $\mathcal{TTP}$ generates A_C, ω_C and γ_C, as described in the WZW protocol. So $\omega_c \in \mathbb{F}_q^*$ and A_C is computed by $A_c = \omega_c G - Y_c$. Lastly γ_C is computed by $\gamma_c = H_0\omega_c + R_x(A_c)X_{AMC} \mod q$. The $\mathcal{TTP}$ stores (A_c, γ_c, Y_c, ID_c, ω_c) in its database and returns A_c and γ_c to the user. This represents the long term certificate of the user in the sense of its use within the mobile agent system. The user C can verify that the certificate from the $\mathcal{TTP}$ by checking that γ_C satisfies the equation (1).

5.5 User Releases a Mobile Agent

Many of the steps that user C performs will be similar to the steps in the WZW protocol, however there will be several extremely important additional steps/parameters unique to our system.

Creating One-Off keys and the Mobile Agent Data Structure
The user creates one-off keys. The user C selects $x \in_R \mathbb{F}_q^*$ and then computes $L = xG$.

The data structure that is included with the mobile agent is very similar to the data structure in the WZW protocol, which was illustrated in Figure 1. This data structure includes M, which contains the user's request (instructions that need to be executed), information base, routing table, time stamp, and other static data. The data structure will also include dynamic data, the Virtual certificate V_{cert} (which will be different than the V_{cert} described in the WZW protocol). We will also include in the data structure the scalar multiple L of the one-off keys BUT WE DO NOT INCLUDE the scalar x (this will remain as a private/secret one-off key for user C and will NOT be made available to anyone).

Creating the Virtual Certificate
The user will need to generate a virtual certificate called V_{cert} much in the same way as the WZW protocol with a few minor changes. The V_{cert} is described as

follows. For the sake of simplicity we will denote $H(\mathcal{D})$ to represent the hash of some of the session parameters in the structure with the message M such that $H(\mathcal{D}) = H(H(M)||H(zG)||H(tG)||H(B)||H(L)||H(tx)||H(tL))$. The use of the hash $H(\mathcal{D})$ will be discussed in the security analysis. Observe the parameter t is calculated differently in our scheme.

Algorithm 4. AGENT$_C$'s Virtual Certificate Generation V_{cert} for static agent data $\mathcal{D}$ by User C

```
Input: User's $A_C, k_C$, one-off keys $x, L$ and static agent data $\mathcal{D}$
Output: $V_{cert} = (zG, tG, B, \lambda, L, tx, tL)$
1: Selects $\psi \in_R \mathbb{F}_q^*$
2: Computes the elliptic curve point $T = \psi G$
3: Selects $t \in_R \mathbb{F}_q^*$, and computes $tG$
4: Computes $z = tR_x(A_C) \mod q$, and $zG$
5: Computes $B = t(A_C + T)$
6: Computes $\lambda = \gamma_C t + (\psi t - k_C t + xH(\mathcal{D}))H_0 \mod q$
7: Computes $tx$ and $tL = txG$
8: Outputs the $V_{cert} = (zG, tG, B, \lambda, L, tx, tL)$
```

Observe that by making the parameter x private, no party can modify the data structure and the corresponding λ. Thus we have defeated one of the attacks outlined in 4.1. Note that this version of V_{cert} contains modified parameters from the V_{cert} that was defined in [12].

5.6 Agent Authentication to Server and Validity Verification

In the agent authentication process, the server first checks the timestamp and the validity period of the agent. Next, the server $\mathcal{S}$ computes λG where λ is included in the virtual certificate V_{cert}. The server $\mathcal{S}$ will use the λG calculation to compute the term $\mathcal{W}$ by.

$$\mathcal{W} = \lambda G - H_0(B + H(\mathcal{D})L). \tag{4}$$

The server will now check if the following relation is valid.

$$e(\mathcal{W}, G) \overset{?}{=} e(zG, Y_{\mathcal{TTP}}) \tag{5}$$

If equation (5) is satisfied, then $\mathcal{W} = zY_{\mathcal{TTP}}$. Note that zG is published in the V_{cert}, and is thus known to the server.

If all the steps are valid, then the agent is authenticated. We will discuss the security of this authentication technique in Section 6.

5.7 Signature and Verification

After the server $\mathcal{S}$ has executed the agent's instructions, a bid/output $\mathcal{S}_{bid}$ is generated. The server signs the output using its private key producing $Sign_{\mathcal{S}}(m)$ where $m = H(H(D)||ID_{\mathcal{S}}||\mathcal{S}_{bid})$. Since the agent is an intermediary for user C, the agent needs to sign the bid. The signature $Sig = \{K, r\}$ where the server $\mathcal{S}$ allows the intermediary agent to select $k \in_R \mathbb{F}_q^*$ and compute $K = kG$. The agent also computes, $r = H(H(m)||H(D)||H(V_{cert,C}))k + R_x(K)tx \mod q$. Recall that $V_{cert,C}$ contains both tx and tL. Observe that the mobile agent instructions, the $\mathcal{S}_{bid}$ and the authentication material $V_{cert,C}$ have been bound into the signature. The Algorithm 5 addresses the verification of the signature.

Algorithm 5. Signature Verification

```
Input: $m = H(ID_{\mathcal{S}}, \mathcal{S}_{bid})$, data structure $\mathcal{D}$, $V_{cert,C}$
Output: TRUE or FALSE
 1: if $Sign_{\mathcal{S}}(m)$ is not a valid signature produced by server $\mathcal{S}$ then
 2:     Return FALSE
 3: if the routing table is provided in $M$ and identity $ID_{\mathcal{S}}$ routing table then
 4:     Return FALSE
 5: if Authentication of $V_{cert,C}$ fails then
 6:     Return FALSE
 7: if $e(tL, G) \neq e(L, tG)$ then
 8:     Return FALSE
 9: Compute $rG$
10: if $rG = H(H(m)||H(D)||H(V_{cert,C)})K + R_x(K)tL$ then
11:     Return FALSE
12: else
13:     Return TRUE
```

5.8 Tracing

We do not claim that we can trace all users who release mobile agents. We will demonstrate here that the $\mathcal{TTP}$ can trace users who are honest and follow the protocol honestly. The only value of demonstrating this tracing is that it establishes the importance of showing that no user can masquerade/replay parameters generated by another user. This will be discussed in Section 6.

Suppose that user U has acted honestly when releasing their mobile agent. Then $\mathcal{TTP}$ can determine the user U by:

Algorithm 6. Tracing by $\mathcal{TTP}$

```
Input: $V_{cert} = (zG, tG, B, \lambda, L, tx, tL)$
Output: user $U$ who released the mobile agent
1: for all user $C$ in $\mathcal{TTP}$ database do
2:     if $R_X(A_C) \cdot tG \overset{?}{=} zG$ then
3:         Return user $C$
4: Return cannot determine
```

6 Security Analysis

Due to space limitations we will briefly outline the security analysis of our protocol. A more complete analysis will be provided in the extended version of the paper.

Authentication. The system should not allow a non-authentic user to pass the authentication step.

Anonymity. The system should protect the identity of an authenticated user.

We now make a series of observations that follow from Algorithm 4 and the hardness of the ECDLOG problem.

Observation 1. *Observe that a user C does not know ω_C. Further, user C does not know the discrete log of A_C base G, which is denoted by $DLOG_G(A_C)$.*

Observation 2. *Consider a user U who has computed wG and wY_{ttp} then user U knows some scalar a, with $a \neq 1$, and points Q_1 and Q_2 such that $DLOG_{Q_1} wG = a = DLOG_{Q_2} wY_{ttp}$*[2]*.*

This observation will be applied in the context of proving the security of our protocol. Thus when we use the term *computing* we imply the user has not borrowed or replayed two points that were generated by another user. The above result is trivial if we allowed a to equal 1, since the user could replay two points generated by another user. In the case that $a = w$ then in the context of our protocol the user U appears to be acting honestly.

Of course, the correctness of this observation does not necessarily imply the user knows w nor does it prevent the user U from replaying two points generated by another user. That is, user U could borrow (replay) two points bG and bY_{ttp} from another user U', by selecting $a \in \mathbb{Z}_q$ and computing $a(bG)$ and $a(bY_{ttp})$, in this case $w = ab$ and $Q_1 = bG$ and $Q_2 = bY_{ttp}$. In this situation the user U would not know w.

The correctness of this observation follows from the difficulty of the discrete log problem, in the sense that the user U needs to compute two points wG and wY_{ttp} that share a discrete log relation, i.e. $DLOG_G wG = w = DLOG_{Y_{ttp}} wY_{ttp}$.

In the following, the term *non-authentic user* is referring to a user who has not registered and does not possess a A_C and a γ_C that is generated by the $\mathcal{TTP}$.

Observation 3. *It is computationally hard for a non-authentic user U to compute λ such that $\lambda G = Q + wY_{\mathcal{TTP}}$, where w and Q are known by the user U.*

Note that an authentic user C can be considered to be non-authentic (in the sense of the above observation) if they attempt to engage in a mobile agent transaction and decide not to use their secret registration parameters A_C and γ_C that were generated by the $\mathcal{TTP}$.

Observation 4. *Observe that the hash function $H(\mathcal{D})$ contains the message M, which includes a timestamp and a period of validity. $\mathcal{D}$ contains other parameters of the V_{cert} Further, the parameters t and ψ are random, secret and vary for each mobile agent that is generated.*

Observation 5. *Observe that user C's one-off private key x is secret. Thus any user U, where $U \neq C$, cannot modify the mobile agent data structure $\mathcal{D}$ that was*

[2] By the use of the term computing, we imply that user U is not using (replaying) two points that were generated by another user.

created by user C to some $\mathcal{D}'$, with $\mathcal{D}' \neq \mathcal{D}$, since user U would need x, to compute $xH(\mathcal{D}') - xH(\mathcal{D})$ to add to λ in order to compute λ'. [3]

First consider the **authentication aspect** of the protocol. Recall that in the authentication process, the server $\mathcal{S}$ computes λG and calculates $\mathcal{W}$. Then the server checks the pairing $e(\mathcal{W}, G) \overset{?}{=} e(zG, Y_{\mathcal{TTP}})$.

By a series of arguments we can prove the following Theorem (the proof will be provided in the extended version of the paper).

Theorem 1. *It is infeasible for a user U to masquerade as user C where $U \neq C$.*

When we say server $\mathcal{S}$ *authenticates the mobile agent* we are referring that the series of steps described in Section 5.6 are completed and verified, in particular equation (5) must be satisfied. Recall that a *non-authentic user* we are referring to a user who has not registered and does not possess a A_C and a γ_C that is generated by the $\mathcal{TTP}$.

Theorem 2. *Suppose server $\mathcal{S}$ authenticates a mobile agent then server $\mathcal{S}$ knows that the user that created the* AGENT *is authentic.*

Now consider the **privacy (anonymity) aspects** of the protocol. Those parameters which can be used to trace the identity of a user are k_C, A_C and γ_C. Observe tx, tL, tG, L possess no sensitive materials. The sensitive materials are k_C, A_C and γ_C. The only parameters that consist of sensitive materials are zG, B, and λ where each of them are masked with randomness, independently selected (in some cases they are protected by more than one mask). For example, zG is masked by t. B is masked by T, and λ is masked by ψ. Thus user C's identity is protected and they remain anonymous.

Consider the security of the **signature/verification scheme** of the protocol. There are two important properties in signing. A user can initially act nonmalicious when contacting the server, but can later decide to become malicious and deny its signature for example. The second property is that NO ONE other than the user and its intermediary agent should be able to construct a signature. In our protocol we solved these signature problems by requiring additional verification steps, where anyone can verify using these steps. If the identity of the server $ID_{\mathcal{S}}$ is not in the routing table of the message M, then the server is not an authentic server and is not the right intermediary server. The verifier should check the V_{cert} of the user, and if it's not authentic then the verifier caught a bad user. We also introduced a step that checks the pairing $e(tL, G) \neq e(L, tG)$ where this pairing binds one-off keys x and L. Lastly, the verifier computes and checks rG as before.

7 Open Problems and Conclusion

In the WZW protocol, when an agent visits a server, if the server doubts that this agent is malicious, the server submits V_{cert} and Sig to the AMC to find out

[3] Here $\lambda' = \lambda + xH(\mathcal{D}') - xH(\mathcal{D})$.

the agent owner's identity. AMC has a database record of $(A_i, \gamma_i, Y_i, ID_i, \omega_i)$. For each user there is a unique record, however if there are many users, it is very difficult to identify the owner of the mobile agent, thus this remains as an open problem. Thus the original AMC tracing algorithm that was described in WZW protocol [12] would require a search of the entire database. An open problem is to find a tracing algorithm for which the search requires $O(1)$ time.

Wang et. al [12] thought their protocol enables the AMC to trace, however we demonstrated in our attacks that the AMC is incapable of tracing malicious users. Future work should support tracing of malicious users that try not to follow the protocol correctly. Non-malicious users should remain anonymous through the protocol.

References

1. Algesheimer, J., Cachin, C., Camenisch, J., Karjoth, G.: Cryptographic security for mobile code. In: SP 2001: Proceedings of the IEEE Symposium on Security and Privacy, pp. 2–11. IEEE Computer Society, Los Alamitos (2001)
2. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
3. Borselius, N.: Mobile agent security. Electronics and Communication Engineering Journal 14, 211–218 (2002)
4. Boukerche, A., Ren, Y.: A Novel Solution based on Mobile Agent for Anonymity in Wireless and Mobile Ad hoc Networks. In: Proceedings of the 3rd ACM Workshop on QoS and Security for Wireless and Mobile Networks, pp. 86–94 (2007)
5. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM, 84–88 (1981)
6. Chaum, D.: The dining cryptographers problem: unconditional sender and recipient untraceability. Journal of Cryptography, 65–75 (1988)
7. Claessens, J., Preneel, B., Vandewalle, J.: (How) can mobile agents do secure electronic transactions on untrusted hosts? ACM Transactions on Internet Technology (TOIT) 3(1), 28–48 (2003)
8. Dolev, D., Yao, A.C.: On the security of public key protocols. In: Proceedings of the IEEE 22nd Annual Symposium on Foundations of Computer Science, pp. 350–357 (1981)
9. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation 48, 203–209 (1987)
10. Reed, M., Syverson, P., Goldschlag, D.: Proxies for anonymous routing. In: Proceeding of the 12th Annual Computer Security Applications, pp. 95–104 (1995)
11. Saxena, A., Soh, B.: A novel method for authenticating mobile agents with one-way signature chaining. In: Proceedings of The 7th International Symposium on Autonomous Decentralized Systems (ISADS 2005), pp. 187–193 (2005)
12. Wang, C.J., Zhang, F.G., Wang, Y.: Secure Web Transaction with Anonymous Mobile Agent over Internet. Journal of Computer Science and Technology 18(1), 84–89 (2003)
13. Zwierko, A., Kotulski, Z.: Mobile Agents: Preserving Privacy and Anonymity. In: Bolc, L., Michalewicz, Z., Nishida, T. (eds.) IMTCI 2004. LNCS (LNAI), vol. 3490, pp. 246–258. Springer, Heidelberg (2005)

A Formal Language for Specifying Complex XML Authorisations with Temporal Constraints

Sean Policarpio and Yan Zhang

Intelligent Systems Laboratory
School of Computing and Mathematics
University of Western Sydney
Penrith South DC, NSW 1797, Australia
{spolicar,yan}@scm.uws.edu.au

Abstract. The Extensible Markup Language (XML) is utilised in many Internet applications we are using today. However, as with many computing technologies, vulnerabilities exist in XML that can allow for malicious and unauthorised use. Applications that utilise XML are therefore susceptible to security faults if they do not provide their own methods. Our research focuses on developing a formal language which can provide access control to information stored in XML formatted documents. This formal language will have the capacity to reason if access to an XML document should be allowed. Our language, $\mathcal{A}^{xml(T)}$, allows for the specification of authorisations on XML documents based on the popular Role-based Access Control model. Temporal interval reasoning is the study of logically representing time intervals and relationships between them. As part of our research, we have also included this aspect in our language $\mathcal{A}^{xml(T)}$ because we believe it will allow us to specify even more powerful access control authorisations.

Keywords: AI in computer security, AI in database, logic programming, knowledge representation and reasoning, access control, authorisations, XML databases and security.

1 Introduction

Many applications utilise the Extensible Markup Language [9] as a tool to store and retrieve information. However, the guarantee that information stored in XML documents is secure and is only accessible by authorised users is not possible unless an external method is used. XML does not have any inherent security methods as part of its specification [9]. An XML document is essentially a formatted plain text file that can be freely viewed and edited. Therefore there is a demand for methods in which access to XML documents can be controlled.

In this paper, we present our work on the development of a formal language that will provide access control to XML documents. We incorporate the XML query language, XPath [8], into our formal language so that we can define which documents (or elements within a document) we would like to restrict access to.

An XPath is a string representation of traversing through an XML document to return an element within the document. For example, the following is an XPath that follows the tree-like structure of a document to return the element *author*:

```
/library/books/book/author
```

Our formal language uses the Role-based Access Control model [15] as a basis for the structure of authorisations to subjects. This primarily means rather than applying authorisations directly to subjects, we create "roles" that can have one or more specified authorisations. This gives us better control over which subjects have what authorisations. It also allows us to include features like separation of duty and conflict resolution directly into the language [15].

Finally, we include Allen's Temporal Interval Relationship logic [1]. Allen's temporal relationships cover all possible ways in which intervals can relate to one another (such as *before*, *meets*, *equal*, etc.) and are incorporated into the syntax of our formal language. We include this aspect of temporal reasoning in our language so that we can specify time constraints on authorisations to designate when they should be applied.

We utilise the formal language to produce a *security policy base*. The policy base contains all the $\mathcal{A}^{xml(T)}$ rules of authorisation for the XML documents requiring access control. The policy base can be reasoned upon to determine which authorisations should be followed.

The rest of this paper is organised as follows. Section 2 presents the formal syntax of our language $\mathcal{A}^{xml(T)}$, illustrates its expressive power through various XML access control scenarios, and defines queries on XML policy bases. Section 3 describes the semantics of language $\mathcal{A}^{xml(T)}$ based on its translation to a logic program under answer set semantics. In section 4, an example is also presented to show the application of $\mathcal{A}^{xml(T)}$ in XML authorisation specification and reasoning. Section 5 briefly discusses the related work. Finally, Section 6 concludes the paper with some remarks.

2 Formal Language $\mathcal{A}^{xml(T)}$

Our language, $\mathcal{A}^{xml(T)}$, consists of a finite set of predicate statements. These statements are used to create various rules in a security policy base. We present the syntax of our language in Backus-Naur Form (Table 1) with a definition of each element. The statements are written from the point of view of the policy base writer or *admin*. This single subject represents the author of the access control policies.

2.1 Syntax

A *rule* is a conditional statement that allows the policy writer to specify a predicate statement to be validated based on the truth of other predicates. Rules include nonmonotonic reasoning derived through the absence of predicates. Our language also includes *deny rule* statements which are for specifiying conditional states that should never be allowed.

Table 1. BNF for $\mathcal{A}^{xml(T)}$

```
                   <rule> ::= <head-statement> [ if [ <body-statements> ] [ with absence
                              <body-statements> ] ]
              <deny-rule> ::= admin will deny [ if [ <body-statements> ] [ with absence
                              <body-statements> ] ]
         <head-statement> ::= <relationship-statement> | <grant-statement> |
                              <request-statement> | <auth-statement> |
                              <role-statement>
        <body-statements> ::= <body-statement> | <body-statement>, <body-statements>
         <body-statement> ::= <relationship-statement> | <grant-statement> |
                              <request-statement> | <auth-statement> |
                              <role-statement>
 <relationship-statement> ::= admin says <relationship-atom>
        <grant-statement> ::= admin grants <role-name> to <subject> during
                              <temporal-interval>
      <relationship-atom> ::= below( <role-name>, <role-name> ) |
                              separate( <role-name>, <role-name> ) |
                              during( <temporal-interval>, <temporal-interval> ) |
                              starts( <temporal-interval>, <temporal-interval> ) |
                              finishes( <temporal-interval>, <temporal-interval> ) |
                              before( <temporal-interval>, <temporal-interval> ) |
                              overlap( <temporal-interval>, <temporal-interval> ) |
                              meets( <temporal-interval>, <temporal-interval> ) |
                              equal( <temporal-interval>, <temporal-interval> )
                <subject> ::= <subject-constant> | <subject-variable>
              <role-name> ::= <role-name-constant> | <role-name-variable>
      <temporal-interval> ::= <temporal-interval-constant> | <temporal-interval-variable>
      <request-statement> ::= admin asks is <subject> a member of <role-name>
                              during <temporal-interval>
         <auth-statement> ::= admin says that <subject> can use the <role-atom>
                              during <temporal-interval>
         <role-statement> ::= admin creates <role-atom>
              <role-atom> ::= role( <role-name>, <sign>, <xpath-statement>, <privilege> )
                   <sign> ::= + | -
        <xpath-statement> ::= in <document-name>, return <xpath-expressions>
          <document-name> ::= <document-name-constant> | <document-name-variable>
      <xpath-expressions> ::= <xpath-node> | <xpath-node>, <xpath-expressions>
             <xpath-node> ::= [ / ] <node-name> [<xpath-predicate>] /
              <node-name> ::= <node-name-constant> | <node-name-variable> | * | //
        <xpath-predicate> ::= <child-node-name> <predicate-relationship> <variable-value> |
                              <attribute-name> <predicate-relationship> <variable-value>
        <child-node-name> ::= <child-node-name-constant> | <child-node-name-variable>
         <attribute-name> ::= <attribute-name-constant> | <attribute-name-variable>
 <predicate-relationship> ::= < | > | =
              <privilege> ::= read | write
```

The *head-statement* from a *rule* consists of the predicate statements that will be validated true if the rules conditions are true as well. The head-statement itself can either be one of five statements; a *relationship-statement*, *grant-statement*, *request-statement*, *auth-statement*, or *role-statement*.

The *body-statement(s)* of a *rule* are the conditions that are reasoned upon to validate the *head-statement*. These are also made up of the same five statements used in the *head-statement*.

A *relationship-statement* confirms that some relationship between two objects in the security policy base is true. These relationships are represented by those predicate symbols found under the *relationship-atom*. There are a few *relationship-atoms* available that can be used in *relationship-statements*. Relationships for example could be hierarchical (below), mutually exclusive (separate), or be based on Allen's Temporal Interval relationships (during, starts, meets, etc.) [1].

The *role-statement* creates an access control role. The *role-atom* used in the statement includes a *role-name*, a *sign* which represents either positive or negative access to the object in question, an *xpath-statement* to identify an XML object, and finally the *privilege* that can be performed on the object.

An *xpath-statement* in $\mathcal{A}^{xml(T)}$ is a formal representation of an XPath expression. These expressions include the primary features of the syntax of XPath, such as single node queries, tree-like structured queries, wildcard queries, and predicate filters on nodes and attributes [8].

Grant-statements serve the purpose of assigning an access control *role* to a *subject* (a person requiring authorisation). This statement also includes a temporal argument to specify when the roles authorisation should be applied.

A *request-statement* is used when a query for subject authorisation is made. It represents the policy writers attempt to discover if a particular *subject* is a member of a *role* at a specific *temporal-interval.*

Auth-statements specify that a *subject* who has been previously granted a *role* now has authorisation to access an object. We create rules in the policy base that will validate these statements by checking if a subject has positive authorisation to a role and that there are no conflicting rules. If these are true, then an *auth-statement* is created.

2.2 Expression Examples with $\mathcal{A}^{xml(T)}$

In this section, we demonstrate utilising our formal language to express some common relationships and rules for a security policy base.

Creating a temporal interval relationship. The policy base writer specifies that the interval *morning_tea* is before *afternoon_tea* and that the interval *play_time* meets *nap_time*:

```
admin says before(morning_tea, afternoon_tea).
admin says meets(play_time, nap_time).
```

XML elements and attributes. Using the *xpath-statement*, an arbitrary element named *cleaning_log* with the child element *cleaning_area* from the document "database.xml" can be represented like this:

```
in database.xml, return cleaning_log/cleaning_area
```

The policy writer can also specify more in the XPath by using predicates or wildcards. This *xpath-statement* uses the wildcard (*) to specify a single step between the elements *cleaning_information* and *cleaning_log*. The policy writer also uses a predicate expression to filter *cleaning_area*'s that have the attribute *type* equal to *office.*

```
in database.xml, return /janitor_logs/cleaning_information/*/cleaning_log/
cleaning_area[@type="office"]
```

Role Creation, Role Relationships, and Granting Authorisations. The policy writer creates the *janitor* role. This role is allowed to *read* the element specified in our XPath from the previous example.

```
admin creates role(janitor, +, in database.xml, return /janitor_logs/
cleaning_information/*/cleaning_log/cleaning_area[@type="office"], read).
```

The policy writer specifies relationship statements between roles. They can state that the role *staff* is below the role *manager*, or in other words, is a child role, and that they also be mutually exclusive by specifying that they be separate.

```
admin says below(staff, manager).
admin says separate(staff, manager).
```

The policy writer adds a subject to a roles membership. They add the subject *tyler* to the role *janitor*. He will be able to access this role only during the *afternoon* temporal interval.

```
admin grants janitor to tyler during afternoon.
```

Here, the policy writer creates a complex rule stating that if any subject is a member of the role *janitor* during any time, then they should also be a member of the role *window_washers* during the same interval. The interval must also finish at the same time as *maintenance_time*. They add the condition that the subject also **not** be a member of the *electrician* role.

```
admin grants window_washer to SubX during TimeY
   if admin grants janitor to SubX during TimeY,
   admin says finishes(TimeY, maintenance_time),
   with absence admin grants electrician to SubX during TimeZ.
```

The *deny rule* is useful for specifying rules where the validity of the *body-statements* are not desired. A deny rule can be written to indicate that *patrick* should never be a member of the role *janitor* during any interval.

```
admin will deny if admin grants janitor to patrick during TimeY.
```

Request Statements. The policy writer can query if a subject is a member of a role at a specific time. They will check if *taro* is a member of the *musician* role during *rehearsals*.

```
admin asks is taro a member of musicians during rehearsals.
```

2.3 Producing Authorisations and Querying the XML Policy Base

With a security policy base written in $\mathcal{A}^{xml(T)}$, it is possible to find which subjects have authorisations to what objects based on the roles they have been granted membership to. To do this, we reason upon statements that have been written in the policy base. The subject authorisations are found with a rule we refer to as the *authorisation rule*.

```
admin says that SUBJECT can use role(ROLE-NAME, +, XPATH, PRIVILEGE) during INTERVAL
   if admin grants ROLE-NAME to SUBJECT during INTERVAL,
   admin asks is SUBJECT a member of ROLE-NAME during INTERVAL,
   admin creates role(ROLE-NAME, +, XPATH, PRIVILEGE),
   with absence role(ROLE-NAME, -, XPATH, PRIVILEGE)
```

This rule is written to pertain to all *grant-statements*. It ensures that a role be postively authorised for use by a subject only if it does not conflict with a possible negative role with the same privileges and temporal interval (*conflict resolution* [15]). If this rule produces an *auth-statement*, that is the indication that the subject in question does in fact have authorisation based on those specified in the *role-statement*.

There are other defined rules like this that apply to many aspects of our formal language that must also be reasoned upon before authorisation is given to a subject. We refer to these as *language rules* within $\mathcal{A}^{xml(T)}$. They are discussed in more depth in the formal semantics of our language and are defined in two groups:

– **Role-based Access Control Rules** are included to ensure that features of the model are present (ie. separation of duty, conflict resolution, role propagation) and that authorisations are generated when querying the policy base (ie. the *authorisation rule*).
– **Temporal Interval Relationship Reasoning Rules** allow for defined temporal intervals to adhere to the relationships defined in Allen's work [1].

By using $\mathcal{A}^{xml(T)}$ to define a security policy base, we now have a determinable way to reason who has authorisation to what XML objects based on facts about subject privileges. However, to produce these authorisations and to also prove that our policy base written in $\mathcal{A}^{xml(T)}$ is satisfiable, we need a method to compute a result. To do this, we provide the semantics of our language in the form of an answer set program.

3 Semantics

We chose answer set programming [4] as the basis for our semantics because it provides the reasoning capabilities to compute the authorisations defined using our formal language. If properly translated, we can use an ASP solver (such as *smodels* [18]) to find which authorisations will be validated true. What we want to produce is an answer set that will have the same results as those produced from our formal language $\mathcal{A}^{xml(T)}$. We first present the alphabet of our ASP based language $\mathcal{A}_{LP}$ and then its formal semantics.

3.1 The Language Alphabet $\mathcal{A}_{LP}$

Entities Subjects, temporal intervals, role names, role properties, XPaths, and XPath properties make up the types of entities allowed in the language. These can either be constant or variable entities, distinguished by a lowercase or uppercase first letter respectively.

Function symbols

– `role(role-name, sign, xpath(), priv)`, where *role-name* is the name of this role, *sign* is a + or − depending on if the role is allowing or disallowing a privilege, *xpath* is an xpath function representing an element(s) from an XML document, and *priv* is the privilege that can be performed on the object (ie. read or write).

- `node(name, id, level, doc)`, represents a node in an XML document, where `name` is the name of that node (element), `id` is a distinct key in the document, `level` represents its hierarchical placement, and `doc` the document it originates from.
- `xpred(axis, query)`, represents an XPath predicate, where *axis* is the location of the node to apply the predicate *query* on.
- `xpath(node(), xpred())`, represents an XPath, consisting of a `node()` and `xpred()`.

Predicate symbols The first set of symbols are used for representing relationships between roles and temporal intervals. Their definitions are taken directly from $\mathcal{A}^{xml(T)}$.

```
below(role-name2, role-name1)
separate(role-name2, role-name1)
during(tempint2, tempint1)
starts(tempint2, tempint1)
finishes(tempint2, tempint1)
before(tempint2, tempint1)
overlap(tempint2, tempint1)
meets(tempint2, tempint1)
equal(tempint2, tempint1)
```

These next set of symbols are used for defining and querying authorisations in the policy base and are also similar to their $\mathcal{A}^{xml(T)}$ equivalents.

```
grant(subject, role-name, tempint)
request(subject, role-name, tempint)
auth(subject, xpath(), priv, tempint)
```

A new predicate symbol is introduced in $\mathcal{A}_{LP}$ for conflict resolution reasoning on subject authorisations.

- `exist_neg(subject, xpath(), priv, tempint)` states that at least one negative `grant` for a *subject* exists.

And finally, four predicates are also introduced for providing relationships between XML nodes.

- `isNode(node())`, indicates that the *node()* function exists.
- `isParent(node`$_2$`(), node`$_1$`())`, means *node*$_2$ is the parent or is hierarchically above *node*$_1$, where both are node functions.
- `isLinked(node`$_2$`(), node`$_1$`())`, means *node*$_2$ can be reached directly (is descended) from *node*$_1$, where both are node functions.
- `isAttr(attr_name, node())`, means *attr_name* is an XML attribute of the *node* function

In most cases, with an understanding of $\mathcal{A}^{xml(T)}$, the transformations and meanings of symbols and rules from $\mathcal{A}_{LP}$ are self explanatory. However, extra consideration must be given to the method in which XPaths are handled in $\mathcal{A}_{LP}$.

3.2 Handling XPaths in $\mathcal{A}_{LP}$

There was a problem handling particular XPaths in $\mathcal{A}_{LP}$ because our formal language implements features and syntax of XPath that are difficult to translate and use in answer set programming. Specifically, this is the use of wildcards and predicate queries. We refer to these problematic XPaths as *dynamic* XPaths because they can represent zero to many XML nodes, as opposed to *static* XPaths which can only refer to zero or one node. When dynamic XPaths are used in a logic program, it is difficult to specify which nodes should have authorisation rules applied to them. This is because the nodes that are meant to be represented by the XPath are not yet known. With static XPaths, it is clearly stated what node requires authorisation.

We consider the approach by Almendros-Jimenez et al. [2] which described a method to represent the structure and data of an XML document as a logic program. Using a similar technique, we provide new rules that *rewrite* dynamic XPaths into static ones. The idea of *query rewriting* is to use the structure of XML documents to understand and define what dynamic XPaths might mean [13,14].

These concepts are what make up the majority of the language rules for *XPath Translation* in the policy base. We introduced the functions `node`, `xpred`, and `xpath`, and the predicate symbols `isNode`, `isParent`, `isLinked` and `isAttr` into $\mathcal{A}_{LP}$ for this reason. These various new functions and predicates allow us to write rules to satisfy different kinds of dynamic XPaths. However, due to space constraints, we will show only one example of an XPath transformation rule.

XPath Tranformation Rule. For this example, we will assume that the structure (schema) of the XML documents are already defined using the `node` function and predicate statements introduced earlier.

This rule will determine the meaning of a wildcard (*) in an XPath like this `/A/*/C`. It will produce an `xpath` that will return the element C which can only be reached from the parent element A.

```
xpath(node(C, ID3, 3, DOC), xpred(self, ‘‘’’)) ←
   isParent(
   node(X, ID2, 2, DOC),
   node(C, ID3, 3, DOC)),
   isParent(
   node(A, ID1, 1, DOC),
   node(X, ID2, 2, DOC)).
```

The rule determines if an arbitrary node, X, is the parent of node C and is the child of node A. In each `node`, we specify the `level` and use variables for the `id` and `document`. This allows for the rule to determine all possible XPaths that satisfy the relationship conditions. For those `nodes` that satisify the rule, we can produce an XPath function for C that we can guarantee is meant to be produced from the XPath `/A/*/C`.

Remarks. It is important to note that these rules are not part of our semantics, which will be presented in the next section. This is because the process of transforming the XPaths occur at an intermediate level that is not concerned with the reasoning of the security policy base. Also, because these rules are written specific to varying XPaths, it is not possible to give a formal definition of their translation. We consider the transformations of XPaths as happening before the translation of an $\mathcal{A}^{xml(T)}$ policy base into $\mathcal{A}_{LP}$.

3.3 Formal Definitions

We define the semantics of our formal language by translating $\mathcal{A}^{xml(T)}$ into an answer set logic program. We refer to this translation as *Trans*. A *policy base*, D_A, is a finite set of rules and/or deny rules, ψ, written in $\mathcal{A}^{xml(T)}$ as specified in Table 1. The generic rules, or *language rules*, for the same policy base, D_A, are a finite set of statements, α, written in $\mathcal{A}^{xml(T)}$.

α contains statements to provide:

- conflict resolution,
- separation of duty,
- role propagation,
- temporal interval relationship reasoning, and
- authorisation reasoning

Definition 1. *Let D_A be a policy base. We define $Trans(D_A)$ to be a logic program translated from D_A as follows:*

1. *for each rule or deny rule, ψ, in D_A, $Trans(\psi)$ is in $Trans(D_A)$*
2. *for each statement α in D_A, $Trans(\alpha)$ is in $Trans(D_A)$*

A translated *rule* or *deny rule*, $Trans(\psi)$, has the same form as those defined in Gelfond's Stable Model Semantics [17] and answer set programming [4]. A translated *rule* has the following form:

```
Trans(head-statement)_k ←
   Trans(body-statement)_k+1 ,...,
   Trans(body-statement)_m ,
   not Trans(body-statements)_m+1 ,...,
   not Trans(body-statements)_n .
```

A translated *deny rule* has the same form except for the dismissal of the *head-statement.*

The conflict resolution rules in α are located in the *authorisation rule* (Section 2.3). In $Trans(\alpha)$, conflict resolution rules are transformed into a new rule that checks if a subject has at least one negative grant for a role. We use this to reason if a conflict with a positive grant is possible. In $\mathcal{A}_{LP}$, exist_neg was introduced for this purpose. The translated rule is as follows:

```
exist_neg(S, xpath(node(N, I, L, D), xpred(A, Q)), P, T) ←
   grant(S, R, T),
   role(R, -, xpath(node(N, I, L, D),
   xpred(A, Q)), P).
```

Separation of duty in α is translated with a simple deny rule:

```
← grant(S, R1, T1), grant(S, R2, T2), separate(R2, R1).
```

If grant predicates are giving a subject membership to roles that are stated as being separate, then the statement should be denied and the logic program faulted.

Role propagation in α is also translated similarly in $Trans(\alpha)$ with two generic rules. The original rules were (1.) to do with transitivity between roles and (2.) for propagation of role properties. The propagation rule searches for roles that are hierarchically related (using `below`) and copies the authorisation properties from the parent role to the descendent role. Their translation is as follows:

1. `below(R1, R3) ← below(R1, R2), below(R2, R3).`

2.
   ```
   role(R1, Si, xpath(node(N, I, L, D), xpred(A, Q)), P) ←
   below(R1, R2), role(R2, Si,
   xpath(node(N, I, L, D), xpred(A, Q)), P)
   ```

α contains numerous rules that pertain to temporal interval relationship reasoning. Again, many of these rules are transformed from $\mathcal{A}^{xml(T)}$ to $\mathcal{A}_{LP}$ trivially. We show some of these rules from $Trans(\alpha)$ [1]:

Temporal Interval Bounded Rule:

This rule searches for an interval that is contained within another but does not overlap the beginning or end of the outer interval.

```
during(T4, T1) ←
   starts(T2, T1), finishes(T3, T1),
   before(T2, T4), before(T4, T3).
```

Temporal Interval Containment Rule:

If the temporal interval used in a `grant` predicate is found to have an interval contained within it, then a similar `grant` will be applied to the subject for that contained interval.

```
grant(S, R, T2) ←
   grant(S, R, T1), during(T2, T1).
```

Implicit Temporal Interval Relationships:

These rules apply some implied temporal relationships we have choosen to implement for our own purposes. Namely, that `starts` and `finishes` should imply `during` and that `meets` should imply `before`.

```
during(T2, T1) ← starts(T2, T1).
during(T2, T1) ← finishes(T2, T1).
before(T2, T1) ← meets(T2, T1).
```

Finally, the *authorisation rule* (Section 2.3) in $\mathcal{A}^{xml(T)}$ is translated in $Trans(\alpha)$ as follows:

```
auth(S, xpath(node(N, I, L, D), xpred(A, Q)), P, T) ←
   request(S, R, T), grant(S, R, T),
   role(R, +, xpath(node(N, I, L, D),
   xpred(A, Q)), P),
   not exist_neg(S, xpath(node(N, I, L, D),
   xpred(A, Q)), P, T).
```

A query on D_A, ϕ, written in $\mathcal{A}^{xml(T)}$ is a request statement, as specified in Table 1, and its translation, $Trans(\phi)$, is a `request` predicate from $\mathcal{A}_{LP}$.

Definition 2. *Let ϕ be a query on a policy base D_A written in $\mathcal{A}^{xml(T)}$. We define $Trans(\phi)$ as the translation of the request statement from $\mathcal{A}^{xml(T)}$ to $\mathcal{A}_{LP}$.*

[1] All rules can be found in original manuscript

An answer from a query ϕ is denoted as π and has the form of an authorisation statement, specified in Table 1, while its translation, $Trans(\pi)$, is an auth predicate from $\mathcal{A}_{LP}$.

Definition 3. *Let π be the answer from a query ϕ on policy base D_A written in $\mathcal{A}^{xml(T)}$. We define $Trans(\pi)$ as the translation of the authorisation statement from $\mathcal{A}^{xml(T)}$ to $\mathcal{A}_{LP}$.*

We define the relationship between our formal language and its translation into the semantics of answer set programming.

Definition 4. *Let D_A be a policy base, ϕ a query on it, and π the answer from that query. We say D_A entails π, or $D_A \models \pi$, iff for every answer set, S, of the logic program $Trans(D_A)$ with the query $Trans(\phi)$, $Trans(\pi)$ is in S.*
$D_A \models \pi$ iff $Trans(D_A) \models Trans(\pi)$

4 An Example

We will demonstrate the creation of a security policy for a scenario requiring access control to XML documents. Due to space constraints, we will only have a very small example and also forgo XPath's utilising predicates.

Scenario Description. A hospital requires the implementation of an access control model to protect sensitive information it stores in a number of XML documents. We will discuss roles created for two particular subjects at the hospital.

Hospital Roles. An *administration* role in the hospital will have access to read two nodes named *board_minutes* and *financial_info* from a document named *board_db*. Roles that are below the *administration* role will also inherit this rule. For example, a role named *board_member* will inherit these privileges. However, we will also include within the *board_member* role the privilege to write to the document.

The role *admin_doctor* will be able to write to the *board_minutes* section of the *board_db* document.

In our policy base, we will allow the *admin_doctor* role to read and write to a few other documents. They will have access to read a *staff_contact_info* document and both read and write to the *patient_db* and *doctor_db* documents.

Table 2 shows these roles written in $\mathcal{A}^{xml(T)}$.

Policy Base Subjects and Rules. Within our case scenario, we will focus on two subjects, Lucy and Rita, both administrative doctors. Lucy will utilize the privileges of the *admin_doctor* role for a single specific interval while Rita must be active in that same role at any other time.

The XML access control rules in which these subjects must abide to in the hospital are as follows.

Table 2. Hospital Roles

```
Administration
 admin creates role(administration, +, in board_db, return /, read).
 admin says below(board_member, administration).
 admin creates role(board_member, +, in board_db, return /, write).
Administrative doctors
 admin says below(admin_doctor, administration).
 admin creates role(admin_doctor, +, in board_db, return /board_minutes, write).
 admin creates role(admin_doctor, +, in staff_contact_info, return /, read).
 admin creates role(admin_doctor, +, in patient_db, return /, read).
 admin creates role(admin_doctor, +, in patient_db, return /, write).
 admin creates role(admin_doctor, +, in doctor_db, return /, read).
 admin creates role(admin_doctor, +, in doctor_db, return /, write).
```

For Lucy, we can specify that she has the *admin_doctor* role on some arbitary interval *tuesday*. We could state some relationships for this interval like so:

```
admin says meets(monday, tuesday).
admin says meets(tuesday, wednesday).
```

Now we can make a rule that states that Lucy be active with the *admin_doctor* role on *tuesday* and Rita be active during any other time. In this case, possibly *monday and wednesday* or any other interval defined in the policy base.

```
admin grants admin_doctor to lucy during tuesday.
admin grants admin_doctor to rita during INT_I
   if with absence admin grants admin_doctor to lucy during INT_I.
```

4.1 Logic Program Translation

With a completed policy base, we can translate all of the $\mathcal{A}^{xml(T)}$ rules into an $\mathcal{A}_{LP}$ answer set program. For our case study, we will demonstrate the translation of our policies for Lucy and Rita.

Role Translations From the defined roles, we will show the translation of one of the $\mathcal{A}^{xml(T)}$ rules for the *admin_doctor*.

This rule was orignally written in $\mathcal{A}^{xml(T)}$ to specify that the *admin_doctor* role be allowed to write to the *board_minutes* node in the *board* database. In $\mathcal{A}_{LP}$ it is written as:

```
role(admin_doctor, +, xpath(node(/board_minutes, ID, 0, board_db), xpred(self, ‘‘’’)),
write).
```

As mentioned earlier, we will forgo the explanation of the translation of XPaths. However, briefly, this XPath represents the /board_minutes node, with any *ID*, at the top-level (0) of the *board_db* document.

Grant Translations We will now translate some rules granting subjects membership to roles. We take a look at the rule that specified that Lucy be granted the *admin_doctor* on the interval *tuesday* and that Rita have it at any other interval. In $\mathcal{A}_{LP}$, they are translated as:

```
meets(monday, tuesday).
meets(tuesday, wednesday).
grant(lucy, admin_doctor, tuesday).
grant(rita, admin_doctor, I) ← not grant(lucy, admin_doctor, I).
```

4.2 Experimenting with Queries on the $\mathcal{A}_{LP}$ Program

We now present authorisation queries on our policies. We will show the queries and results in $\mathcal{A}^{xml(T)}$ and $\mathcal{A}_{LP}$.

We first create requests of authorisation for Lucy and Rita to obtain the role *admin_doctor* during the temporal interval *tuesday*.

In $\mathcal{A}^{xml(T)}$, we would write request-statements like so:

```
admin asks is lucy a member of admin_doctor during tuesday.
admin asks is rita a member of admin_doctor during tuesday.
```

In $\mathcal{A}_{LP}$, those statements are translated into the following `request` statements:

```
request(lucy, admin_doctor, tuesday).
request(rita, admin_doctor, tuesday).
```

If we were to logically reason upon our $\mathcal{A}^{xml(T)}$ policy base with the previous request-statements, the following statements would be produced by the *authorisation rule* (Section 2.3).

```
admin says that lucy can use role(admin_doctor, +, in board_db, return /, read) during tuesday.

admin says that lucy can use role(admin_doctor, +, in board_db, return /board_minutes, write)
during tuesday.

admin says that lucy can use role(admin_doctor, +, in staff_contact_info, return /, read) during
tuesday.

admin says that lucy can use role(admin_doctor, +, in patient_db, return /, read) during tuesday.
admin says that lucy can use role(admin_doctor, +, in patient_db, return /, write) during tuesday.
admin says that lucy can use role(admin_doctor, +, in doctor_db, return /, read) during tuesday.
admin says that lucy can use role(admin_doctor, +, in doctor_db, return /, write) during tuesday.
```

When our translated policy base and `request` predicates are computed in a stable model solver, an answer set containing the following equivalent facts is generated.

```
auth(lucy, xpath(node(/, idbdb00, 0, board_db), xpred(self, ‘‘’’), read, tuesday).

auth(lucy, xpath(node(/board_minutes, idbdb01, 0, board_db),
xpred(self, ‘‘’’), write, tuesday).

auth(lucy, xpath(node(/, idscidb00, 0, staff_contact_info),
xpred(self, ‘‘’’), read, tuesday).

auth(lucy, xpath(node(/, idpdb00, 0, patient_db), xpred(self, ‘‘’’), read, tuesday).
auth(lucy, xpath(node(/, idpdb00, 0, patient_db), xpred(self, ‘‘’’), write, tuesday).
auth(lucy, xpath(node(/, idddb00, 0, doctor_db), xpred(self, ‘‘’’), read, tuesday).
auth(lucy, xpath(node(/, idddb00, 0, doctor_db), xpred(self, ‘‘’’), write, tuesday).
```

Because Rita's rule specifies that she cannot be a member of *admin_doctor* at any interval that Lucy is, her requests for authorisation do not generate any results. Lucy however retrieves all the authorisations that an *admin_doctor* should. Referring to Table 2, we can identify all the *admin_doctor* roles that are used to create the auth-statements and `auth` predicates above.

5 Related Work

Damiani et al. [11,12] provided some essential work in the field of XML access control. In [12], a fine-grained access control model is discussed. This model takes an XML document and designates access rights on each element. They provide implementations of rule propagation and also include features such as both positive and negative authorisations and conflict resolution in the model. Through their algorithm, a source XML document can be processed by removing all objects of negative authorisation and returning a document with only elements that are allowed to be viewed [12].

Crampton [10] utilised the role-based access control model for specific use with XML documents. He utilises the same object-based approach by using the XPath query language. However, Crampton points out that his work only focuses on reading XML documents [10].

In [5,7], Bertino et al. discussed their own implementation of an access control system for XML documents. Their work does follow the role-based access control model to a certain extent (however, we did not see methods for role propagation or separation of duty). Subjects are granted authorisation through credentials and objects are specified through XPath's [5,7]. The implementation includes features such as the propagation of policy rules and conflict resolution. Bertino et al. include in their formalisation temporal constraints based on their previous work in [6]. However, their approach seems restricted in terms of handling XPath expressions in authorisation reasoning.

Besides Bertino et al., only a small group of other researchers have produced research utilising logic programming for XML policy base descriptions [3,16]. To the best of our knowledge, a logic-based formal language for XML authorisations has not yet been developed with the inclusion of temporal constraints, the complete role-based access control model, and nonmonotonic reasoning capabilities of answer set programming.

6 Conclusion

In this paper, we presented a formal language of authorisation for XML documents. We demonstrated its expressive power to provide role-based access control with temporal constraints. We provided a semantic definition through the translation of the high level language into an answer set program. Finally, we gave a brief example of defining a security policy base in $\mathcal{A}^{xml(T)}$, translating it into an $\mathcal{A}_{LP}$ logic program, and then computing the authorisations from it.

Acknowledgement. This research is supported in part by an Australian Research Council Linkage grant LP0883646.

References

1. Allen, J.F.: Towards a general theory of action and time. Artif. Intell. 23(2), 123–154 (1984)
2. Almendros-Jiménez, J.M., Becerra-Terón, A., Enciso-ba, F.J.: Nos. Querying xml documents in logic programming*. Theory Pract. Log. Program. 8(3), 323–361 (2008)

3. Anutariya, C., Chatvichienchai, S., Iwaihara, M., Wuwongse, V., Kambayashi, Y.: A rule-based xml access control model. In: Schröder, M., Wagner, G. (eds.) RuleML 2003. LNCS, vol. 2876, pp. 35–48. Springer, Heidelberg (2003)
4. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press, Cambridge (2003)
5. Bertino, E., Braun, M., Castano, S., Ferrari, E., Mesiti, M.: Author-x: A java-based system for xml data protection. In: IFIP Workshop on Database Security, pp. 15–26 (2000)
6. Bertino, E., Bettini, C., Ferrari, E., Samarati, P.: An access control model supporting periodicity constraints and temporal reasoning. ACM Trans. Database Syst. 23(3), 231–285 (1998)
7. Bertino, E., Carminati, B., Ferrari, E.: Access control for xml documents and data. Information Security Technical Report 9(3), 19–34 (2004)
8. The WWW Consortium. Xml path language (xpath) version 1.0. (1999), `http://www.w3.org/TR/xpath`
9. The WWW Consortium. Extensible markup language (xml) 1.0 (fifth edition) (November 2008), `http://www.w3.org/TR/REC-xml`
10. Crampton, J.: Applying hierarchical and role-based access control to xml documents. In: SWS 2004: Proceedings of the 2004 Workshop on Secure Web Wervice, pp. 37–46. ACM, New York (2004)
11. Damiani, E., De Capitani Vimercati, S., Paraboschi, S., Sarnarati, P.: Securing xml documents. In: Zaniolo, C., Grust, T., Scholl, M.H., Lockemann, P.C. (eds.) EDBT 2000. LNCS, vol. 1777, pp. 121–135. Springer, Heidelberg (2000)
12. Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P.: A fine-grained access control system for xml documents. ACM Trans. Inf. Syst. Secur. 5(2), 169–202 (2002)
13. De Capitani di Vimercati, S., Marrara, S., Samarati, P.: An access control model for querying xml data. In: SWS 2005: Proceedings of the 2005 Workshop on Secure Web Services, pp. 36–42. ACM, New York (2005)
14. Fan, W., Chan, C., Garofalakis, M.: Secure xml querying with security views. In: SIGMOD 2004: Proceedings of the 2004 ACM SIGMOD International Conference on Management Data. ACM Press, New York (2004)
15. Ferraiolo, D.F., Cugini, J.A., Richard Kuhn, D.: Role-based access control (rbac): Features and motivations. In: 11th Annual Computer Security Applications Proceedings (1995)
16. Gabillon, A.: A formal access control model for xml databases. In: Jonker, W., Petković, M. (eds.) SDM 2005. LNCS, vol. 3674, pp. 86–103. Springer, Heidelberg (2005)
17. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Kowalski, R.A., Bowen, K. (eds.) Proceedings of the Fifth International Conference on Logic Programming, pp. 1070–1080. The MIT Press, Cambridge (1988)
18. Niemelä, I., Simons, P., Syrjänen, T.: Smodels: a system for answer set programming. In: Proceedingsof the 8th International Workshop on Non-Monotonic Reasoning (April 2000)

Author Index