Dongho Won
Seungjoo Kim (Eds.)

Information Security and Cryptology – ICISC 2005

LNCS 3935

8th International Conference
Seoul, Korea, December 2005
Revised Selected Papers

Springer

Lecture Notes in Computer Science 3935

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

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
University of Dortmund, Germany

Madhu Sudan
Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos
University of California, Los Angeles, CA, USA

Doug Tygar
University of California, Berkeley, CA, USA

Moshe Y. Vardi
Rice University, Houston, TX, USA

Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany

Dongho Won Seungjoo Kim (Eds.)

Information Security and Cryptology – ICISC 2005

8th International Conference
Seoul, Korea, December 1-2, 2005
Revised Selected Papers

Springer

Volume Editors

Dongho Won
Seungjoo Kim
Sungkyunkwan University
School of Information and Communication Engineering
Information Security Group
300 Cheoncheon-dong, Jangan-gu, Suwon-si, Gyeonggi-do 440-746, Korea
E-mail: {dhwon, skim}@security.re.kr

Library of Congress Control Number: 2006924114

CR Subject Classification (1998): E.3, G.2.1, D.4.6, K.6.5, F.2.1, C.2, J.1

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-33354-1 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-33354-8 Springer Berlin Heidelberg New York

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

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2006
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11734727 06/3142 5 4 3 2 1 0

Preface

The 8th International Conference on Information Security and Cryptology was organized by the Korea Institute of Information Security and Cryptology (KIISC) and sponsored by the Ministry of Information and Communication of Korea (MIC). This conference aims at providing a forum for the presentation of new results in research, development, and application in information security and cryptology. This is also intended to be a place where research information can be exchanged.

The conference received 192 submissions from 29 countries, and the Program Committee selected 32 (from 15 countries) of these for presentation. The conference program includes two invited lectures by David Naccache on "National Security, Forensics and Mobile Communications" and by Shigeo Tsujii on "Information Security as Interdisciplinary Science Based on Ethics".

We would like to thank the many researchers from all over the world who submitted their work to this conference. The submission review process had two phases. In the first phase, Program Committee members compiled reports and entered them, via Web interface, into Web Review software. In the second phase, committee members used the software to browse each other's reports, discuss, and update their own reports. We are extremely grateful to the Program Committee members for their enormous investment of time and effort in the difficult and delicate process of review and selection. Moreover, we would like to thank all the authors who submitted papers to ICISC 2005 and the authors of accepted papers for their preparation of camera-ready manuscripts. Last but not least, we thank our students, Junghyun Nam, Yunho Lee, Jin Kwak, Younggyo Lee, and Seokhyang Cho, who helped us during the whole process of preparation for the conference.

February 2006

Dongho Won
Seungjoo Kim

Organization

General Chair

Dae Ho Kim	NSRI, Korea

Program Committee Co-chairs

Dongho Won	Sungkyunkwan University, Korea
Seungjoo Kim	Sungkyunkwan University, Korea

Program Committee

Giuseppe Ateniese	The Johns Hopkins University, USA
Tuomas Aura	Microsoft Research, UK
Alex Biryukov	Katholieke Universiteit Leuven, Belgium
John Black	University of Colorado, USA
Liqun Chen	Hewlett-Packard Labs, UK
Jung Hee Cheon	Seoul National University, Korea
Kyo-il Chung	ETRI, Korea
Jean-Sebastien Coron	University of Luxembourg, Luxembourg
Ed Dawson	Queensland University of Technology, Australia
Alexander W. Dent	Royal Holloway, UK
Anand Desai	NTT MCL, USA
Yevgeniy Dodis	New York University, USA
Gerhard Eschelbeck	Qualys, USA
Serge Fehr	CWI, The Netherlands
Pierre-Alain Fouque	École Normale Supérieure, France
Eiichiro Fujisaki	NTT Labs, Japan
Juan A. Garay	Bell Laboratories, USA
Marc Girault	France Telecom, France
Philippe Golle	Palo Alto Research Center, USA
Dieter Gollmann	TU Hamburg, Germany
SangGeun Hahn	KAIST, Korea
Yongfei Han	ONETS, China
Goichiro Hanaoka	University of Tokyo, Japan
Markus Jakobsson	Indiana University of Pennsylvania, USA
Marc Joye	Gemplus Card International, France
Jonathan Katz	University of Maryland, USA
Hiroaki Kikuchi	University of Tokai, Japan
Hyoung-Joong Kim	Kangwon National University, Korea
Kwangjo Kim	Information and Communications University, Korea

Kaoru Kurosawa	Ibaraki University, Japan
Taekyoung Kwon	Sejong University, Korea
Young-Bin Kwon	Chung-Ang University, Korea
Chi Sung Laih	National Cheng Kung University, Taiwan
Kwok-Yan Lam	Tsinghua University, China
Dong Hoon Lee	Korea University, Korea
Sang-Ho Lee	Ewha Womans University, Korea
Arjen Lenstra	Lucent Technologies' Bell Laboratories, USA, and Eindhoven University of Technology, The Netherlands
Yingjiu Li	Singapore Management University, Singapore
Helger Lipmaa	Cybernetica AS & University of Tartu, Estonia
Javier Lopez	University of Malaga, Spain
Masahiro Mambo	University of Tsukuba, Japan
Keith Martin	Royal Holloway, University of London, UK
Mitsuru Matsui	Mitsubishi Electric Corporation, Japan
Chris Mitchell	Royal Holloway, University of London, UK
Atsuko Miyaji	JAIST, Japan
SangJae Moon	Kyungpook National University, Korea
Yi Mu	University of Wollongong, Australia
Jesper Buus Nielsen	Aarhus University, Denmark
DaeHun Nyang	Inha University, Korea
Rolf Oppliger	eSECURITY technologies, Switzerland
Carles Padro	Technical University of Catalonia, Spain
Raphael Chung-Wei PHAN	Swinburne University of Technology, Malaysia
Josef Pieprzyk	Macquarie University, Australia
Vincent Rijmen	Graz University of Technology, Austria
Rei Safavi-Naini	University of Wollongong, Australia
Kouichi Sakurai	Kyushu University, Japan
Palash Sarkar	Indian Statistical Institute, India
Nigel Smart	University of Bristol, UK
JungHwan Song	Hanyang University, Korea
Willy Susilo	University of Wollongong, Australia
Tsuyoshi Takagi	Future University, Hakodate, Japan
Wen-Guey Tzeng	National Chiao Tung University, Taiwan
Guilin Wang	Institute for Infocomm Research, Singapore
William Whyte	NTRU Cryptosystems, USA
Michael Wiener	Cryptographic Clarity, Canada
Chuan-Kun Wu	Chinese Academy of Sciences, China
Shouhuai Xu	The University of Texas at San Antonio, USA
Sung-Ming Yen	National Central University, Taiwan
Yongjin Yeom	NSRI, Korea
Moti Yung	Columbia University, USA

Organizing Committee Chair

Heekuck Oh	Hanyang University, Korea

Organizing Committee

Dowon Hong	ETRI, Korea
Daewoon Jeon	KIISC, Korea
Ji Hong Kim	Semyung University, Korea
Jung-Tae Kim	Mokwon University, Korea
Kyung Sim Kim	KIISC, Korea
Young Sub Koo	MIC, Korea
HyungWoo Lee	Hanshin University, Korea
Sangjin Lee	Korea University, Korea
Dong Gue Park	Soonchunhyang University, Korea
Sangwoo Park	NSRI, Korea
Kyung Hyune Rhee	Pukyong National University, Korea
Sang Uk Shin	Pukyong National University, Korea
Yoojae Won	KISA, Korea

External Reviewers

Joe Choo	Masayuki Abe	Caroline Kudla
Christophe Doche	Heng Swee Huay	Orr Dunkelman
Krystian Matusiewicz	Emily Shen	John Kelsey
Ron Steinfeld	Siu-Leung Chung	Stefan Lucks
Huaxiong Wang	Hongwei Sun	Jorge Nakahara Jr
Christian Rechberger	Martijn Stam	Frederic Valette
Norbert Pramstaller	Dan Page	Ju-Sung Kang
Florian Mendel	Pooya Farshim	Jehong Park
Sanjit Chatterjee	John Malone-Lee	Choong-Hoon Lee
Chien-Ning Chen	David Galindo	DongHoon Lee
Hsi-Chung Lin	Kumar Viswanath	Woong Hee Kim
Xinyi Huang	Steven Galbraith	Yung Hur
Fabien Laguillaumie	Kenny Paterson	Sang Woon Jang
Katja Schmidt-Samoa	John Malone-Lee	ChangKyun Kim
DongGuk Han	Carlos Cid	Hyunrok Lee
Michel Abdalla	Geraint Price	Zeen Kim
Dario Catalano	Maura Paterson	Vo Duc Liem
Sebastien Zimmer	Rui Zhang	Dang Nguyen Duc
Tetsu Iwata	Nuttapong Attrapadung	Jaemin Park
Takeshi Koshiba	Yang Cui	Sungcheol Heo
Hiroki Koga	SeongHan Shin	Youngjoon Seo

Sponsoring Institutions

MIC (Ministry of Information and Communication), Korea
IITA (Institute of Information Technology Assessment), Korea
Solmaze Co., Ltd., Korea
K-Bell Co., Ltd., Korea
HAN Infocomm Co., Ltd., Korea

Table of Contents

System/Network Security

Block/Stream Ciphers (I)

Block/Stream Ciphers (II)

Efficient Implementations

Digital Rights Management

Public Key Cryptography

National Security, Forensics and Mobile Communications

David Naccache

Ecole Normale Superieure, France
david.naccache@ens.fr

Abstract. There are nearly 1.5 billion handset users in the world. As the recent attacks in London illustrate, this proliferation of inexpensive mobile phones provides criminals and terrorists with flexible communication means. In this talk we will describe the forensic methodology used for analysing SIM cards and handsets within the French legal context. We will describe specific technical problems applicable to the analysis of cards (e.g., the impossibility to interact with a SIM without altering its internal state), underline seizure details, demonstrate the tools allowing to extract the contents of a handset and a SIM knowing their associated PIN codes. We will overview some of the "heavy" analysis methods (physical reverse-engineering) used with various degrees of success when PIN codes are unknown and describe the protocol allowing to negotiate with the judiciary authorities an evidence destruction risk before undertaking the actual analysis of the SIMs and handsets.

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, p. 1, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Information Security as Interdisciplinary Science Based on Ethics

Shigeo Tsujii

Institute of Information Security, Japan
tsujii@iisec.ac.jp

Abstract. In my definition, the concept of information security is "the dynamic process for establishing an integrated social infrastructure without infringing freedom broadened by information technology and with closer coordination among technologies, administration and management skills, legal and social systems and information morals in order to simultaneously attain efficiency, enhanced security, protected privacy and minimized surveillance over people." I will discuss the paradigm of information security as an interdisciplinary comprehensive science based on information ethics and the way to develop human resources, showing an example of Institute of Information Security where I serve as the president.

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, p. 2, 2006.
© Springer-Verlag Berlin Heidelberg 2006

A Timed-Release Key Management Scheme for Backward Recovery*

Maki Yoshida[1], Shigeo Mitsunari[2], and Toru Fujiwara[1]

[1] Osaka University,
1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
{maki-yos, fujiwara}@ist.osaka-u.ac.jp
[2] u10 Networks,
2-14-8 Fukazawa, Setagaya, Tokyo 158-0081, Japan
herumi@nifty.com

Abstract. The timed-release encryption scheme is to encrypt a message so that a ciphertext can be decrypted when specific time in the future comes. Recently, interesting constructions of the timed-release encryption scheme have been proposed. The central concept of the constructions is a public agent which periodically broadcasts self-authenticated time information, called a time token. A time token contains absolute time information such as "08:09AM Dec. 1, 2005 GMT." A sender encrypts a message so that a receiver of the ciphertext can generate a decryption key from a time token of the designated release time. Although the constructions have many advantages, resilience to missing time tokens is not still satisfactory since a time token can be used only for computing a decryption key of the corresponding time. A promising approach is to construct decryption keys so that a decryption key (e.g., of 08:09AM) can be computed not only from the corresponding time token but also from decryption keys of later time instants (e.g., 08:10AM, 08:11AM and so on). A trivial construction to realize such backward recovery is to use keys, which constitute a hash chain, for encrypting messages and encrypt these keys by using the timed-release encryption scheme. This construction is simple but requires the overhead of encryption. To reduce the overhead, this paper introduces a timed-release key management scheme in which decryption keys are related so that the backward property is provided. The feature is that a sender can choose freely and flexibly the time instants of which decryption keys have the backward property. The paper also gives an efficient construction based on a bilinear map.

1 Introduction

1.1 Background and Motivation

The goal of timed-release encryption is to "send a message into the future," in other words, to encrypt a message so that the receiver cannot decrypt the

* A preliminary version of this paper was published as "Time-Capsule Encryption," IEICE Technical Report, ISEC2004-98, pp.1–5,2004.

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 3–14, 2006.
© Springer-Verlag Berlin Heidelberg 2006

ciphertext until specific time in the future [18]. The problem of timed-release encryption was first mentioned by May in [12] and then discussed by Rivest et al. in [18]. The existing approaches are categorised into two ways [16]:

Time-lock puzzle approach [1, 6, 8, 9, 11, 18]. A sender would encrypt his/her message such that a receiver would have to perform non-parallelizable computation without stopping for the required wait time in order to obtain a decryption key;

Agent-based approach [3, 5, 10, 12, 13, 15, 16, 18, 19]. A sender could use trusted agents and encrypt his/her message such that a receiver will need some secret value, published by the agents on the required time in order to compute a decryption key.

Previous constructions of agent-based approach in [3, 10, 15, 16, 19] are noteworthy. The trusted agent, called the *time server*, only broadcasts self-authenticated time information, called a *time token*, periodically. The time token contains absolute time information, for example "08:09AM Dec. 1, 2005 GMT." A sender encrypts a message by using public information so that the receiver of the ciphertext can generate a decryption key from a time token of the designated time. To do so, the time server generates a time token in the similar way to the short signature scheme proposed in [4], and a sender (resp. a receiver) computes an encryption key (resp. the corresponding decryption key) by using a bilinear map. These constructions have many advantages compared with the others [5, 1, 6, 8, 9, 11, 12, 13, 18]. First, the time server does not need to know who participates as a sender or a receiver, and does not interact with any participant. Secondly, a sender does not need to make any trigger to control decryption of received ciphertexts. Lastly, a receiver does not need computation until the designated time.

However, as mentioned in [3], the constructions in [3, 10, 15, 16, 19] are not resilient to missing time tokens since a time token could only be used to compute the decryption key of the corresponding time. We aim to improve the resilience while keeping the advantages.

Two approaches to the resilience to missing time tokens were given in [3]. One approach is to store all the old time tokens at a public place for the receivers to look up. Then, even if a receiver misses a time token, no problem would cause. However, the overhead becomes large when a release time is designated in minute detail. The other approach is to designate two or more time instants as a release time. A receiver only needs to obtain a time token of one of the designated time instants. However, the overhead of each ciphertext increases proportionally to the number of the designated time instants. Then, the overhead can be also large.

Another promising approach to the resilience is to relate ciphertexts so that ciphertexts could be decrypted when other ciphertexts with later release time could be decrypted. For example, if a ciphertext with release time 08:10AM is decrypted, then a ciphertext with release time 08:09AM received from the same sender can be also decrypted. A receiver receives two or more ciphertexts from the same sender, in many cases. Therefore, such backward recovery improves resilience to missing time tokens enough for practical use.

Compare these three approaches to the resilience to missing time tokens in the point of the resilience and the overhead of required memory. The former two approaches are more resilient, but are less efficient. On the other hand, the last approach is less resilient, but is most efficient. Moreover, by combining the last approach with another approach, we can improve the efficiency while keeping the resilience. For example, by combining the last approach with the second one, the sender only needs to designate additional release time instants for a message with the latest release time. Thus, the overhead of other ciphertexts is reduced.

1.2 Contribution of This Study

We follow the last approach to the resilience to missing time tokens, that is, relating ciphertexts in the timed-release encryption scheme so that the backward property is provided. We can relate ciphertext by combining so-called hash chain technique straightforwardly. Keys which constitute a hash chain are used for encrypting messages. Then, these keys, called message-encryption keys, are encrypted by using the timed-release encryption scheme. This straightforward construction is simple, but requires the overhead of message-encryption keys.

In this paper, to avoid the overhead of message-encryption keys, we relate decryption keys directly so that, for example, a key of 08:09AM can be computed not only from a time token at 08:09AM but also from another key of a later time instant than 08:09AM directly where the related time instants can be freely chosen by the sender.

For simplicity, we concentrate on an essential part of the timed-release encryption scheme. The essential part is generation of time tokens and encryption/decryption keys, that is, management of keys. We give a formal model of the key management scheme, called the backward timed-release key management scheme, and define the security of the scheme by means of probabilistic polynomial time algorithms. Then, we propose a construction to realize the backward timed-release key management scheme, and prove the security formally.

In the proposed construction, decryption keys computed from time tokens are related to constitute a chain structure. A bilinear map is used as in the previous constructions [3, 10, 15, 16, 19], but time tokens are generated by using a different technique. A similar technique is used for generating a signature and a decryption key in the short signature scheme [2] and the traitor tracing scheme [14], respectively.

The proposed construction avoids the overhead of message-encryption keys by relating decryption keys to constitute the hash chain. Moreover, the overhead of memory is the same as that required by the previous constructions which do not provide the backward property.

2 Review of Previous Works

We show the key management part of the previous constructions [3, 10, 15, 16, 19], and discuss the efficiency of relating decryption keys in the previous constructions.

2.1 Preliminaries

The previous constructions [3, 10, 15, 16, 19] and the proposed construction use a bilinear map. We briefly review the necessary facts about bilinear maps.

- $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, \cdot)$ are two cyclic groups of prime order m. The group action in $\mathbb{G}_1$ and $\mathbb{G}_2$ can be computed efficiently.
- $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a bilinear map. The bilinear map e is a map with the following properties:
  - Bilinearity: For all P, Q, $R \in \mathbb{G}_1$,

$$e(P + Q, R) = e(P, R) \cdot e(Q, R),$$
$$e(P, Q + R) = e(P, Q) \cdot e(P, R).$$

    From the bilinearity, it holds that for all $a, b \in \mathbb{Z}_m$, $e(aP, bQ) = e(P, Q)^{ab}$.
  - Non-degeneracy: For a generator of $\mathbb{G}_1$, denoted by P_1, $e(P_1, P_1) \neq 1$.
  - Efficiency: For all P, $Q \in \mathbb{G}_1$, $e(P, Q)$ can be computed efficiently.

2.2 Overview of the Previous Constructions

We show how to generate time tokens and keys of designated release time in the previous constructions. The previous constructions use a collision-free full domain hash function $h : \{\text{time information}\} \to \mathbb{G}_1$.

The cryptographic key of the time server consists of a secret key $sk = a \in \mathbb{Z}_m^*$ and the corresponding public key $pk = \langle P, aP \rangle$ where P is a generator of $\mathbb{G}_1$. The time server generates a time token $ttkn_t$ of time t by $ttkn_t = \langle t, ah(t) \rangle$.

A sender generates a key k_t and a header $header_t$ of designated release time t by

$$k_t = e(r(aP), h(t)), \ header_t = rP,$$

where r is chosen randomly and uniformly from $\mathbb{Z}_m^*$. The header is used for computing k_t from the time token $ttkn_t$. The receiver computes k_t from $header_t = rP$ and $ttkn_t = (t, ah(t))$ by $e(header_t, ah(t)) = e(rP, ah(t))$. The equality $e(r(aP), h(t)) = e(rP, ah(t))$ comes from the bilinearity of e.

In order to relate keys of any different time instants t_1 and t_2 where t_1 is former than t_2, the sender needs to set r_1 and r_2 so that the receiver can compute $k_{t_1} = e(r_1 P, ah(t_1)) = e(P, ah(t_1))^{r_1}$ from $k_{t_2} = e(P, ah(t_2))^{r_2}$ but cannot compute k_{t_2} from k_{t_1}. A possible way is to set $r_1 = h'(k_{t_2})$ where h' is a hash function $h' : \mathbb{G}_2 \to \mathbb{Z}_m$. Since r_1 can be computed from k_{t_2}, the receiver can compute k_{t_1} by the same way as the sender, that is, by $k_{t_1} = e(r_1(aP), h(t_1))$. In this way, the sender can relate keys to constitute the chain structure. However, the receiver needs to apply exponentiation functions defined by different bases $e(aP, h(t_i))$. Therefore, the complexity of computing keys of earlier time is large. This problem comes from the generation of time tokens. Therefore, we generate time tokens in a different way.

3 The Model of the Backward Timed-Release Key Management Scheme

In this section, a formal model and the formal security goal of the *backward timed-release key management scheme* are defined.

In the formal discussion for security, we need to consider an algorithm which "randomly and uniformly" chooses a time instant. The problem is that it is not possible to choose an element "uniformly" from an infinite set. To avoid this issue, we assume that the number of time instants is polynomially upper-bounded. Let $\mathbb{T}$ be the set of time instants where $|\mathbb{T}|$ is in the polynomial-order to the security parameter but large enough to realize applications. Under this assumption, choosing a time instant corresponds to choosing an element from $\mathbb{T}$. For time instants $t_1, t_2 \in \mathbb{T}$, we write $t_1 \prec t_2$ if t_1 is earlier than t_2. We write $t_1 \preceq t_2$ if $t_1 \prec t_2$ or $t_1 = t_2$.

Definition 1. A *backward timed-release key management scheme* is a 6-tuple of polynomial-time algorithms $\mathcal{BTR} = (INIT, TGen, SSGen, SKGen, RKGen, RKbGen)$. The algorithms $INIT$ and $TGen$ are used by the time server to generate time tokens. $SSGen$ and $SKGen$ are used by a sender to generate encryption keys. $RKGen$ and $RKbGen$ are used by a receiver to compute decryption keys. We assume that an encryption key and the corresponding decryption key are symmetric, that is, identical.

- $INIT$ is a probabilistic expected algorithm which takes an integer n (the *security parameter*) as an input, and outputs a pair $\langle pk, sk \rangle$ where pk is the public key and sk is the corresponding secret key for time tokens.
- $TGen$ is an algorithm to determine *time tokens*. Formally, $TGen$ is a deterministic algorithm which takes the key pair $\langle pk, sk \rangle$ and a time instant $t \in \mathbb{T}$, and outputs $ttkn_t$ which is called the *time token* at t.
- $SSGen$ is an algorithm to determine a seed for a sender to control the behaviour of the algorithms $SKGen$, $RKGen$, and $RKbGen$. Formally, $SSGen$ is a probabilistic expected algorithm which takes an integer n (the *security parameter*) as an input, and outputs certain information σ which is called a *private seed*. For example, σ may contain random seeds and/or some cryptographic keys.
- $SKGen$ is an algorithm to determine keys so that the keys can be computed from time tokens and are related to constitute the chain structure. Formally, $SKGen$ is a deterministic algorithm which takes a private seed σ, a public key pk, a set of time instants $T \subseteq \mathbb{T}$, and a time instant $t \in T$, and outputs $k_{t,T}$ and $header_{t,T}$. $k_{t,T}$ is called the *key of release time t in chain T*. $header_{t,T}$ is called the *header of release time t in chain T*. $header_{t,T}$ is used for computing $k_{t,T}$ from the time token $ttkn_t$. We assume that $header_{t,T}$ contains information on t and T.
- $RKGen$ is an algorithm to compute a key of release time t from a time token at t and a header of release time t. Formally, $RKGen$ is a deterministic algorithm which takes a public key pk, a time token $ttkn_t$, and a header $header_{t,T}$, and outputs the key $k_{t,T}$ of release time t in chain T.

– *RKbGen* is an algorithm to compute one key from another key in the same chain backwardly. Formally *RKbGen* is a deterministic algorithm which takes a public key pk, a set of time instants $T \subseteq \mathbb{T}$, time instants t, $t' \in T$, and a key $k_{t',T}$. If $t \preceq t'$, then *RKbGen* outputs a key $k_{t,T}$ of release time t in chain T. Otherwise *RKbGen* outputs a special symbol $\perp$ meaning that the key $k_{t,T}$ cannot be available.

The algorithms must satisfy the consistency: For any $T \subseteq \mathbb{T}$ and t, $t' \in T$ with $t \preceq t'$,

$$k_{t,T} = RKGen(pk, ttkn_t, header_{t,T}) = RKbGen(pk, T, t, t', k_{t',T}),$$

where $\langle k_{t,T}, header_{t,T} \rangle = SKGen(\sigma, pk, T, t)$ and $\langle k_{t',T}, header_{t',T} \rangle = SKGen(\sigma, pk, T, t')$.

Strictly saying, the integer input n for the above algorithms must be given in a unary representation such as 1^n (see [7], for example, for that reason). We write $INIT(n)$ instead of $INIT(1^n)$ just for a notational convenience. ∎

By using the backward timed-release key management scheme, we can realize the timed-release encryption scheme with the backward property as shown in

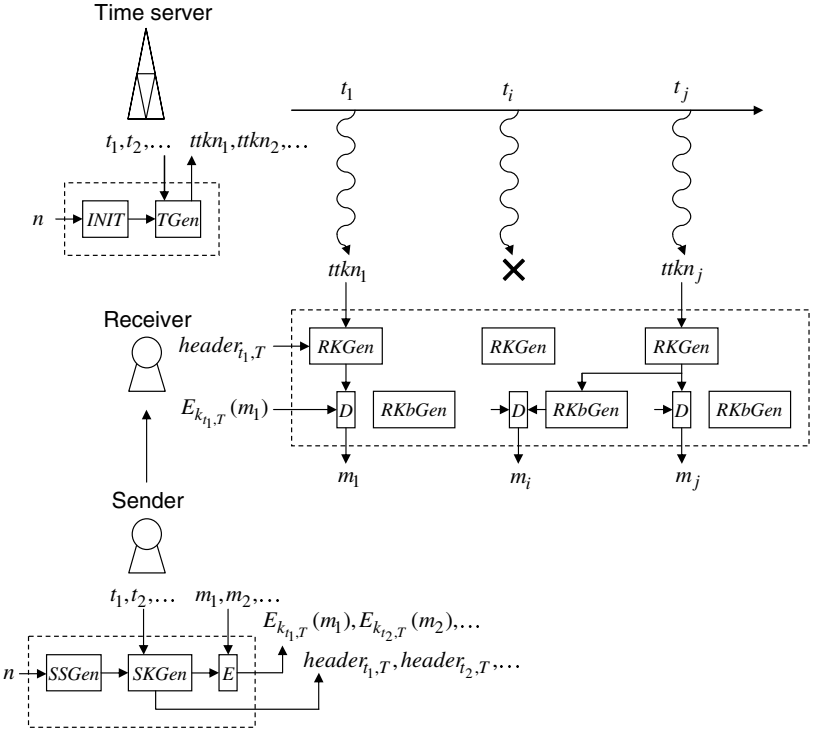


Fig. 1. Usage of the backward timed-release key management scheme

Fig. 1. The time server broadcasts the time token $ttkn_t$ at time t. We assume that Alice sends Bob u messages, m_1, m_2, ..., m_u. The release time of m_i with $1 \leq i \leq u$ is denoted by t_i. Let T be $\{t_1, t_2, \ldots, t_u\}$. Alice generates keys $k_{t_i,T}$ with $1 \leq i \leq u$ for a new private seed σ. Then, Alice encrypts m_i by using $k_{t_i,T}$, and appends the header $header_{t_i,T}$ to the ciphertext. Bob can decrypt m_i at time t_i by computing $k_{t_i,T}$ from $ttkn_{t_i}$ and $header_{t_i,T}$. Even if Bob misses $ttkn_{t_i}$, Bob can compute $k_{t_i,T}$ from $k_{t_j,T}$ with $t_i \prec t_j$, and can decrypt m_i. It is rather obvious that the timed-release encryption scheme without the backward property can be realized by using $INIT$, $TGen$, $SSGen$, $SKGen$, and $RKGen$ with $T = \{t\}$.

Note that Bob does not need to use $header_{t_j,T}$ to compute $k_{t_i,T}$ with $t_i \prec t_j$ from $k_{t_j,T}$. This means that $header_{t_j,T}$ does not need to contain information to compute $k_{t_i,T}$ with $t_i \prec t_j$. This property reduces the size of the header.

A formal definition of the security of the backward timed-release key management scheme is then considered. Intuitively, the security means that even if the receiver has all time tokens earlier than the last time instant t in chain T, he/she cannot compute the key $k_{t,T}$.

Definition 2. For the backward timed-release key management scheme $\mathcal{BTR} = (INIT, TGen, SSGen, SKGen, RKGen, RKbGen)$ and a pair of probabilistic polynomial-time algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, denote the following probability by $\mathrm{Adv}_{\mathcal{A},\mathcal{BTR}}$:

$$\Pr\left[k = k_{t,T} \;\middle|\; \begin{array}{l} \langle pk, sk\rangle \leftarrow INIT(n), \\ \langle T, t, z\rangle \leftarrow \mathcal{A}_1(\mathbb{T}, pk) \text{ where } t \text{ is the last time instant in } T, \\ TKN = \{ttkn_{t'} \leftarrow TGen(sk, pk, t') | t' \in \mathbb{T}, t' \prec t\}, \\ \sigma \leftarrow SSGen(n), \\ HDR = \{header_{t',T} | t' \in T, \\ \qquad \langle k_{t',T}, header_{t',T}\rangle \leftarrow SKGen(pk, \sigma, T, t')\}, \\ k \leftarrow \mathcal{A}_2(TKN, HDR, T, t, z), \\ \langle k_{t,T}, header_{t,T}\rangle \leftarrow SKGen(pk, \sigma, T, t) \end{array} \right].$$

The algorithm $\mathcal{A}_1$ outputs a set of time instants T, a time instant t with $t \in T$, and inner information z which is expected to include $\mathbb{T}$ and pk. If $\mathrm{Adv}_{\mathcal{A},\mathcal{BTR}}$ is negligible for an arbitrary pair of probabilistic polynomial-time algorithms $\mathcal{A}$, then we say that $\mathcal{BTR}$ is secure. ∎

4 Construction Using a Bilinear Map

In this section, we first show the construction of the backward timed-release key management scheme. Then, we clarify the condition with which the construction is secure.

Let $\mathbb{G}_1$, $\mathbb{G}_2$, and e be an additive cyclic group, a multiplicative cyclic group, and a bilinear map defined in Sect. 2.1, respectively. Let P be a generator of $\mathbb{G}_1$. Let h' be a hash function $h' : \mathbb{G}_2 \to \mathbb{Z}_m$. For simplicity, we assume that the set of time instants $\mathbb{T}$ is a subset of $\mathbb{Z}_m^*$.

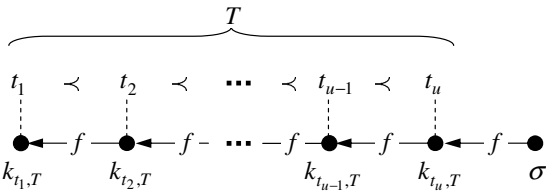


Fig. 2. The chain structure of keys

Keys are elements of $\mathbb{G}_2$ and are related to constitute the chain structure generated by the hash function $f : \mathbb{G}_2 \to \mathbb{G}_2$ defined as $f(x) = g^{h'(x)}$ where $g = e(P, P)^{h'(x)}$. Specifically, as shown in Fig. 2, for $T = \{t_1, t_2, \ldots, t_u\} \subseteq \mathbb{T}$ where $t_i \prec t_{i+1}$ with $1 \leq i < u$, $k_{t_i,T} = f(k_{t_{i+1},T})$. For $t \in T$, let $i_{t,T}$ be the integer such that $t_{|T|+1-i_{t,T}} = t$. That is, t is the $i_{t,T}$-th latest time instant in T.

A backward timed-release key management scheme $\mathcal{BTR}$ is constructed as follows.

- $INIT(n)$:
  1. Choose a secret key $sk = a \in \mathbb{Z}_m^*$ randomly and uniformly.
  2. Compute the corresponding public key as $pk = \langle P, aP \rangle$.
- $TGen(sk, pk, t)$: Compute a time token $ttkn_t$ at time t as

$$ttkn_t = \langle t, \frac{1}{a+t} P \rangle.$$

- $SSGen(n)$: Choose a private seed $\sigma \in \mathbb{G}_2$ randomly and uniformly.
- $SKGen(pk, \sigma, T, t)$:
  1. Compute the key $k_{t,T}$ of release time t in chain T as

$$k_{t,T} = f^{i_{t,T}}(\sigma).$$

  2. Generate the header $header_{t,T}$ as

$$header_{t,T} = \langle t, T, h'(f^{i_{t,T}-1}(\sigma))(aP + tP) \rangle.$$

- $RKGen(pk, ttkn_t, header_{t,T})$: Compute the key $k_{t,T}$ of release time t in chain T from the second component of $ttkn_t$ and the third one of $header_{t,T}$ as

$$\begin{aligned} k_{t,T} &= e\left(\frac{1}{a+t}P, h'(f^{i_{t,T}-1}(\sigma))(aP + tP)\right) \\ &= e(P, P)^{h'(f^{i_{t,T}-1}(\sigma))} \\ &= f(f^{i_{t,T}-1}(\sigma)) = f^{i_{t,T}}(\sigma). \end{aligned}$$

- $RKbGen(pk, T, t, t', k_{t',T})$: If $t' \prec t$, then output $\perp$. Otherwise compute $k_{t,T}$ as

$$\begin{aligned} k_{t,T} &= f^{i_{t,T}-i_{t',T}}(k_{t',T}) \\ &= f^{i_{t,T}-i_{t',T}}(f^{i_{t',T}}(\sigma)) = f^{i_{t,T}}(\sigma). \end{aligned}$$

We show the condition with which the construction is secure. The condition, called the bilinear q-DH-DL assumption, is a combination of the bilinear q-Diffie-Hellman (DH) assumption and the discrete logarithm (DL) assumption. Let P be a generator of $\mathbb{G}_1$. Roughly speaking, the bilinear q-DH assumption states that the following problem is intractable: Given P, aP, $\{\langle t_i, \frac{1}{a+t_i}P\rangle\}_{1\le i<q}$, t_0 as input, output $g^{\frac{1}{a+t_0}}$ where $g = e(P,P)$. The combined DL assumption is the one that the discrete logarithm is $h'(g^{\frac{1}{a+t_0}})$ with respect to each of P and aP. That is, information on the output is added to the input as follows.

Definition 3. The bilinear q-DH-DL problem in $(\mathbb{G}_1, \mathbb{G}_2, e, h')$ where $h' : \mathbb{G}_2 \to \mathbb{Z}_m$ and m is the order of $\mathbb{G}_1$ and $\mathbb{G}_2$ is defined as follows: Given P, aP, $\{\langle t_i, \frac{1}{a+t_i}P\rangle\}_{1\le i<q}$, t_0, $h'(g^{\frac{1}{a+t_0}})P$, and $h'(g^{\frac{1}{a+t_0}})aP$, as input where P is a generator of $\mathbb{G}_1$, $g = e(P,P) \in \mathbb{G}_2$, and a, $t_i \in \mathbb{Z}_m^*$ with $0 \le i < q$, output $g^{\frac{1}{a+t_0}}$. Here, t_i with $0 \le i < q$ are different from each other. For a probabilistic polynomial-time algorithm $\mathcal{A}$, let $\mathrm{Adv}_{\mathcal{A},q}$ denote the following probability.

$$\Pr\left[\mathcal{A}(P, aP, \{(t_i, \tfrac{1}{a+t_i}P)\}_{1\le i<q}, t_0, h'(g^{\frac{1}{a+t_0}})P, h'(g^{\frac{1}{a+t_0}})aP) \to g^{\frac{1}{a+t_0}}\right].$$

If $\mathrm{Adv}_{\mathcal{A},q}$ is negligible for any probabilistic polynomial-time algorithm $\mathcal{A}$, then we say that the bilinear q-DH-DL assumption in $(\mathbb{G}_1, \mathbb{G}_2, e, h')$ holds. ∎

It is obvious that the bilinear q-DH-DL assumption is not weaker than the bilinear q-DH assumption. But we conjecture that the bilinear q-DH-DL assumption is not so strong since the DL assumption in $\mathbb{G}_1$ is not stronger than the q-DH assumption. The q-DH assumption has been first introduced in [14] and been shown to be equivalent to the q-weak DH assumption in [14]. The q-weak DH assumption states that the following problem is intractable: Given P, aP, a^2P, $\ldots$, a^qP as the input, output $a^{-1}P$. The following stronger version of the q-weak DH assumption is often used as the reasonable one [2]: Given P, aP, a^2P, $\ldots$, a^qP, output $\langle t, (a+t)^{-1}P\rangle$.

In the next theorem, we prove that the proposed construction is secure if the bilinear q-DH-DL assumption holds.

Theorem 1. If the bilinear q-DH-DL assumption in $\langle \mathbb{G}_1, \mathbb{G}_2, e, h'\rangle$ with $h' : \mathbb{G}_2 \to \mathbb{Z}_m$ holds, then the proposed construction is secure. ∎

Proof: Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary for the proposed construction. It is shown that if $\mathrm{Adv}_{\mathcal{A},\mathcal{BTR}}$ is not negligible, then we can construct an adversary $\mathcal{B}$ for the bilinear q-DH-DL problem such that $\mathrm{Adv}_{\mathcal{B},q}$ is non-negligible. The algorithm $\mathcal{B}$, which is provided with

$$\langle P, aP, \{\langle t_i, \frac{1}{a+t_i}P\rangle\}_{1\le i<q}, t_0, h'(g^{\frac{1}{a+t_0}})P, h'(g^{\frac{1}{a+t_0}})aP\rangle,$$

gives parameters to the algorithm $\mathcal{A}$. The simulation is as follows:

1. Set $\mathbb{T} = \{t_i | 0 \le i < q\}$ so that $t_i \prec t_0$ with $1 \le i < q$.
2. Set $pk = \langle P, aP\rangle$. This means that $sk = a$. Let $f(x) = e(P,P)^{h'(x)}$.

3. Let $\langle T, t, z\rangle \leftarrow \mathcal{A}_1(\mathbb{T}, pk)$. If $t \neq t_0$, then halt. In this case, the algorithm fails. If $t = t_0$, then proceed to the next step.
4. Set $TKN = \{\langle t_i, \frac{1}{a+t_i}P\rangle\}_{1\leq i<q}$.
5. Set $HDR = \{header_{t',T} | t' \in T\}$ where $header_{t',T}$ is generated as follows.
   - For $t' = t_0$, $header_{t',T} = \langle t_0, T, P\rangle$. This means that σ is set such that $f(\sigma) = g^{\frac{1}{a+t_0}}$ since $k_{t_0,T} = e(\frac{1}{a+t_0}P, P) = g^{\frac{1}{a+t_0}}$.
   - For $t' \neq t_0$, $header_{t',T} = \langle t', T, h'(f^{i_{t',T}-1}(\sigma))(aP + t'P)\rangle$.
     If $i_{t',T} = 2$, then the third component of $header_{t',T}$ is computed as

$$h'(f(\sigma))(aP + t'P) = h'(g^{\frac{1}{a+t_0}})aP + t'h'(g^{\frac{1}{a+t_0}})P.$$

     Otherwise, since $i_{t',T} > 2$, $h'(f^{i_{t',T}-1}(\sigma))$ is computed as

$$h'(f^{i_{t',T}-1}(\sigma)) = h'(f^{i_{t',T}-3}(e(P, h'(g^{\frac{1}{a+t_0}})P))).$$

     From the bilinearity and the definition of f, it holds that

$$\begin{aligned} h'(f^{i_{t',T}-3}(e(P, h'(g^{\frac{1}{a+t_0}})P))) &= h'(f^{i_{t',T}-3}(e(P,P)^{h'(g^{\frac{1}{a+t_0}})})) \\ &= h'(f^{i_{t',T}-2}(g^{\frac{1}{a+t_0}})) \\ &= h'(f^{i_{t',T}-1}(\sigma)). \end{aligned}$$

6. Let $k = \mathcal{A}_2(TKN, HDR, T, t, z)$. From the definition, if $\mathcal{A}_2$ outputs the correct value, then $k = k_{t,T} = k_{t_0,T} = g^{\frac{1}{a+t_0}}$.

Note that if $\mathcal{A}_1$ did not choose the time instant $t = t_0$ in step 3, then $\mathcal{B}$ has few chance to find $g^{\frac{1}{a+t_0}}$. Therefore, $\mathcal{B}$ has chance to compute $g^{\frac{1}{a+t_0}}$ only if $\mathcal{A}_1$ chooses $t = t_0$, this happens with probability $\frac{2^{q-2}}{2^{q-1}-1} > \frac{1}{2}$. Therefore we have $\mathrm{Adv}_{\mathcal{B},q} > \frac{1}{2}\mathrm{Adv}_{\mathcal{A},\mathcal{BTR}}$. Thus, $\mathrm{Adv}_{\mathcal{B},q}$ is negligible if and only if $\mathrm{Adv}_{\mathcal{A},\mathcal{BTR}}$ is negligible. This completes the proof. ∎

We also present a simpler but stronger assumption on which the proposed construction is secure. The assumption, called the bilinear q-DH-OW assumption, is a combination of the bilinear q-DH assumption in $(\mathbb{G}_1, \mathbb{G}_2, e)$ and the one-wayness of h'. Roughly speaking, the bilinear q-DH-OW assumption states that the following problem is intractable: Given P, aP, $\{\langle t_i, \frac{1}{a+t_i}P\rangle\}_{1\leq i<q}$, t_0, and $h'(g^{\frac{1}{a+t_0}})$ as input, output $g^{\frac{1}{a+t_0}}$ where $g = e(P, P)$. The details are omitted because of space limitation.

Lastly, we briefly compare the efficiency of the proposed construction with that of the previous constructions [3, 10, 15, 16, 19] (see Sect. 2), which do not have the backward property, in the point of the size of a time token, a key, and a header. The essential element of a time token, a key, and a header is a single element of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_1$, respectively, in any construction. That is, the backward property is realized without increasing the memory complexity. The complexity of computing keys of earlier time instants is large because f is based on exponentiation. However, compared with the construction shown in Sect. 2,

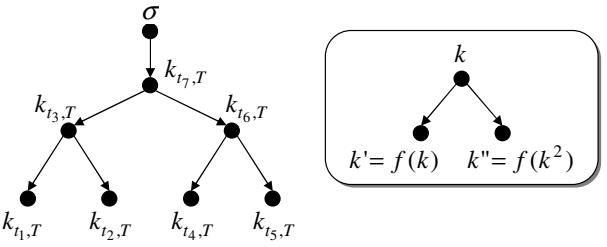


Fig. 3. The tree structure of keys for $T = \{t_1, t_2, \ldots, t_7\}$

the proposed construction is more efficient because the receiver only needs to apply the exponentiation function f defined by the single base $g = e(P, P)$ while in the construction shown in Sect. 2, the receiver needs to apply exponentiation functions defined by different bases $e(aP, h(t_i))$.

5 Conclusion

The backward timed-release key management scheme has been proposed in this paper. The backward property of keys improves resilience to missing time tokens. The construction using a bilinear map has been proposed, and the condition under which the construction is secure has been clarified. The backward property is realized by relating keys to constitute the one-way chain structure.

Keys can be related in various ways such as a tree structure. The straightforward tree structure is shown in Fig. 3 where the key of the latest time instant, which is associated with the root, can be used for computing any key of an earlier time instant and the number of iterations of f is at most twice. That is, the computational complexity of recovering a key of an earlier time instant can be reduced. But the perfect backward property is not provided since keys associated with internal nodes are used only for computing their descendents. A possible direction to the future study is to provide preferable properties by relating keys in various ways.

References

1. M. Bellare and S. Goldwasser: "Encapsulated Key-Escrow," MIT LCS Tech. Report MIT/LCS/TR-688, 1996.
2. D. Boneh and X. Boyen: "Short Signatures without Random Oracles," EUROCRYPT '04, LNCS 3027, pp.56–73, 2004.
3. I.F. Blake and A.C-F. Chan: "Scalable, Server-Passive, User Anonymous Timed Release Public Key Encryption from Bilinear Pairing," http://eprint.iacr.org/2004/211/, 2004.
4. D. Boneh, B. Lynn, and H. Shacham: "Short Signatures from the Weil Pairing," ASIACRYPT '01, LNCS 2248, pp.514–532, 2001.

5. D. Boneh and M. Franklin: "Identity-Based Encryption from the Weil Pairing," CRYPTO '01, LNCS 2139, pp.213–229, 2001.
6. D. Boneh and M. Naor: "Timed Commitments (Extended Abstract)," CRYPTO 2000, LNCS 1880, pp.236–254, 2000.
7. S. Goldwasser and M. Bellare: Lecture Notes on Cryptography, http://www.cs.ucsd.edu/users/mihir/papers/gb.pdf, 2001.
8. J. Garay and M. Jakobsson: "Timed Release of Standard Digital Signatures," FC '02, LNCS 2357, pp.168–182, 2002.
9. J. Garay and C. Pomerance: "Timed Fair Exchange of Standard Signatures," FC '03, LNCS 2742, pp.190–207, 2003.
10. Y.H. Hwang, D.H. Yum, and P.J. Lee: "Timed-Release Encryption with Pre-open Capability and Its Application to Certified E-mail System," ISC '05, LNCS 3650, pp.344–358, 2005.
11. W. Mao: "Timed-Release Cryptography," SAC '01, LNCS 2259, pp.342–357, 2001.
12. T. May: "Timed-Release Crypto," http://www.hks.net.cpunks/cpunks-0/1560.html, 1992.
13. M.C. Mont, K. Harrison, and M. Sadler: "The HP Time Vault Service: Innovating the Way Confidential Information is Disclosed at the Right Time," HP Lab. Report HPL-2002-243, 2002.
14. S. Mitsunari, R. Sakai and M. Kasahara: "A New Traitor Tracing," IEICE Trans. Fundamentals, vol.E85-A, no.2, pp.481-484, 2002.
15. D. Nali, C. Adams, and A. Miri: "Time-Based Release of Confidential Information in Hierachical Settings," ISC '05, LNCS 3650, pp.29–43, 2005.
16. I. Osipkov, Y. Kim, and J.H. Cheon: "New Approaches to Timed-Release Cryptography," http://eprint.iacr.org/2004/231/, 2004.
17. T.P. Pederson: "A Threshold Cryptosystem without a Trusted Party," EUROCRYPT'91, LNCS 547, pp.522–526, 1991.
18. R.L. Rivest, A. Shamir, and D. A. Wagner: "Time-Lock Puzzles and Timed-Release Crypto," MIT LCS Tech. Report MIT/LCS/TR-684, 1996.
19. M. Yoshida and S. Mitsunari: "A Time-Capsule Encryption," the IPAX Autumn 2004 (oral presentation), http://homepage1.nifty.com/herumi/mtt/timecapsule.20040909.ppt (in Japanese), 2004.

Property-Based Broadcast Encryption for Multi-level Security Policies*

André Adelsbach, Ulrich Huber, and Ahmad-Reza Sadeghi

Horst Görtz Institute for IT Security,
Ruhr-Universität Bochum
andre.adelsbach@nds.rub.de, {huber, sadeghi}@crypto.rub.de

Abstract. Most current Digital Rights Management (DRM) systems rely on the doubtful assumption that all devices are equally trustworthy. This allows a pirate to obtain access if he undetectedly breaks just one arbitrary device. As there is a multitude of different devices, trust assumptions and policies should depend on the security level of each device type. For each content item to be distributed, the content providers should be able to base their access decision on various properties that the devices might have, such as the devices' tamper resistance, geographical region, output interface or device model.

We propose a hierarchical property-based broadcast encryption scheme enabling a variety of new business models. It operates with an arbitrary number of properties, including one hierarchical property such as tamper resistance. The scheme is secure and more efficient than existing Broadcast Encryption (BE) schemes in the hierarchical setting. As the first building block, we formalize the notion of properties with respect to BE and show an approach for representing them in a binary tree structure. As the second building block, we use existing BE schemes to achieve short ciphertexts. Specifically, we enhance the Complete Subtree scheme with pseudo-random chains to embed the hierarchical property.

1 Introduction

The development of software and hardware architectures for Digital Rights Management (DRM) systems has enjoyed increasing interest in recent years. A DRM system allows content providers to distribute digital content to users in such a way that the associated usage rights are enforced. Current DRM systems perform badly when a pirate, i.e., a malicious user, breaks a single device and remains undetected. Consider the example of CD and DVD protection; as each compliant device, e.g., DVD player or PC, obtains the same privileges, it is sufficient for the pirate to steal the keys of one arbitrary device. Subsequently, he can use these keys to illegally obtain access, e.g., to create copies of identical quality. If he distributes them on a small scale, say to his family and friends, countermeasures such as watermarking [2, 3, 4], traitor tracing [5, 6] and revocation [7, 8] will fail because the incident will remain undetected. This weakness is due to the

* This paper is an extended abstract of a technical report [1].

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 15–31, 2006.
© Springer-Verlag Berlin Heidelberg 2006

doubtful assumption that all devices are equally trustworthy and comply with the usage rights by construction.

In practice however, there is a multitude of different devices. In the CD example, the device types include hi-fi CD players, portable players, PCs and car radios. Similar examples hold for content on DVDs or MP3 files. Depending on the use case and business model, different devices may have different requirements. For example, the cost pressure on portable players is usually higher than on hi-fi players, making it more difficult for a manufacturer to build secure devices while maintaining competitive production costs. Therefore, it is highly unlikely that all devices show the same resistance to attacks. Tamper resistance always comes in different flavors [9, 10], and therefore trust assumptions should be different for each type of device.

To avoid unconditional access through the weakest device, Popescu et al. propose multi-level security policies [11]. With such policies, the content providers can base their access decision on the tamper resistance of devices. Consider an example with weak, medium and strong tamper resistance; if the content provider grants access to medium and strong devices, then the pirate cannot obtain access by breaking a weak device. Tamper resistance is an example of a *hierarchical property* that partitions the devices into hierarchical classes.

In addition to the hierarchical property, there may be other properties that influence the content providers' decision to grant access. One popular example from current DVD encryption is the *geographical region* (e.g., country) in which the device was sold. A DVD player may be intended for a specific region, such that access to content of other regions is impossible. Some additional examples in this context are the device's *output interface* (e.g., digital vs. analogue or HDCP-protected vs. DTCP-protected[1]), *manufacturer name* (a content provider's trust model may depend on the manufacturer), *manufacture date* (old devices may have weaker security mechanisms), the *device model* (expensive models may be more tamper-resistant) and the *data format* that a device understands (e.g., video vs. audio data).

Our Contribution We propose a hierarchical Property-Based Broadcast Encryption (PBBE) scheme that allows content providers to base their access decision on the devices' properties, including one hierarchical property such as their tamper resistance. In the hierarchical setting, i.e., when devices can be partitioned into hierarchical classes, our PBBE scheme is more efficient than existing Broadcast Encryption (BE) schemes. Specifically, we achieve a header length reduction by a factor that is logarithmic in the number l of hierarchical classes, e.g., $\lceil \frac{1}{2} \cdot \lceil \log_2(l) \rceil \rceil$ compared to the Subset Difference (SD) scheme and Inclusion Exclusion (IE) trees of [14]. In addition, we show that our PBBE scheme is IND-CCA1-secure in the sense of [8].

As a first building block, we formalize the notion of properties with respect to BE and show a detailed approach for representing them in a binary tree. Each leaf of this tree represents one possible property combination of a device. The content providers can formulate their access decision as a list of arbitrary Boolean

[1] The HDCP scheme [12] is considered to be broken whereas DTCP [13] is not.

expressions over the properties. Although properties have been mentioned in related work (see Section 2), the notion of properties hasn't been formalized in this context. Existing schemes lack either flexibility in the handling of *several* properties or efficiency in the context of a hierarchical property. We argue that formalization of properties is necessary for any implementation, e.g., to allow a security proof.

As the second building block, we use existing BE schemes to achieve efficiency. If we assume that there is no hierarchical property, then the proposed scheme is simply the Complete Subtree (CS) scheme of [8] enriched with guidelines on the optimum assignment of property combinations to the leaves of the CS key tree. Given a set of devices, their properties and possibly some a-priori information on their decision relevance,[2] we order the leaves of the CS tree such that the length of an average message header is minimized. Although related work mentions that this is possible, we are unaware of any specific proposal to achieve this. Now when we add a hierarchical property, we achieve a logarithmic header length reduction by embedding a computational key assignment using pseudo-random chains into the CS key tree.

2 Related Work

Berkovits first used the notion of broadcasting a secret [15]. Fiat and Naor were the first to formally define the functional requirements of a BE scheme [7]. Naor, Naor and Lotspiech introduced a general class of BE schemes called *Subset Cover BE* (SCBE) schemes which cover the set of non-revoked devices with a collection of well-defined subsets [8]. Specifically, they proposed two SCBE schemes based on binary key trees: *Complete Subtree* (CS) and *Subset Difference* (SD). Although they mentioned properties, they didn't give a formal definition or an analysis of their effect on efficiency. With the *Layered SD* (LSD) scheme, Halevy and Shamir reduced the storage size of devices at the price of doubling the header length [14]. In the same paper, they introduced Inclusion Exclusion (IE) trees, but didn't formalize the concept of properties. Using + and − signs as predicates for the tree's nodes, IE trees describe logical operations over properties. Dodis and Fazio extended CS, SD and LSD to the public key setting using (hierarchical) identity-based encryption [16]. A recent SCBE scheme based on one-way chains is due to Jho et al. [17]. We give further references in the technical report [1].

3 Need for Property-Based BE

3.1 Underlying Scenario

We refer to a scenario where all devices are stateless, i.e., they can't update their key material after the setup phase, and have a fixed number of properties.

[2] Decision relevance is the relative importance of properties to the content providers. Some properties will influence their decisions more often than others.

Before formally defining properties, we give an informal example. Let there be three properties: the data format that a device understands (α or β), the region in which it is sold (A, B or C) and its tamper resistance (between 0 and 3). In this example, all devices which understand data format α, belong to region B and have tamper resistance level 2, share the same combination of property values or *(property) configuration*.

There may be a hierarchical relation between the devices which we model as a hierarchical property. To give an example, let the tamper resistance levels above be ordered in a meaningful way: 0 represents "minimum tamper resistance", and the values increase from 0 to 3 which represents "maximum resistance". A device with tamper resistance 2 automatically inherits the permission to receive content which was addressed to devices with tamper resistance < 2.

We assume the number of such configurations to be significantly smaller than the number of devices. In the above example, there are 24 configurations $(\alpha, A, 0)$, ..., $(\beta, C, 3)$, but possibly millions of devices. The reason for our assumption is two-fold: First, if there are more configurations than devices, then some properties are redundant and can be removed or merged. Second, if there are still as many configurations as devices, i.e., there is only 1 device per configuration, then it is impossible to achieve any efficiency gains by considering properties; configurations and devices are two names for the same thing.

3.2 Available Methods

CS, SD, LSD and the one-way chain based schemes of [17] are SCBE schemes for stateless devices. To grant access, they *cover* the set of non-revoked devices with the available *subsets*, hence their common name. To save bandwidth, the center of any SCBE scheme encrypts the long broadcast message with a short session key and subsequently encrypts the session key for every subset in the cover. The broadcast message thus has the following elements: (i) the indexing information, indicating which subsets are covered, (ii) the encryptions of the session key with the subsets' keys, and (iii) the encryption of the message with the session key. (i) and (ii) are called the *(message) header*. In our proposed solution in Section 5, we refer to any such scheme by using a generalized key generation, key extraction, encryption and decryption algorithm of an SCBE scheme: KeyGenSCBE(), KeyExtrSCBE(), EncSCBE() and DecSCBE().

3.3 Difficulties

The CS, SD and LSD schemes assume that all devices are equal at manufacture time and assign the leaves of their key trees to devices in arbitrary order. If we want to enhance the key assignment by considering device properties, there are three straightforward possibilities. As an example, we look at the first 3 configurations of the above example; let there be 6 devices with configuration $(\alpha, A, 0)$, 7 with $(\alpha, A, 1)$ and 3 with $(\alpha, A, 2)$.

The first possibility for considering properties is to assign devices to leaves in the order of the property configurations. In Part 1 of Figure 1, we assign the

first 6 leaves to the devices of configuration $(\alpha, A, 0)$, the next 7 to $(\alpha, A, 1)$, the next 3 to $(\alpha, A, 2)$, and so forth.[3] The disadvantage of this approach is obvious: The devices of a particular configuration do not group below a common node in the tree. Therefore, the cover algorithm of the aforementioned SCBE schemes needs *several* subsets to cover them. For example, the devices of configuration $(\alpha, A, 1)$ have no common ancestor to exclusively cover them.

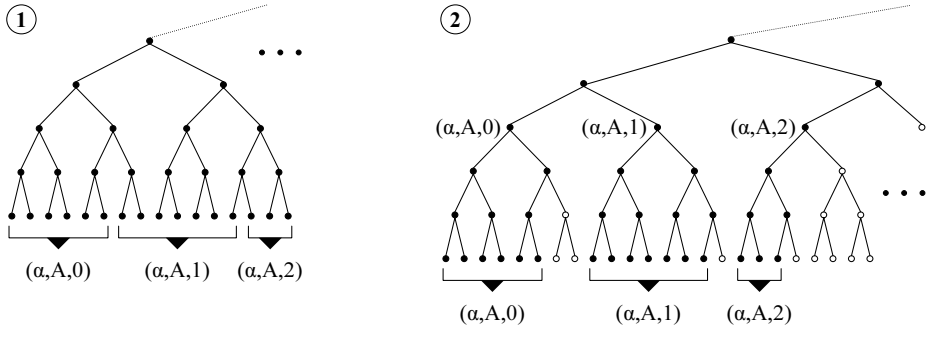


Fig. 1. Disadvantages of conventional trees related to property configurations

The second possibility is to define a tree level in which each node corresponds to a configuration. For the 24 configurations of our example, we need the fifth level with $2^5 = 32$ nodes as the tree is binary. Part 2 of Figure 1 shows the first three such nodes. The corresponding devices are grouped below these nodes, i.e., the 6 devices of $(\alpha, A, 0)$ below the first node, the 7 devices of $(\alpha, A, 1)$ below the second and so forth. All leaves have the same distance to the root. The disadvantage of this approach is that many leaves remain unassigned. For example, the node of configuration $(\alpha, A, 2)$ has 5 unassigned leaves.

The third possibility is to allow varying height of the subtrees rooted at configuration nodes. This modification is similar to Part 2 of Figure 1. The only difference is that below each configuration node, the subtree has the minimum number of levels allowing to assign a leaf to each property value. For example, in contrast to the 3 additional tree levels below $(\alpha, A, 0)$ and $(\alpha, A, 1)$, we only need 2 additional levels below $(\alpha, A, 2)$. This gives us 4 leaves, allowing to assign each of the 3 property values its own leaf. This approach is already close to our proposed solution. However, it still has a longer message header if there is a hierarchical property as we will show in Section 5.7. Our proposed solution extends the third possibility. However, we cut the resulting tree below the configuration nodes and obtain separate subtrees for each configuration. Figure 2 gives an impression of the approach, while we give further details in Section 5.

[3] We only show the left-most part of the tree which will continue to the right with the remaining 21 configurations.

4 Preliminaries

4.1 Notations

We recall some standard notations that will be used throughout the paper. First, we denote objects with lower-case variables, e.g., o_1, and roles with upper-case variables, e.g., X_1. When we summarize objects in set notation, we use an upper-case calligraphic variable, e.g., $\mathcal{O} = \{o_1, o_2, \ldots\}$. Second, let $\mathsf{A}()$ be an algorithm. By $y \leftarrow \mathsf{A}(x)$ we denote that y was obtained by running $\mathsf{A}()$ on input x. If $\mathsf{A}()$ is deterministic, then y is a variable with a unique value; conversely, if $\mathsf{A}()$ is probabilistic, then y is a random variable. Finally, $\wedge$ and $\vee$ denote the logical `and`- and `or`-operator, respectively.

4.2 Roles and Objects

After having introduced several terms informally, we now give their precise meaning. A *device* d is capable of obtaining and processing digital content. $\mathcal{D}$ is the set of all devices $\mathcal{D} := \{d_{ind} | d_{ind} \text{ is a device}\}$, where the device index $ind := (k, i_k)$ is an unambiguous identifier consisting of the index k of configuration c_k and the running index i_k of devices in c_k. A *content provider* A_s distributes content items, referred to as *messages* m, via the broadcast center. A_s selects an arbitrary set of privileged configurations that have access to m. The fully trusted *(broadcast) center* manages the broadcast encryption infrastructure. The center's two main tasks are key management and encrypted broadcasting of messages. A *device manufacturer* produces and distributes devices that contain key material assigned by the center.

4.3 Properties

Definition 1. *A* ***property*** p_j *is an unambiguous quality of a device which we represent as a function* $p_j : \mathcal{D} \to \mathcal{P}_j$*, where* $\mathcal{P}_j$ *is a finite set* $\mathcal{P}_j := \{0, 1, \ldots, l_j - 1\} \subset \mathbb{N}$ *of* ***property values****.*[4] *The number of properties is denoted with* g*, i.e.,* $1 \leq j \leq g$*.*

Definition 2. *A* ***(property) configuration*** c_k *is a* g*-tuple* $c_k := (x_1, \ldots, x_g)$ *of property values corresponding to the* g *properties* $(p_1, \ldots, p_g)$ *of a device. The set of all configurations is* $\mathcal{C} := \{c_1, c_2, \ldots\}$*.*

Definition 3. *Let* j *be the index of the hierarchical property and* $c_k = (x_1, \ldots, x_j, \ldots, x_g)$ *be a configuration. Configuration* $c_k^{(1)}$ *is* ***superior*** *to* $c_k^{(2)}$ *if it has (i) a strictly greater value* $x_j^{(1)} > x_j^{(2)}$ *of the hierarchical property and (ii) identical values in all other properties:* $\forall\, j' \neq j :\ x_{j'}^{(1)} = x_{j'}^{(2)}$ *. Conversely,* $c_k^{(2)}$ *is* ***inferior*** *to* $c_k^{(1)}$*. Devices with configuration* c_k *inherit access to all messages to which* c_k*'s inferior configurations have access.*

Definition 4. *A configuration is* ***non-intact*** *if it either (i) contains at least one revoked device or (ii) is inferior to a non-intact configuration.*

[4] The choice of integers as elements of $\mathcal{P}_j$ is simply a design choice which facilitates the notation. We might as well use sets of the form $\{\alpha, \beta, \ldots\}$ as introduced before.

4.4 Pseudo-random Chains

Definition 5. [8, Section 3.2] *We say that* $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ *is a* ***pseudo-random chain generator*** *(PRCG) if no probabilistic* λ*-polynomial-time adversary can distinguish the output of* $\mathsf{G}()$ *on a randomly chosen seed from a truly random string of identical length.*

Definition 6. *A* ***pseudo-random chain*** *(PRC) is a sequence of bit strings* $(str_0, str_1, str_2, \ldots)$ *that are derived from the random seed* str_0 *by iteratively applying a PRCG:* $str_{i+1} \leftarrow \mathsf{G}(str_i)$

5 Proposed Solution

5.1 Overview

We propose a Property-Based Broadcast Encryption (PBBE) scheme for an arbitrary number of properties, including one hierarchical property. A PBBE scheme should allow content providers to grant access based on properties, i.e., the criteria in their business models, *and* revocation information for individual devices. Our scheme uses existing SCBE schemes on two levels. The upper level applies the CS scheme enhanced with pseudo-random chains; the lower level independently applies an arbitrary SCBE scheme.

The upper level represents the properties of a device: Each leaf of the upper tree stands for a single configuration c_k, thus the name *(property) configuration*

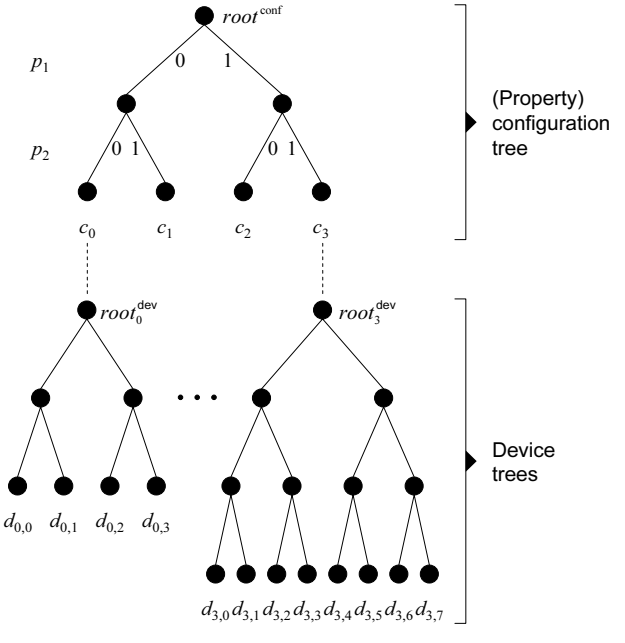


Fig. 2. Two levels of SCBE schemes

tree. The lower level represents all devices that have the same configuration. To illustrate the approach, we show an example with tree-based schemes on the lower level (see Figure 2); we will call them *device trees*. Each leaf of the forrest of device trees stands for one individual device d_{k,i_k}. In Figure 2, devices $d_{0,0} - d_{0,3}$ possess configuration c_0, while devices $d_{3,0} - d_{3,7}$ have configuration c_3.

In the setup phase, each device obtains all keys that are related to its configuration in the configuration tree. In addition, each device obtains key material in its device tree. In the example of Figure 2, device $d_{3,5}$ obtains all keys related to configuration c_3 in the configuration tree and all keys related to leaf $d_{3,5}$ in the device tree. When a device is compromised, i.e., its key material has been misused for purposes such as key sharing, the center can remove its privileges by performing revocation. When a device d_{k,i_k} is revoked, the center no longer uses any key known to this device. Therefore, it cannot use this device's keys in the configuration tree. However, the device tree of c_k still allows to broadcast to all non-revoked devices of c_k.

5.2 Tree Arithmetic

Encoding of Property Values. In the above example, we have seen three properties p_1, p_2 and p_3 representing data format, region and tamper resistance. They have $l_1 = 2$, $l_2 = 3$ and $l_3 = 4$ property values each: $\mathcal{P}_1 = \{0,1\} := \{\alpha, \beta\}$, $\mathcal{P}_2 = \{0,1,2\} := \{A, B, C\}$ and $\mathcal{P}_3 := \{0,1,2,3\}$. These property values are encoded into the configuration tree as follows.[5] First, we calculate the number L_j of tree levels that we need for representing a property with l_j property values in a binary tree: $L_j := \lceil \log_2(l_j) \rceil$. For example, we need $L_2 = \lceil \log_2(3) \rceil = 2$ tree levels to encode the 3 property values of p_2.

Second, we calculate how many nodes we obtain at the lowest level of this property. Ideally, there would be l_j nodes, i.e., just the number of property values. However, as the tree is binary, we have $l_j^{\mathsf{real}} = 2^{L_j}$ nodes. In the example of p_2, we have $l_2^{\mathsf{real}} = 2^2 = 4$ nodes although we use only $l_2 = 3$ of them. Note that $l_j^{\mathsf{real}} = l_j$ iff l_j is a power of 2, and $l_j^{\mathsf{real}} > l_j$ otherwise.

We assign the values of a property to the nodes on the lowest level of this property, going from left to right. The left-most node obtains the first value and so on until we reach the l_j^{real}-th node and start again. We repeat until each node corresponds to a property value. A configuration $c_k = (x_1, \ldots, x_g)$ therefore easily translates into a path from the root to the leaf representing c_k. We simply use the binary representation of the property values and descend as follows: When a 0 occurs, we descend to the left child, while a 1 implies the right child. For example, the configuration $(\beta, B, 2)$ corresponds to $c = (1,1,2)_{10} = (1,01,10)_2$.

Often-used Parameters. In order to simplify the notation, we introduce some often-used parameters of the configuration tree. All variables are a direct function of the number of property values $l_1, \ldots, l_g$. The calculations simply rely on the fact that the tree is binary. First, we calculate the number n^{leaves} of leaves of the configuration tree. For g properties, we find $n^{\mathsf{leaves}} = \prod_{j=1}^{g} l_j^{\mathsf{real}}$.

[5] For a detailed example, we refer to the technical report [1].

On several occasions, it will be useful to decompose n^{leaves} into two factors $n^{\mathsf{leaves}} = n_{p_g}^{\mathsf{nodes}} \cdot n_{p_g}^{\mathsf{leaves}}$. The first is the number $n_{p_g}^{\mathsf{nodes}}$ of nodes at the uppermost level of property p_g. This number is calculated as $n_{p_g}^{\mathsf{nodes}} = \prod_{j=1}^{g-1} l_j^{\mathsf{real}}$. For the second factor, note that there is a subtree rooted in each of these nodes. Each of the subtrees has a number $n_{p_g}^{\mathsf{leaves}}$ of leaves with $n_{p_g}^{\mathsf{leaves}} = l_g^{\mathsf{real}} = 2^{L_g}$.

Indexing of Configurations. The indexing method for assigning the index k to configuration c_k is trivial: We start with index $k = 0$ for the configuration of the left-most leaf in the tree. Then going from the left-most to the right-most leaf or configuration, we increase k by 1 for each configuration and obtain n^{leaves} configurations. Therefore, k is an unambiguous index for $c_k = (x_1, \ldots, x_g)$, and we use them interchangeably.

5.3 Key Generation in the Tree Setup Phase

This section details our proposal for instantiating the key generation algorithm. It will consist of three subroutines. We start with the first two subroutines, which are related to the property configuration tree. The third subroutine, related to the device trees, follows at the end of Section 5.3.

Property Configuration Tree. The (property) configuration tree is the upper tree in Figure 2 and allows the center to directly broadcast to intact configurations. Before we can generate this tree, the parties need to agree on the properties on which the content providers base their access decisions. As we do not want to restrict the flexibility of the business model, we leave the details of this agreement to the implementation.

The setup of the configuration tree consists of two subroutines. First, we generate a non-hierarchical configuration tree which we instantiate with a standard CS tree. Second, we implant the hierarchical property into this tree by adding pseudo-random chains. Note that for efficiency reasons, the center orders the g properties according to their decision relevance to the content providers. The most decision-relevant property appears on the uppermost levels of the configuration tree while the least decision-relevant property appears at the bottom. This ensures that the important properties correspond to high-level nodes that cover many leaves, thus reducing the header length.

1. $(\mathcal{MK}^{\mathsf{conf}}, \mathcal{SK}'^{\mathsf{conf}}) \leftarrow \mathsf{ConfTreeGen}(P, 1^\lambda)$: This subroutine generates a non-hierarchical configuration tree. The tree's leaves represent configurations instead of devices. The input parameters are the ordered properties $P := (p_1, p_2, \ldots, p_g)$ and a security parameter 1^λ. The output is the master key $\mathcal{MK}^{\mathsf{conf}}$ of the configuration tree and the set $\mathcal{SK}'^{\mathsf{conf}}$ of all keys in the tree. The center derives the number n^{leaves} of configurations from the properties P as shown in Section 5.2. With n^{leaves} as the tree's number of leaves, the non-hierarchical configuration tree is instantiated with a standard CS tree: $(\mathcal{MK}^{\mathsf{conf}}, \mathcal{SK}'^{\mathsf{conf}}) \leftarrow \mathsf{KeyGenCS}(n^{\mathsf{leaves}}, 1^\lambda)$.

2. $\mathcal{SK}^{\mathsf{conf}} \leftarrow \mathsf{HierarImplant}(\mathcal{SK}'^{\mathsf{conf}}, P, 1^\lambda)$: This subroutine implants the hierarchical property into the configuration tree by adding pseudo-random chains. The input parameters are the keys $\mathcal{SK}'^{\mathsf{conf}}$ from Step 1, the ordered properties P and

the security parameter 1^λ. The output is the tree's set of keys, this time with pseudo-random chains. To simplify the notation, we always show the hierarchical property on the lowest layer of the configuration tree and assign it the index g. In the technical report [1], we release this restriction and allow the hierarchical property to be in any tree layer.

The subroutine HierarImplant() alters the CS tree as follows. In a regular CS tree, the center generates (pseudo-)independent keys. We modify this pattern in exactly one level of the tree: In the lowest level of the hierarchical property, we replace the CS keys by deriving them from right to left in the order of the configurations' superiority relation. To do so, we associate one pseudo-random chain (see Definition 6) with all configurations that are inferior to each other. The first string corresponds to the leaf of the most superior configuration, the last string to that of the most inferior configuration.

Note that in Part (ii) of Definition 3, we require all non-hierarchical property values to be identical. For two configurations to be superior/inferior, all property values above the hierarchical property p_g must match. As there are $n_{p_g}^{\mathsf{nodes}}$ possible combinations above p_g, we obtain $n_{p_g}^{\mathsf{nodes}}$ pseudo-random chains. Figure 3 shows a configuration tree with $n_{p_3}^{\mathsf{nodes}} = 8$ implanted pseudo-random chains, indicated with arrows ($\leftarrow$). As an example for superiority, take $c = (x_1, x_2, x_3) = (2, 1, 2)$.[6] It can follow its chain and derive the keys of the inferior configurations $(2, 1, 1)$ and $(2, 1, 0)$ by evaluating the PRCG.

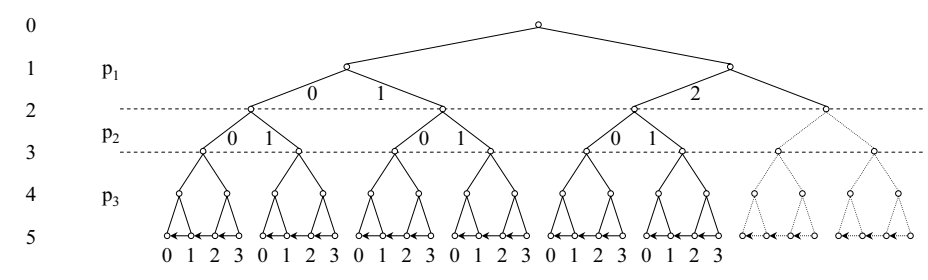


Fig. 3. A configuration tree: The example shows three properties (p_1, p_2, p_3) where p_3 is hierarchical and $l_1 = 3$, $l_2 = 2$ and $l_3 = 4$

We call the strings of the chain *labels* from which the actual keys are derived. If c^{inf} is inferior to c^{sup} with a distance of δ,[7] then c^{inf}'s key is the result of two independent PRCGs applied to c^{sup}'s label:

$$sk^{\mathsf{inf}} \leftarrow \mathsf{G_K}(\mathsf{G_L^\delta}(label^{\mathsf{sup}})) \tag{1}$$

Like the SD scheme [8], we need more than one PRCG to ensure key indistinguishability: We use $\mathsf{G_L}()$ to derive labels and $\mathsf{G_K}()$ to derive the actual key from

[6] To find the path from root to leaf, note that $(2, 1, 2)_{10} = (10, 1, 10)_2$.

[7] I.e., their values of the hierarchical property differ by δ.

a label. The output $\mathcal{SK}^{\mathsf{conf}}$ of $\mathsf{HierarImplant}()$ contains the labels of the pseudo-random chains on the lowest tree level. We denote with $\mathcal{SK}^{*\mathsf{conf}}$ the actual key set where labels are replaced with keys using $\mathsf{G_K}()$.

Device Trees. As soon as an individual device is revoked, its configuration becomes non-intact (see Definition 4). We cannot individually revoke this device in the configuration tree because all other devices in its configuration obtained the same keys. We therefore revoke the whole configuration[8] in the configuration tree and thereby also exclude all of its non-revoked devices. We thus need to use the device trees to re-include the non-revoked devices of non-intact configurations. For each device tree, a secure[9] SCBE scheme can be chosen arbitrarily and independently. If the center chooses a tree-based scheme from [8, 14], the individual devices correspond to the tree's leaves. The center generates the device trees as follows:

3. $(\mathcal{MK}^{\mathsf{dev}}, \mathcal{SK}^{\mathsf{dev}}) \leftarrow \mathsf{DevTreeGen}(\mathcal{N}_{\mathcal{C}}, 1^{\lambda})$ with $\mathcal{N}_{\mathcal{C}} := \{n_{c_0}, n_{c_1}, \ldots\}$: The input parameters are the set $\mathcal{N}_{\mathcal{C}}$ containing the numbers n_{c_k} of devices in configuration c_k and the security parameter 1^{λ}. The output parameters are the master keys $\mathcal{MK}^{\mathsf{dev}}$ and the set $\mathcal{SK}^{\mathsf{dev}}$ of keys of all device trees. For each configuration c_k, the center invokes the key generation algorithm of an arbitrary secure SCBE scheme: $(\mathcal{MK}_k^{\mathsf{dev}}, \mathcal{SK}_k^{\mathsf{dev}}) \leftarrow \mathsf{KeyGenSCBE}(n_{c_k}, 1^{\lambda})$.

5.4 Join and Leave Operations

Join = Key Extraction. In the setup phase, each device obtains an individual set of keys. Before giving any details, we outline the approach for key extraction: Each device obtains keys in the configuration tree *and* keys in its device tree. In the configuration tree, the center provides the device with the keys related to its configuration. In the device tree, the device obtains the key material of its leaf node as defined in the underlying SCBE scheme.

The input parameters of the key extraction algorithm are the set of secret keys and the device index (k, i_k). The output is the set $\mathcal{SK}_{k,i_k}$ of secret keys of device d_{k,i_k} in *both* trees. The algorithm proceeds as follows: It first extracts the keys $\mathcal{SK}_k^{\mathsf{conf}} \leftarrow \mathsf{KeyExtrCS}(\mathcal{SK}^{\mathsf{conf}}, k)$ of configuration c_k in the configuration tree using the key extraction algorithm of the CS scheme. Then it extracts the keys $\mathcal{SK}_{k,i_k}^{\mathsf{dev}} \leftarrow \mathsf{KeyExtrSCBE}(\mathcal{SK}_k^{\mathsf{dev}}, i_k)$ of device i_k in device tree c_k using the key extraction algorithm of the selected SCBE scheme. The result is the device's secret key set $\mathcal{SK}_{k,i_k} := \{\mathcal{SK}_k^{\mathsf{conf}}, \mathcal{SK}_{k,i_k}^{\mathsf{dev}}\}$.[10]

Leave = Key Revocation. The center keeps track of revoked devices. Whenever it learns that a device d_{k,i_k} has been compromised, it adds it to the (initially empty) set $\mathcal{R}^{\mathsf{dev}}$ of revoked devices:

[8] and, as we will soon see, all inferior configurations.

[9] Specifically, we assume an SCBE scheme that fulfills the key-indistinguishability property.

[10] The update of the counter i_k of devices in c_k is straightforward: $i_k \leftarrow i_k + 1$.

$$\mathcal{R}_k^{\mathsf{dev}} := \{i_k \mid d_{k,i_k} \text{ is revoked}\} \quad \text{and} \quad \mathcal{R}^{\mathsf{dev}} := \{\mathcal{R}_0^{\mathsf{dev}}, \mathcal{R}_1^{\mathsf{dev}}, \ldots\}$$

The center maintains a similar revocation list for non-intact configurations:

$$\mathcal{R}^{\mathsf{conf}} := \{c_k \mid c_k \text{ is non-intact}\},$$

After introduction of the notational elements, we now give the actual revocation algorithm $\mathcal{R} \leftarrow \mathsf{Revoke}(\mathcal{R}, k, i_k)$, where $\mathcal{R} := \{\mathcal{R}^{\mathsf{dev}}, \mathcal{R}^{\mathsf{conf}}\}$. When d_{k,i_k} is revoked, the center first performs the following two steps:

$$\mathcal{R}_k^{\mathsf{dev}} \leftarrow \mathcal{R}_k^{\mathsf{dev}} \cup \{i_k\} \quad \text{and} \quad \mathcal{R}^{\mathsf{conf}} \leftarrow \mathcal{R}^{\mathsf{conf}} \cup \{c_k\}$$

Then it needs to update the pseudo-random chain of the revoked configuration c_k and declare all inferior configurations non-intact. The reason for this step is d_{k,i_k}'s hierarchical position: The revoked device has the keys of configuration c_k and—by following the pseudo-random chain—it can derive the keys of all inferior configurations. The center prevents this as follows: Each of the pseudo-random chains has $n_{p_g}^{\mathsf{leaves}}$ members (see Section 5.2). There are $\Delta k_{\mathsf{max}} = k \bmod n_{p_g}^{\mathsf{leaves}}$ inferior configurations from c_k to the end of its pseudo-random chain. Each of them is added to the list $\mathcal{R}^{\mathsf{conf}}$ of non-intact configurations.

5.5 Encryption

The encryption of a message involves an access structure based on properties and devices. The combination of both criteria and the cooperation of content provider and center unambiguously defines this structure. While the provider derives the allowed configurations from his business model, the center excludes all devices in the revocation list. The access structure needs to be expressed in terms of keys in the SCBE schemes. Obviously, this involves keys from configuration tree *and* device trees. First, we show how the content provider's business model determines the selection of keys in the configuration tree. Then, we outline how the center adds the selection of keys in the device trees which enforce revocation of individual devices.

The Provider's Access Rules. The content provider will be faced with the following situation: He wants to distribute content according to his business model in order to generate revenues. His business model will contain certain textual rules, e.g., "only allow access to devices with output interface X", "restrict access to a certain region" or "exclude insecure devices". We show how to combine these rules to arbitrary Boolean expressions of predicates over the properties, i.e., expressions with $\wedge$ and $\vee$ operators.

Consider a simple example where the content providers use three properties: region p_1, device model p_2 and data format p_3. Let there be 2 regions, 2 models and 3 formats. If a content provider's business model contains the two `or`-connected textual rules "either region 0 and model 1 or region 1 and model 0"

for a content item, independent of the format, we translate them into the following set of allowed configurations:

$$\mathcal{C}^{\mathsf{allow}} = \{(x_1, x_2, x_3) \mid \Big(x_1 \in \{0\} \ \wedge \ x_2 \in \{1\}\Big) \vee \Big(x_1 \in \{1\} \ \wedge \ x_2 \in \{0\}\Big)\}$$

As the data format p_3 wasn't relevant, we obtain the following list of allowed configurations c_k:

$$\boxed{(0,1,0)}\boxed{(0,1,1)}\boxed{(0,1,2)}\boxed{(1,0,0)}\boxed{(1,0,1)}\boxed{(1,0,2)}$$

In general, each rule α translates into a tuple $\mathcal{P}_\alpha^{\mathsf{allow}}$ of allowed values:

$$\mathcal{P}_1^{\mathsf{allow}} = (\{0\}, \{1\}, \{0,1,2\}) \quad \text{and} \quad \mathcal{P}_2^{\mathsf{allow}} = (\{1\}, \{0\}, \{0,1,2\})$$

Each tuple $\mathcal{P}_\alpha^{\mathsf{allow}}$ represents an `and`-statement and defines a set of configurations $\mathcal{C}_\alpha^{\mathsf{allow}}$ as follows:

$$\mathcal{P}_\alpha^{\mathsf{allow}} = (\mathcal{P}_{\alpha,1}^{\mathsf{allow}}, \ldots, \mathcal{P}_{\alpha,g}^{\mathsf{allow}}) \quad \text{s.t.} \quad \mathcal{C}_\alpha^{\mathsf{allow}} := \{(x_1, \ldots, x_g) | \ \textstyle\bigwedge_{j=1}^{g} x_j \in \mathcal{P}_{\alpha,j}^{\mathsf{allow}}\}$$

Finally, we denote the `or`-connection between the rules $\mathcal{P}_\alpha^{\mathsf{allow}}$ in set notation:

$$\mathcal{P}^{\mathsf{allow}} := \{\mathcal{P}_1^{\mathsf{allow}}, \mathcal{P}_2^{\mathsf{allow}}, \ldots\} \quad \text{s.t.} \quad \mathcal{C}^{\mathsf{allow}} := \{c_k | \ \textstyle\bigvee_{\alpha=1}^{|\mathcal{P}^{\mathsf{allow}}|} c_k \in \mathcal{C}_\alpha^{\mathsf{allow}}\}$$

The Center's Revocation Lists. When the provider has chosen $\mathcal{P}^{\mathsf{allow}}$, the center enhances this information with his revocation lists. If the provider unknowingly includes some non-intact configurations, then the center excludes these configurations from the access structure, but includes the non-revoked devices of these configurations. Remember that within a non-intact configuration, usually only a few devices are revoked while all others are still privileged.

Encryption Overview. The encryption algorithm consists of four subroutines. We first outline the approach before giving details: First, the center declares *all* configurations to be excluded. The content provider chooses the rule for allowed configurations in the form $\mathcal{P}^{\mathsf{allow}}$, thus defining $\mathcal{C}^{\mathsf{allow}}$. The center removes $\mathcal{C}^{\mathsf{allow}}$ from the exclusion list. Second, it adds all non-intact configurations to the exclusion list. Third, it takes advantage of the hierarchy and excludes all configurations that can descend in the hierarchy and follow a pseudo-random chain. Fourth, it encrypts the message in the configuration tree *and* device trees.

Encryption Steps. We detail the four steps:

1. This subroutine excludes all but the allowed configurations: $\mathcal{C}^{\mathsf{excl}} \leftarrow \mathcal{C} \setminus \mathcal{C}^{\mathsf{allow}}$
2. This subroutine adds all revoked configurations: $\mathcal{C}^{\mathsf{excl}} \leftarrow \mathcal{C}^{\mathsf{excl}} \cup \mathcal{R}^{\mathsf{conf}}$
3. This subroutine leverages the hierarchy which the pseudo-random chains induce. Among the included configurations, only the most inferior configuration of any chain needs to be covered, while all superior configurations can descend in

the hierarchy. For each pseudo-random chain in the configuration tree, we first go to the most inferior configuration, i.e., the left-most leaf. Moving to superior configurations, we look for the first leaf corresponding to an included configuration. We let this configuration be included, but exclude all superior configurations because they can descend in the hierarchy and follow the pseudo-random chain.

After this subroutine, the excluded configurations contain all disallowed configurations, the non-intact allowed configurations and the intact allowed configurations that are in a superior position.

4. This subroutine generates the message header in both the configuration tree *and* the device trees. To do so, it encrypts the session key k^{sess} with keys in both trees. In the configuration tree, it addresses all intact allowed configurations

$$(\mathcal{I}^{\mathsf{conf}}, \mathcal{K}^{\mathsf{conf}}) \leftarrow \mathsf{EncCS}(\mathcal{SK}^{*\mathsf{conf}}, \mathcal{C}^{*\mathsf{excl}}, k^{\mathsf{sess}}) \ ,$$

where $\mathcal{C}^{*\mathsf{excl}}$ is the set of indices of $\mathcal{C}^{\mathsf{excl}}$'s members like k is the index of c_k.[11] Although only the most inferior intact configurations are directly covered, *all* intact configurations are addressed because they can use their superiority and descend in the pseudo-random chain.

In the device trees, it addresses all non-intact allowed configurations. For each configuration c_k that is non-intact and allowed, the center covers the non-revoked devices by invoking the SCBE scheme in this device tree:

$$(\mathcal{I}_k^{\mathsf{dev}}, \mathcal{K}_k^{\mathsf{dev}}) \leftarrow \mathsf{EncSCBE}(\mathcal{SK}_k^{\mathsf{dev}}, \mathcal{R}_k^{\mathsf{dev}}, k^{\mathsf{sess}})$$

5.6 Decryption

Running the decryption algorithm, a non-revoked device d_{k,i_k} of an allowed configuration c_k decrypts the session key k^{sess}. The input variables are the device's index (k, i_k), its set of secret keys $\mathcal{SK}_{k,i_k}$ and the message header[12] $(\mathcal{I}, \mathcal{K})$. The output is the session key k^{sess} if c_k was allowed and d_{k,i_k} was non-revoked at encryption time.

The algorithm proceeds as follows: First, it checks whether c_k is covered in the configuration tree by evaluating the indexing information $\mathcal{I}^{\mathsf{conf}}$. Second, it verifies whether it can reach one of the covered configurations by leveraging c_k's hierarchical position and descending a pseudo-random chain. Third, it checks whether c_k was allowed, but non-intact:

1. $k^{\mathsf{sess}} \leftarrow \mathsf{DecCS}(k, \mathcal{SK}_k^{*\mathsf{conf}}, \mathcal{I}^{\mathsf{conf}}, \mathcal{K}^{\mathsf{conf}})$: This step succeeds if c_k was among the covered configurations. That is, c_k is the most inferior of the intact allowed configurations in its pseudo-random chain.

2. This step succeeds if c_k is superior to a covered configuration. The algorithm descends to all inferior configurations in c_k's pseudo-random chain. For each inferior configuration $c_{k'}$, the algorithm derives the secret keys $\mathcal{SK}_{k'}^{\mathsf{conf}}$ from its

[11] For example, $\mathcal{C}^{\mathsf{excl}} = \{c_3, c_9, c_{12}\}$ implies $\mathcal{C}^{*\mathsf{excl}} = \{3, 9, 12\}$.

[12] The message header consists of $\mathcal{I} := \{\mathcal{I}^{\mathsf{conf}}, \mathcal{I}^{\mathsf{dev}}\}$ with $\mathcal{I}^{\mathsf{dev}} := \bigcup_k \{\mathcal{I}_k^{\mathsf{dev}}\}$ and $\mathcal{K} := \{\mathcal{K}^{\mathsf{conf}}, \mathcal{K}^{\mathsf{dev}}\}$ with $\mathcal{K}^{\mathsf{dev}} := \bigcup_k \{\mathcal{K}_k^{\mathsf{dev}}\}$.

own secret keys $\mathcal{SK}_k^{\mathsf{conf}}$ by applying the generator $\mathsf{G_L}()$ to c_k's label. This is possible due to the construction of the pseudo-random chains according to (1). Then it replaces the derived label with the actual key using the generator $\mathsf{G_K}()$ and obtains $\mathcal{SK}_{k'}^{*\mathsf{conf}}$. Finally, the algorithm tries to decrypt with the derived secret key: $k^{\mathsf{sess}} \leftarrow \mathsf{DecCS}(k', \mathcal{SK}_{k'}^{*\mathsf{conf}}, \mathcal{I}^{\mathsf{conf}}, \mathcal{K}^{\mathsf{conf}})$. Note that the first two steps only affect the configuration tree and allow decryption in intact configurations.
3. $k^{\mathsf{sess}} \leftarrow \mathsf{DecSCBE}(i_k, \mathcal{SK}_{k,i_k}^{\mathsf{dev}}, \mathcal{I}_k^{\mathsf{dev}}, \mathcal{K}_k^{\mathsf{dev}})$: This step succeeds if d_{k,i_k} is a non-revoked device in a non-intact allowed configuration, i.e., $i_k \notin \mathcal{R}_k^{\mathsf{dev}}$ and $c_k \in \mathcal{R}^{\mathsf{conf}} \cap \mathcal{C}^{\mathsf{allow}}$. Although the keys in the configuration tree are compromised and therefore cannot be used, a non-revoked device d_{k,i_k} still has the keys in its device tree. If the device is non-revoked and the configuration allowed (although non-intact), then decryption succeeds in the device tree of configuration c_k.

5.7 Efficiency Gains

The third possibility of Section 3.3 was to order a standard CS or SD tree according to the configurations (Part 2 of Figure 1) and to allow each configuration to have its own depth. Without a hierarchical property, this possibility still has the shorter header, provided that it is instantiated with an SD tree. The reason is that, without a hierarchical property, our proposed solution will not add any pseudo-random chains and therefore be equivalent to the CS scheme. Obviously, this leads to a longer header than that of the SD scheme [8].

However, as soon as one of the properties is hierarchical, our proposed solution is more efficient than CS and SD. In the hierarchical setting, it reduces the header length, while storage requirements and computational complexity are identical. We discuss the header length reduction in two steps; in both steps we assume that the content providers base their decision on properties and that one of the properties is hierarchical. In the first step, we discuss the effect of having a hierarchical property among the properties. We assume that revocation is unnecessary, i.e., all devices are compliant and needn't be revoked. In other words, only the properties matter. In the second step, we release this assumption and generalize by allowing revocation of arbitrary devices. In other words, properties *and* revocation of individual devices matter.

1. As revocation depends merely on the properties, the proposed scheme performs revocation in the configuration tree. Let l_g be the number of hierarchy levels induced by the hierarchical property. In this case, the proposed scheme achieves a header length reduction of a factor between 1 and $\lceil \log_2(l_g) \rceil$ compared to CS and between 1 and $\lceil \frac{1}{2} \cdot \lceil \log_2(l_g) \rceil \rceil$ compared to SD. For example, with $l_g = 16$ hierarchy levels our scheme reduces the header length by a factor of up to 4 compared to CS and up to 2 compared to SD. Due to space constraints, we give the proof of this claim in a technical report [1].

2. As revocation depends on properties *and* individual devices, the proposed scheme performs revocation in both the configuration tree *and* the device trees. The beneficial effect of ordering the devices according to their properties is lost. In a non-intact configuration, the proposed scheme therefore degrades to the

performance of the SCBE scheme with which the corresponding device tree is instantiated. However, in intact configurations we can still address all devices with the reduced header length discussed in Step 1.

5.8 Security

We only need to consider the effect of adding pseudo-random chains in the configuration tree. Each device tree directly inherits the security of the SCBE scheme with which it is instantiated. We assume the device trees' keys to be indistinguishable as defined in [8] because all candidate schemes [8, 14, 17] fulfill this requirement. As for the pseudo-random chains in the configuration tree, we show that even a coalition of all revoked devices in allowed configurations and all devices in disallowed configurations cannot distinguish the encrypted session key from a random key:

Theorem 1. *If the generators* $\mathsf{G_L}$ *and* $\mathsf{G_K}$ *fulfill Definition 5, then our PBBE scheme is IND-CCA1-secure.*

We give the proof of this theorem in a technical report [1]. The definition of IND-CCA1 security for SCBE schemes can be found in [8, Section 5.2, Definition 9].

5.9 Extensions and Future Work

We highlight some possible extensions of our proposed scheme in the technical report [1]. They refer to the position of the hierarchical property, the intermediate levels within the layer of the hierarchical property, the use of several instances of the configuration tree, and the use of l_j-ary trees instead of binary trees.

6 Conclusion

We have formalized the concept of properties in the context of broadcast encryption. In the presence of a hierarchical property such as tamper resistance, we have proposed a property-based broadcast encryption scheme that reduces the message header length compared to existing schemes by a factor that is logarithmic in the number of hierarchy levels.

References

1. Adelsbach, A., Huber, U., Sadeghi, A.R.: Property-based broadcast encryption for multi-level security policies. Technical Report, full version of this paper, Horst Görtz Institute for IT Security (2005) `http://www.prosec.rub.de/publications`.
2. Miller, M.L., Cox, I.J., Linnartz, J.P.M.G., Kalker, T.: A review of of watermarking principles and practices. In Parhi, K.K., Nishitani, T., eds.: Digital Signal Processing for Multimedia Systems. IEEE (1999) 461–485
3. Petitcolas, F.A.: Digital watermarking. In Becker, E., Buhse, W., Günnewig, D., Rump, N., eds.: Digital Rights Management: Technological, Economic, Legal and Political Aspects. Volume 2770 of Lecture Notes in Computer Science., Springer (2003) 81–92

4. Kundur, D., Karthik, K.: Video fingerprinting and encryption principles for digital rights management. Proceedings of the IEEE **92**(6) (2004) 918–932
5. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In Desmedt, Y., ed.: CRYPTO 1994. Volume 839 of Lecture Notes in Computer Science., Springer (1994) 257–270
6. Naor, M., Pinkas, B.: Threshold traitor tracing. In Krawczyk, H., ed.: CRYPTO 1998. Volume 1462 of Lecture Notes in Computer Science., Springer (1998) 502–517
7. Fiat, A., Naor, M.: Broadcast encryption. In Stinson, D.R., ed.: CRYPTO 1993. Volume 773 of Lecture Notes in Computer Science., Springer (1994) 480–491
8. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In Kilian, J., ed.: CRYPTO 2001. Volume 2139 of Lecture Notes in Computer Science., Springer (2001) 41–62
9. Anderson, R.J., Kuhn, M.: Tamper resistance—a cautionary note. In Tygar, D., ed.: USENIX Electronic Commerce 1996, USENIX (1996) 1–11
10. Anderson, R.J.: Security Engineering: A Guide to Building Dependable Distributed Systems. First edn. John Wiley & Sons, New York, USA (2001)
11. Popescu, B.C., Crispo, B., Tanenbaum, A.S.: Support for multi-level security policies in DRM architectures. In: NSPW 2004, ACM Press (2005) 3–9
12. Digital Content Protection, LLC: High-bandwidth digital content protection system. Specification Revision 1.1 (2003) URL `http://www.digital-cp.com/home/HDCPSpecificationRev1_1.pdf`.
13. Digital Transmission Licensing Administrator, LLC: Digital transmission content protection specification volume 1. Specification Revision 1.4 (2005) URL `http://www.dtcp.com/data/info%2020050228%20dtcp%20vol%201%20%201p4.pdf`.
14. Halevy, D., Shamir, A.: The LSD broadcast encryption scheme. In Yung, M., ed.: CRYPTO 2002. Volume 2442 of Lecture Notes in Computer Science., Springer (2002) 47–60
15. Berkovits, S.: How to broadcast a secret. In Davies, D.W., ed.: EUROCRYPT 1991. Volume 547 of Lecture Notes in Computer Science., Springer (1991) 535–541
16. Dodis, Y., Fazio, N.: Public key broadcast encryption for stateless receivers. In Feigenbaum, J., ed.: Digital Rights Management Workshop. Volume 2696 of Lecture Notes in Computer Science., Springer (2003) 61–80
17. Jho, N.S., Hwang, J.Y., Cheon, J.H., Kim, M.H., Lee, D.H., Yoo, E.S.: One-way chain based broadcast encryption schemes. In Cramer, R., ed.: EUROCRYPT 2005. Volume 3494 of Lecture Notes in Computer Science., Springer (2005) 559–574

Efficient Cryptographic Protocol Design Based on Distributed El Gamal Encryption

Felix Brandt

Stanford University, Stanford CA 94305, USA
brandtf@cs.stanford.edu

Abstract. We propose a set of primitives based on El Gamal encryption that can be used to construct efficient multiparty computation protocols for certain low-complexity functions. In particular, we show how to privately count the number of true Boolean disjunctions of literals and pairwise exclusive disjunctions of literals. Applications include efficient two-party protocols for computing the Hamming distance of two bitstrings and the greater-than function. The resulting protocols only require 6 rounds of interaction (in the random oracle model) and their communication complexity is $\mathcal{O}(kQ)$ where k is the length of bit-strings and Q is a security parameter. The protocols are secure against active adversaries but do not provide fairness. Security relies on the decisional Diffie-Hellman assumption and error probability is negligible in Q.

1 Introduction

Secure multiparty computation (MPC) deals with protocols that allow a group of agents to jointly compute a function of their individual private inputs, so that only the function value is revealed in the end. Since Yao's and Goldreich et al's seminal completeness results [Yao86, GMW87], it is well known that any function can be computed securely if trapdoor permutations exist. However, the general constructions in [Yao86, GMW87] have proven to be rather inefficient and unpractical. It has been shown that general MPC is feasible in a *constant* number of rounds with polynomial communication complexity for various settings such as 2-party MPC (without fairness) [Lin01] or n-party MPC (with an honest majority) [BMR90]. Although theoretically interesting, the constructions of the underlying proofs do not yield practical constant-round MPC schemes due to the extensive use of generic proofs of knowledge.

In this paper, we propose a set of cryptographic techniques that enable the efficient computation of "low-complexity" functions in the presence of active adversaries. These techniques, some of which have been known for some time, can be used straightforwardly to construct round-efficient protocols for the equality function (solving the so-called socialist millionaires' problem), the Boolean OR function (*e.g.*, for veto voting), or the maximum function. Furthermore, we show how to privately count the number of true Boolean disjunctions of literals and pairwise exclusive disjunctions of literals. Applications include efficient

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 32–47, 2006.
© Springer-Verlag Berlin Heidelberg 2006

two-party protocols for computing the Hamming distance of two bitstrings and the greater-than function (thus providing a solution to Yao's millionaires' problem [Yao82] in which two millionaires (Alice and Bob) want to find out who is richer without revealing their wealth). Our primary objective when designing these protocols was to minimize round complexity as interaction over a computer network is usually the most time-consuming operation in distributed protocols. Our protocol for the millionaires' problem only requires 6 rounds of interaction (in the random oracle model) and its communication complexity is $\mathcal{O}(kQ)$ where k is the length of bit-strings to be compared and Q is a security parameter. To the best of our knowledge, this is the most efficient constant-round protocol for the millionaires' problem. Under reasonable conditions (see Section 5.2), each party only needs to communicate about 73 Kbytes of data. This is achieved by exploiting the homomorphicity of the underlying encryption scheme when evaluating a modified Boolean formula for the greater-than function. The protocol is correct with error probability $\mathcal{O}(2^{-Q})$. It does not provide fairness, *i.e.*, one party might learn the outcome and leave the other party uninformed by quitting the protocol prematurely. However, fairness can be obtained by using known standard techniques of gradual exchange (see *e.g.*, [GMY04]). For this paper, we assume that there is either a fairness-providing third-party, *i.e.*, a party that does not reveal information or quits prematurely,[1] or that only one of the agents, say Alice, is supposed to learn the function value.

Many representations for *general* secure MPC have been suggested in the literature: Boolean circuits [Yao86], arithmetic circuits [GMW87], branching programs [Kil88], low degree polynomials [BFKR90], randomizing polynomials [IK00], *etc.* Our approach differs in that we provide a set of efficient building blocks (distributed homomorphic encryption, random exponentiation, and verifiable mixing) for which there exist efficient proofs of correctness and use these to construct *special-purpose* protocols for a limited, but nevertheless relevant, group of functions.

Our primitives are built around El Gamal encryption [El 85] because it allows for the efficient generation of distributed keys and because encrypted values can easily be exponentiated with jointly created random numbers. The building blocks can be used for any number of parties. We consider full privacy, *i.e.*, $(n-1)$-privacy, rather than threshold privacy.[2] Any party that deviates from the prescribed protocol can be identified (because it fails to prove its correctness in zero-knowledge) and removed from the set of participants. There are very efficient (honest-verifier) zero-knowledge proofs to show the correctness of each protocol step (see Section 3.2).

The remainder of this paper is structured as follows. In Section 2, we review related work. Section 3 contains building blocks to be used in the protocols. Basic protocols consisting of these primitives are presented in Section 4 whereas more sophisticated protocols based on the evaluation of simple Boolean formulae

[1] Please note that such a third-party will not be able to learn any information besides the outcome.

[2] Threshold privacy can easily be obtained by using standard secret sharing techniques.

are proposed in Section 5. The paper concludes with a summary of the results and a brief outlook in Section 6.

2 Related Work

Various recent advances in efficiency for cryptographic protocols build on homomorphic encryption (*e.g.*, [JJ00, CDN01, Fis01, ST04, BGN05]). Most of these protocols ([JJ00] and [ST04] are exceptions) use factorization-based encryption schemes like Paillier encryption [Pai99] due to their versatility. However, these schemes require the joint generation of an RSA modulus before the actual computation begins. Even though protocols for this task improved over the years [BF97, Gil99, DK01, ACS02], they remain inefficient and unpractical, especially when having to tolerate active adversaries.[3] Distributed key generation for discrete logarithm based schemes like El Gamal, on the other hand, is straightforward and very efficient (see [Ped91, GJKR99, GJKR03]).

Our protocols bear some similarities to mix-and-match [JJ00] as distributed El Gamal random exponentiation and the mixing of ciphertexts are also part of the mix-and-match framework. However, in contrast to our work, mix-and-match allows *general* MPC. As a consequence, their approach is more versatile but less efficient in special cases. *E.g.*, a straightforward mix-and-match implementation of a millionaires' protocol is considerably less efficient in both round complexity and communication complexity than our protocol because mix-and-match does not take advantage of El Gamal's homomorphicity (except for the plaintext equality test).

[BST01] give a protocol for the *socialist millionaires'* problem in which two parties compute whether their inputs are equal. Security is based on the decisional Diffie-Hellman problem and a similar protocol (disregarding fairness) can be constructed by using the techniques presented in this paper (see Section 4.1).

Since the publication of Yao's original protocols in [Yao82], numerous solutions to the *millionaires' problem* have been proposed. While the complexity of some protocols is exponential in k [Sch96], others do not consider active adversaries [Fis01, IG03, LT05] or have quadratic complexity [IG03]. The protocol proposed in [Fis01] is quite efficient but also relies on the prior setup of a distributed RSA modulus which (even for just two parties) requires a large amount of communication. Nevertheless, we adopted [Fis01]'s idea of securely evaluating the greater-than function by shuffling k equality tests. The efficiency of Lin and Tzeng's recent protocol [LT05], which can be based on El Gamal encryption, is similar to that of our protocol, although it is based on a quite different, interesting idea. However, their model only allows passive adversaries and the protocol cannot straightforwardly be turned into one that is secure against active adversaries.

[3] Communication complexity in these protocols always contains high orders of the security parameter. For example, the 2-party key generation proposed in [Gil99] requires about 42MB of data to be sent for setting up a 1024-bit key (while only tolerating *passive* adversaries).

There are *randomized* protocols for the millionaires' problem (*e.g.*, [NN01, PBDL04]) whose communication complexity may be less than linear in the input size [NN01]. However, these protocols cannot provide constant round complexity.[4] The same holds for recently proposed protocols based on evaluating circuits that contain so-called conditional gates [ST04]. The communication complexity of [ST04]'s solution for the Millionaires' problem is comparable to ours but their protocol requires k rounds of interaction.

It is difficult to compare our techniques with advanced two-party protocols based on Yao's "garbled circuit" construction [Yao86], *e.g.*, [NPS99, CC00]. Naturally, these protocols have the advantage of being universally applicable. However, [NPS99] relies on the inefficient "cut-and-choose" technique to prove circuits correct and even [CC00], which uses zero-knowledge proofs instead of cut-and-choose, is less efficient than our approach because the correctness of every single gate has to be proven. Even for simple functions, like the greater-than function, the number of gates is relatively high (at least $5k - 4$ [KO02]), resulting in substantially higher complexity than our protocol for the same function. Moreover, our techniques have the advantage of being applicable to settings with any number of parties which is generally not possible with garbled circuit protocols.

3 Building Blocks

This section contains building blocks to enable the construction of efficient protocols for simple functions. In the heart of the system lies El Gamal encryption because it allows for the easy generation of distributed keys and because encrypted values can be exponentiated with a shared random number in a single round. This random exponentiation will be used as a blinding step in our protocols as it transforms every plaintext, *except* 1, into a meaningless random number.

We suggest the following general methodology for efficiently computing low-complexity functions: All parties publish encryptions of their inputs in a representation—*e.g.*, binary (see Section 5) or unary (see Section 4.3)—that at the same time allows efficient proofs of correctness and further processing in order to compute the function outcome. By exploiting the homomorphic property of the underlying encryption scheme, participants compute a vector of encrypted values that contains the function outcome but may also contain additional, unwanted information. In order to get rid of this information, all agents jointly execute random exponentiations for each vector component. Finally, if needed, components can be shuffled to only reveal if (or how many) vector components equal 1. We stress the fact the we do not rely on a strict Boolean or arithmetic representation of the function. We rather suggest a bottom-up approach, *i.e.*, trying to represent the function by using the mentioned limited set of primitives.

[4] Furthermore, the protocol in [PBDL04] is flawed because the proposed primitive "complex zero test" reveals statistical information.

3.1 El Gamal Encryption

El Gamal cipher [El 85] is a probabilistic and homomorphic public-key cryptosystem. Let p and q be large primes so that q divides $p-1$. $\mathbb{G}_q$ denotes $\mathbb{Z}_p^*$'s unique multiplicative subgroup of order q.[5] All computations in the remainder of this paper are modulo p unless otherwise noted. The *private key* is $x \in \mathbb{Z}_q$, the *public key* is $y = g^x$ ($g \in \mathbb{G}_q$ is an arbitrary, publicly known element). A message $m \in \mathbb{G}_q$ is *encrypted* by computing the ciphertext tuple

$$(\alpha, \beta) = (my^r, g^r)$$

where r is an arbitrary random number in $\mathbb{Z}_q$, chosen by the encrypter. A message is *decrypted* by computing

$$\frac{\alpha}{\beta^x} = \frac{my^r}{(g^r)^x} = m.$$

El Gamal is homomorphic as the component-wise product of two ciphertexts $(\alpha\alpha', \beta\beta') = (mm'y^{r+r'}, g^{r+r'})$ represents an encryption of the plaintexts' product mm'. It has been shown that El Gamal is semantical secure, *i.e.*, it is computationally infeasible to distinguish between the encryptions of any two given messages, if the decisional Diffie-Hellman problem is intractable [TY98]. We will use functions $E(\cdot)$ and $D(\cdot)$ to denote the encryption and the decryption of plain- and ciphertexts, respectively.

In the following, we describe how to apply the El Gamal cryptosystem as a fully private, *i.e.*, non-threshold, MPC scheme for n agents.[6] If a value represents an additive share, this is denoted by a "+" in the index, whereas multiplicative shares are denoted by "×". Underlying zero-knowledge proofs will be presented in the next section.

Distributed key generation [Ped91]: Each participant chooses x_{+i} at random and publishes $y_{\times i} = g^{x_{+i}}$ along with a zero-knowledge proof of knowledge of $y_{\times i}$'s discrete logarithm. The public key is $y = \prod_{i=1}^{n} y_{\times i}$, the private key is $x = \sum_{i=1}^{n} x_{+i}$. This requires n multiplications, but the computational cost of multiplications is usually negligible in contrast to exponentiations. Broadcast round complexity and exponentiation complexity of the key generation are $\mathcal{O}(1)$.[7]

Distributed decryption: Given an encrypted message (α, β), each participant publishes $\beta_{\times i} = \beta^{x_{+i}}$ and proves its correctness by showing the equality of logarithms of $y_{\times i}$ and $\beta_{\times i}$. The plaintext can be derived by computing $\frac{\alpha}{\prod_{i=1}^{n} \beta_{\times i}}$. Like key generation, decryption can be performed in a constant number of rounds, requiring n multiplications and one exponentiation.

[5] We will focus on multiplicative subgroups of finite fields here, although El Gamal can also be based on other groups such as elliptic curve groups.

[6] Please note that this multiparty scheme is limited in the sense that it does not allow the computation of *arbitrary* functions.

[7] Finding "unbiased" parameters p and q requires no extra communication in the random oracle model.

Random Exponentiation: A given encrypted value (α, β) can easily be raised to the power of an unknown random number $M = \sum_{i=1}^{n} m_{+i}$ whose addends can be freely chosen by the participants if each bidder publishes $(\alpha^{m_{+i}}, \beta^{m_{+i}})$ and proves the equality of logarithms. The product of published ciphertexts yields (α^M, β^M) in a single step. The computational cost is two exponentiations and $2n$ multiplications.

When adding a commitment round (*e.g.*, using [Ped91]) during key generation and random exponentiation, security of these primitives is evident. We presume that even without these commitment rounds, security is preserved (see Proposition 1).

3.2 Zero-Knowledge Proofs

In order to obtain security against *malicious* or so-called *active* adversaries, agents are required to prove the correctness of each protocol step. One of the objectives when designing the protocols presented in Sections 4 and 5 was to enable *efficient* proofs of correctness for protocol steps. In fact, the proposed protocols can be proven correct by only using so-called Σ-protocols which just need three rounds of interaction [Dam02, CDS94]. Σ-protocols are not known to be zero-knowledge, but they satisfy the weaker property of *honest-verifier* zero-knowledge. This suffices for our purposes as we can use the Fiat-Shamir heuristic [FS87] to make these proofs non-interactive. As a consequence, the obtained proofs are indeed zero-knowledge *in the random oracle model* and only consist of a single message.[8] We will make use of the following four Σ-protocols.

Proof of knowledge of a discrete logarithm. This is a classic Σ-protocol by Schnorr [Sch91]. Alice and Bob know v and g, but only Alice knows x, so that $v = g^x$.

1. Alice chooses z at random and sends $a = g^z$ to Bob.
2. Bob chooses a challenge c at random and sends it to Alice.
3. Alice sends $r = (z + cx) \mod q$ to Bob
4. Bob checks that $g^r = av^c$.

Alice needs to send $\log p + \log q$ bits.

Proof of equality of two discrete logarithms. When executing the previous protocol in parallel, the equality of two discrete logarithms can be proven [CP92]. Alice and Bob know v, w, g_1, and g_2, but only Alice knows x, so that $v = g_1^x$ and $w = g_2^x$.

[8] The additional assumption of a random oracle is only made for reasons of efficiency. Alternatively, we could employ non-interactive zero-knowledge proofs in the *common random string model* (see [DDO+01] and references therein) to obtain non-interactiveness. However, it has become common practice to use secure hash functions like MD5 or SHA-1 as random oracles for practical applications.

1. Alice chooses z at random and sends $a = g_1^z$ and $b = g_2^z$ to Bob.
2. Bob chooses a challenge c at random and sends it to Alice.
3. Alice sends $r = (z + cx) \mod q$ to Bob
4. Bob checks that $g_1^r = av^c$ and that $g_2^r = bw^c$.

Alice needs to send $2\log p + \log q$ bits. It is possible to show the equality of any polynomial number of discrete logarithms in parallel. Thus, for showing that the discrete logarithms of k values are equal, Alice only sends $k\log p + \log q$ bits.

Proof that an encrypted value is one out of two values. The following protocol was proposed by Cramer et al [CGS97]. Alice proves that an El Gamal encrypted value $(\alpha, \beta) = (my^r, g^r)$ either decrypts to 1 or to a fixed value $z \in \mathbb{G}_q$ without revealing which is the case, in other words, it is shown that $m \in \{1, z\}$.

1. If $m = 1$, Alice chooses r_1, d_1, w at random and sends (α, β), $a_1 = g^{r_1}\beta^{d_1}$, $b_1 = y^{r_1}\left(\frac{\alpha}{z}\right)^{d_1}$ and $a_2 = g^w$, $b_2 = y^w$ to Bob.
   If $m = z$, Alice chooses r_2, d_2, w at random and sends (α, β), $a_1 = g^w$, $b_1 = y^w$, $a_2 = g^{r_2}\beta^{d_2}$, and $b_2 = y^{r_2}\alpha^{d_2}$ to Bob.
2. Bob chooses a challenge c at random and sends it to Alice.
3. If $m = 1$, Alice sends d_1, $d_2 = c - d_1 \mod q$, r_1, and $r_2 = w - rd_2 \mod q$ to Bob.
   If $m = z$, Alice sends $d_1 = c - d_2 \mod q$, d_2, $r_1 = w - rd_1 \mod q$, and r_2 to Bob.
4. Bob checks that $c = d_1 + d_2 \mod q$, $a_1 = g^{r_1}\beta^{d_1}$, $b_1 = y^{r_1}\left(\frac{\alpha}{z}\right)^{d_1}$, $a_2 = g^{r_2}\beta^{d_2}$, and $b_2 = y^{r_2}\alpha^{d_2}$.

The total amount of bits Alice sends to Bob is $4\log p + 4\log q$.

Verifiable shuffle of k encrypted values. A shuffle is a rearrangement and reencryption of input ciphertexts. By proving such a shuffle correct, a party can verifiably rearrange a vector of ciphertexts without revealing the applied permutation. [Gro03] proposed a very efficient way of proving the correctness of a shuffle of El Gamal encryptions in honest-verifier zero-knowledge (in fact, the proof is shorter than the vector itself). As the proof is *public-coin* honest-verifier zero-knowledge, it can be executed in a single round in the random oracle model. Alice needs to send $k(\log p + \log q) + 6\log p + 3\log q$ bits to prove the correctness of a shuffle consisting of k ciphertexts. This primitive will be used as a 2-server mix-net in Section 5 in order to hide which component of a vector equals 1.

4 Basic Protocols

In order to demonstrate the applicability of the proposed techniques, we briefly sketch three 4-round protocols that compute the equality-, the OR-, and the maximum-function, respectively.

4.1 Socialist Millionaires' Protocol

Suppose Alice and Bob want to compute the equality function $f(b_1, b_2) = [b_1 = b_2]$. This problem is also known as the socialist millionaires' problem [BST01] and can be solved by executing the following protocol.[9]

- Round 1: Alice and Bob generate a distributed pair of El Gamal keys
- Round 2: Both parties publish El Gamal encryptions of their inputs: $E(b_1)$ and $E(b_2)$.
- Round 3: They jointly compute $A = \left(\frac{E(b_1)}{E(b_2)}\right)^M$ where M is a random number not known to Alice or Bob (see Section 3.1).
- Round 4: Both parties jointly decrypt A. If $D(A) = 1$, both inputs were equal. Otherwise, $D(A)$ is a meaningless random number.

4.2 Veto Protocol

A variation of the previous protocol can be used for veto voting (in other words, the Boolean OR-function): $f(b_1, b_2, \ldots, b_n) = \bigvee_{i=1}^{n} b_i$. Let $Y \in \mathbb{G}_q \backslash \{1\}$ be a publicly known constant. Now, each voter i submits $E(b_i)$ where b_i is 1 if voter i agrees with the issue at hand or Y if he does not agree. The correctness of each vote, *i.e.*, $D(E(b_i)) \in \{1, Y\}$, can be proven by using the zero-knowledge proof given in Section 3.2. Voters than jointly decrypt $\left(\prod_{i=1}^{n} E(b_i)\right)^M$ and only learn whether they unanimously agree or not. No other information is revealed, not even to any (strict) subset of agents.

4.3 Maximum Protocol

Consider a group of n parties that wants to compute the maximum of their private input values: $f(b_1, b_2, \ldots, b_n) = \max\{b_1, b_2, \ldots, b_n\}$. By using a *unary* representation of numbers, this task can be accomplished by the following protocol. Let $\{1, 2, \ldots, k\}$ denote the set of possible input values and let $Y \in \mathbb{G}_q \backslash \{1\}$ again be a publicly known constant. Each participant i publishes $E(b_{ij})$ where $b_{ij} = Y$ if $b_i = j$ or $b_{ij} = 1$ otherwise. Agent i can efficiently prove the correctness of his input by showing that $\forall j \in \{1, 2, \ldots, k\} : \; D(E(b_{ij})) \in \{1, Y\}$ (Section 3.2) and that $\prod_{j=1}^{k} E(b_{ij}) = E(Y)$ (Section 3.2). Then, all agents jointly compute

$$A_j = \left(\prod_{i=1}^{n} \prod_{d=j}^{k} E(b_{id})\right)^{M_j} \qquad \forall j \in \{1, 2, \ldots, k\}$$

where, as above, M_j are jointly created random numbers. For all j greater than the maximum, $D(A_j)$ is equal to 1. All other $D(A_j)$ are random numbers. Clearly, the drawback of this protocol is that its communication complexity is linear in k, *i.e.*, exponential in the length of bitstrings. Nevertheless, it can be practical for small k.

[9] Similar protocols previously appeared in various contexts, *e.g.*, password authentication key exchange or the "plaintext equality test" in [JJ00].

5 Counting Boolean Disjunctions of Literals

In this section, we will show how the primitives defined in Section 3 can be used to evaluate simple Boolean expressions. Consider n parties whose inputs are bitstrings b_i of length k. We define $E[b]$ to be an (El Gamal) encryption of bit b if $E[0]$ decrypts to 1 and $E[1]$ decrypts to any other number in $\mathbb{G}_q$:

$$D[E[b]] \in \begin{cases} \{1\} & \text{if } b = 0 \\ \mathbb{G}_q \backslash \{1\} & \text{otherwise} \end{cases}.$$

As in Section 4.3, let Y be an arbitrary, publicly known, fixed element of $\mathbb{G}_q \backslash \{1\}$. Before the actual protocol starts each agent publishes encryptions of his individual input bits so that

$$E[b_{ij}] = \begin{cases} E(1) & \text{if } b_{ij} = 0 \\ E(Y) & \text{otherwise} \end{cases}.$$

The correctness of inputs can be efficiently proven by showing that each ciphertext either decrypts to 1 or to Y without revealing which case holds (Section 3.2). Based on this representation, we can count the number of true Boolean disjunctions of literals and pairwise exclusive disjunctions of literals by computing

$$f(b_1, b_2, \ldots, b_n) = \#\left(\bigvee_r L_r \vee \bigvee_{s,t} (L_s \oplus L_t) \right) \tag{1}$$

where $L_r, L_s, L_t \in \{b_{ij}, \neg b_{ij}\}$ for all $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, k\}$ and $\#$ is a *count operator* that counts the number of true expressions. The individual operators can be implemented as follows.

- **Negations**
  The Boolean negation of *input* bits can be computed by dividing Y by the input bit's encryption, *i.e.*,

$$E[\neg b_{ij}] = \frac{Y}{E[b_{ij}]}.$$

- **Disjunctions**
  As in Section 4.2, the product of ciphertexts yields the logical OR of the corresponding plaintext bits:

$$E\left[\bigvee_r L_r \right] = \prod_r E[L_r].$$

- **Exclusive Disjunctions**
  The exclusive OR of *pairs of literals* can be computed by dividing the encryptions of these bits,

$$E\left[L_s \oplus L_t\right] = \frac{E[L_s]}{E[L_t]}.$$

When combining these exclusive ORs via the disjunction operator, it has to be made sure that two encryptions that represent $E[1]$ do not "accidently" multiply to $E[0]$. This can be achieved by raising the dth factor to the 2^{d-1}th power:

$$E\left[\bigvee_{d=1}^{k}(L_{s_d} \oplus L_{t_d})\right] = \prod_{d=1}^{k}\left(\frac{E[L_{s_d}]}{E[L_{t_d}]}\right)^{2^{d-1}}.$$

– **Count Operator**
Finally, we can count the number of true bits in a vector of encrypted bits by consecutively letting each party verifiably shuffle its vector of bits (see Section 3.2), thus effectively emulating a mix-net. After exponentiating each component with a jointly created random number (as described in Section 3.1) and decrypting all components, the number of true bits is exactly the number of components not equal to 1.

5.1 Hamming Protocol

A simple function that can be expressed as an arithmetic formula fitting Equation 1 is the Hamming distance of two bitstrings. The Hamming distance is defined as the number of corresponding bits that are not equal. In other words,

$$f(b_1, b_2) = \#_{j=1}^{k}\Big(b_{1j} \oplus b_{2j}\Big).$$

An efficient 6-round protocol for computing this function can be designed based on the constructions proposed in the beginning of this section. For reasons of limited space, we just spell out the similar, but more sophisticated, protocol for computing the greater-than function in the following section.

5.2 Millionaires' Protocol

A protocol for the millionaires' problem can be obtained by reformulating the greater-than function $f(b_1, b_2) = [b_1 > b_2]$ to fit Equation 1. Let Alice's and Bob's inputs, b_1 and b_2, be k-bit numbers so that $b_i = \sum_{j=1}^{k} b_{ij}2^{j-1}$, *i.e.*, the least significant bit is b_{i1} and the most significant bit is b_{ik}. Consider the following Boolean expression for computing $b_1 > b_2$ given in [Fis01].

$$[b_1 > b_2] \iff \bigvee_{j=1}^{k}\left(b_{1j} \wedge \neg b_{2j} \wedge \bigwedge_{d=j+1}^{k}(\neg(b_{1d} \oplus b_{2d}))\right). \tag{2}$$

The outer disjunction is exclusive, *i.e.*, at most one of the k terms can be satisfied. By applying De Morgan's laws, the right expression in Implication 2 can be rewritten as

$$\bigvee_{j=1}^{k}\Bigg(\neg\underbrace{\Bigg(\neg b_{1j} \vee b_{2j} \vee \bigvee_{d=j+1}^{k}(b_{1d} \oplus b_{2d})\Bigg)}_{B_j :=}\Bigg).$$

Using the techniques proposed at the beginning of this section, the inner term B_j can be computed as follows.

$$E[B_j] = \frac{Y \cdot E[b_{2j}]}{E[b_{1j}]} \cdot \prod_{d=j+1}^{k} \left(\frac{E[b_{1d}]}{E[b_{2d}]} \right)^{2^d - 2} \quad .$$

Recall that the outer disjunction is *exclusive*, *i.e.*, counting the number of *false* B_j's will yield either 0 or 1. This implies that $b_1 > b_2$ holds if and only if $\#(B_j) = k - 1$. For this reason, the following procedure suffices: Alice sends a verifiable shuffle of all $E[B_j]$ to Bob who verifiably shuffles the resulting ciphertexts himself and sends them back to Alice. Finally, both parties raise each $E[B_j]$ to a jointly created random exponent M_j and decrypt all $(E[B_j])^{M_j}$. If any of these values equals 1, then $b_1 > b_2$, *i.e.*, Alice is richer than Bob. The detailed 6-round protocol is given in Figure 1.

In the remainder of this section, we use the millionaires' protocol as an example to analyze security and efficiency of our proposed techniques.

Proposition 1. *The millionaires' protocol is* correct *with negligible error probability and* secure *if the decisional Diffie-Hellman problem is intractable.*

Proof. (sketch)

Correctness: The protocol only fails when the random exponentiation for any outcome vector "accidently" yields a one, *i.e.*, $\sum_{h=1}^{n} m_{ij}^{+h} = 0 \mod q$ for any i and j. Due to the exponential size of $\mathbb{G}_q$ and the polynomial number of output components, the probability of this event is negligible. Error probability of the protocol is $(1 - (1 - 2^{-Q})^k) = \mathcal{O}(2^{-Q})$ where $Q = \log q$. The malleability of El Gamal encryption does not pose a problem because bidders prove that they know each plaintext using non-malleable zero-knowledge proofs.

Security: The security of El Gamal cipher as well as the applied zero-knowledge proofs can be based on the intractability of the decisional Diffie-Hellman assumption [TY98]. The security of *distributed* El Gamal cipher, in particular Pedersen's straightforward key generation [Ped91] which might result in non-uniformly distributed keys, follows from a recent argument by Gennaro et al [GJKR03]. Since encryption keys are essentially distributed by using 2-out-of-2 secret sharing, privacy can not be breached (unless Alice and Bob collude). □

We now investigate the computation and communication complexity of the millionaires' protocol. Typically, the computational cost of performing multiplications is negligible. Exponentiation and communication complexity are identical in the proposed protocol. Zero-knowledge proofs we apply in the protocol are non-interactive and have low constant overhead. Table 1 shows the communication complexity of each protocol step and also gives the complexity of the accompanying zero-knowledge proofs.

Depending on $i \in \{1, 2\}$, the directions address Alice ($i = 1$) or Bob ($i = 2$)).

Round 1: Generate public key

- Choose $x_{+i} \in \mathbb{Z}_q$ and $m_j^{+i}, r_{ij} \in \mathbb{Z}_q$ for each j at random.
- Publish $y_{\times a} = g^{x+i}$ along with a zero-knowledge proof of knowledge of $y_{\times a}$'s discrete logarithm (Section 3.2).
- Compute $y = y_{\times 1} \cdot y_{\times 2}$.

Round 2: Encrypt input

- Publish $\alpha_{ij} = Y^{b_{ij}} \cdot y^{r_{ij}}$ and $\beta_{ij} = g^{r_{ij}}$ for each j.
- Prove that $\forall j : \log_g(\beta_{ij})$ equals $\log_y(\alpha_{ij})$ or $\log_y\left(\frac{\alpha_{ij}}{Y}\right)$ (Section 3.2)

Round 3: Mix output (1/2)

- Compute for each j:

$$\gamma_j = \frac{Y \cdot \alpha_{2j}}{\alpha_{1j}} \cdot \prod_{d=j+1}^{k} \left(\frac{\alpha_{1d}}{\alpha_{2d}}\right)^{2^d-2} \quad \text{and} \quad \delta_j = \frac{\beta_{2j}}{\beta_{1j}} \cdot \prod_{d=j+1}^{k} \left(\frac{\beta_{1d}}{\beta_{2d}}\right)^{2^d-2}$$

- Alice ($i = 1$): Verifiably shuffle k vectors (γ_j, δ_j) by index j (Section 3.2).

Round 4: Mix output (2/2)

- Bob ($i = 2$): Verifiably shuffle k vectors (γ_j, δ_j) by index j (Section 3.2).

Round 5: Randomize output

- Compute and publish $\gamma_j^{\times i} = (\gamma_j)^{m_j^{+i}}$ and $\delta_j^{\times i} = (\delta_j)^{m_j^{+i}}$ for each j with a proof of logarithm equality (Section 3.2).

Round 6: Decrypt output

- Publish $\varphi_j^{\times i} = \left(\delta_{ij}^{\times 1} \cdot \delta_{ij}^{\times 2}\right)^{x+a}$ for each j with an accompanying proof of correctness (Section 3.2).
- Compute $v_j = \dfrac{\gamma_j^{\times 1} \cdot \gamma_j^{\times 2}}{\varphi_j^{\times 1} \cdot \varphi_j^{\times 2}}$ for each j. If $v_j = 1$ for any j, then b_1 is greater than b_2.

Fig. 1. Millionaires' Protocol

The total number of bits each party needs to communicate in the protocol is $(15k + 9)P + (6k + 5)Q$ where $P = \lceil \log p \rceil$ and $Q = \lceil \log q \rceil$. To achieve an appropriate level of security today, 1024 bits for p and 160 bits for q are reasonable settings. Then, in order to compare two 36-bit numbers,[10] each party only needs to send around 73 Kbytes of data.

[10] 36 bits are currently sufficient to compare the wealth of any given pair of human beings with a precision of one US dollar.

Table 1. Communication complexity (number of bits each party sends)

	Body	Zero-Knowledge Proofs
Round 1	P	$P+Q$
Round 2	$2kP$	$4k(P+Q)$
Round 3/4	$2kP$	$k(P+Q)+6P+3Q$
Round 5	$2kP$	$k(2P+Q)$
Round 6	kP	$(k+1)P+Q$
Σ	$(7k+1)P$	$(8k+8)P+(6k+5)Q$
Σ Body+ZK	$(15k+9)P+(6k+5)Q$	

$P = \lceil \log p \rceil$, $Q = \lceil \log q \rceil$

6 Conclusion

We have presented a set of primitives based on El Gamal encryption that can be used to construct efficient MPC protocols for certain low-complexity functions. Due to underlying efficient honest-verifier zero-knowledge proofs, the resulting protocols are secure against active adversaries. Security relies on the decisional Diffie-Hellman assumption. To demonstrate the applicability of the proposed techniques, we constructed protocols to compute the equality, the OR, the maximum, the Hamming and the greater-than functions. The latter requires only 6 rounds of interaction in the random oracle model while communication complexity is linear in the length of bitstrings to be compared, and error probability is exponentially small in the security parameter. To the best of our knowledge, this is the most efficient constant-round protocol for the greater-than function to date. The protocol can serve as a building block for the secure computation of more sophisticated functions such as the median [AMP04].

Future work includes the investigation of a more complete algebraic characterization of functions that can be efficiently computed using the proposed primitives. Furthermore, it might be possible to construct a sub-protocol for checking whether a vector of El Gamal ciphertexts contains an encrypted 1 which is considerably more efficient than consecutive mixing and decrypting.

Acknowledgements

Thanks to Jens Groth, Jesper Nielsen, Kobbi Nissim, and the anonymous referees for helpful comments. This material is based upon work supported by the Deutsche Forschungsgemeinschaft under grant BR 2312/1-1.

References

[ACS02] J. Algesheimer, J. Camenisch, and V. Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In *Proc. of 22th CRYPTO Conference*, volume 2442 of *LNCS*, pages 417–432. Springer, 2002.

[AMP04] G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the kth-ranked element. In *Proc. of 21st Eurocrypt Conference*, volume 3027 of *LNCS*, pages 40–55. Springer, 2004.

[BF97] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In *Proc. of 17th CRYPTO Conference*, volume 1294 of *LNCS*, pages 425–439. Springer, 1997.

[BFKR90] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Security with low communication overhead. In *Proc. of 10th CRYPTO Conference*, number 537 in LNCS, pages 62–76. Springer, 1990.

[BGN05] D. Boneh, E. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Proc. of 2nd Theory of Cryptography Conference (TCC)*, volume 3378 of *LNCS*, pages 325–341. Springer, 2005.

[BMR90] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proc. of 22nd STOC*, pages 503–513. ACM Press, 1990.

[BST01] F. Boudot, B. Schoenmakers, and J. Traoré. A fair and efficient solution to the socialist millionaires' problem. *Discrete Applied Mathematics*, 111(1-2):23–36, 2001.

[CC00] C. Cachin and J. Camenisch. Optimistic fair secure computation. In *Proc. of 20th CRYPTO Conference*, volume 1880 of *LNCS*, pages 93–111. Springer, 2000.

[CDN01] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Proc. of 18th Eurocrypt Conference*, volume 2045 of *LNCS*, pages 280–300. Springer, 2001.

[CDS94] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proc. of 14th CRYPTO Conference*, volume 893 of *LNCS*, pages 174–187. Springer, 1994.

[CGS97] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Proc. of 14th Eurocrypt Conference*, volume 1233 of *LNCS*, pages 103–118. Springer, 1997.

[CP92] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Proc. of 12th CRYPTO Conference*, volume 740 of *LNCS*, pages 3.1–3.6. Springer, 1992.

[Dam02] I. Damgård. On Σ-protocols. Lecture Notes, University of Aarhus, Department for Computer Science, 2002.

[DDO+01] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In *Proc. of 21th CRYPTO Conference*, volume 2139 of *LNCS*, pages 566–598. Springer, 2001.

[DK01] I. Damgård and M. Koprowski. Practical threshold RSA signatures without a trusted dealer. In *Proc. of 18th Eurocrypt Conference*, volume 2045 of *LNCS*, pages 152–165. Springer, 2001.

[El 85] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[Fis01] M. Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *Proceedings of the Cryptographers' Track at the 10th RSA Conference*, volume 2020 of *LNCS*, pages 457–472, 2001.

[FS87] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proc. of 12th CRYPTO Conference*, LNCS, pages 186–194. Springer, 1987.

[Gil99] N. Gilboa. Two party RSA key generation. In *Proc. of 19th CRYPTO Conference*, volume 1666 of *LNCS*, pages 116–129. Springer, 1999.

[GJKR99] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Proc. of 16th Eurocrypt Conference*, volume 1592 of *LNCS*, pages 295–310. Springer, 1999.

[GJKR03] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Applications of Pedersen's distributed key generation protocol. In *Proc. of Cryptographers' Track at the 12th RSA Conference*, volume 2612 of *LNCS*, pages 373–390. Springer, 2003.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proc. of 19th STOC*, pages 218–229. ACM Press, 1987.

[GMY04] J. Garay, P. MacKenzie, and K. Yang. Efficient and secure multi-party computation with faulty majority and complete fairness. To appear, 2004.

[Gro03] J. Groth. A verifiable secret shuffle of homomorphic encryptions. In *Proc. of 6th PKC Conference*, volume 2567 of *LNCS*, pages 145–160, 2003.

[IG03] I. Ioannidis and A. Grama. An efficient protocol for Yao's millionaires' problem. In *Proc. of 36th Hawaii International Conference on System Sciences (HICSS)*, pages 205–210. IEEE Press, 2003.

[IK00] Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proc. of 41st FOCS Symposium*, pages 294–304. IEEE Press, 2000.

[JJ00] M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In *Proc. of 6th Asiacrypt Conference*, volume 1976 of *LNCS*, pages 162–177. Springer, 2000.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *Proc. of 20th ACM STOC*, pages 20–31. ACM Press, 1988.

[KO02] K. Kurosawa and W. Ogata. Bit-slice auction circuit. In *Proc. of 7th European Symposium on Research in Computer Security (ESORICS)*, volume 2502 of *LNCS*, pages 24–38. Springer, 2002.

[Lin01] Y. Lindell. Parallel coin-tossing and constant-round secure two-party computation. In *Proc. of 21st CRYPTO Conference*, volume 2139 of *LNCS*, pages 171–189. Springer, 2001.

[LT05] H.-Y. Lin and W.-G. Tzeng. An efficient solution to the Millionaires' Problem based on homomorphic encryption. In *Proc. of 3rd International Conference on Applied Cryptography and Network Security (ACNS)*, volume 3531 of *LNCS*, pages 456–466, 2005.

[NN01] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *Proc. of 33rd STOC*, pages 590–599. ACM Press, 2001.

[NPS99] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proc. of 1st ACM Conference on E-Commerce*, pages 129–139. ACM Press, 1999.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. of 16th Eurocrypt Conference*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.

[PBDL04] K. Peng, C. Boyd, E. Dawson, and B. Lee. An efficient and verifiable solution to the millionaire problem. In *Proc. of 7th International Conference on Information Security and Cryptology (ICISC)*, volume 3506 of *LNCS*, pages 51–66. Springer, 2004.

[Ped91] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Proc. of 11th CRYPTO Conference*, volume 576 of *LNCS*, pages 129–140. Springer, 1991.

[Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[Sch96] B. Schneier. *Applied Cryptography.* John Wiley and Sons, Inc., 2nd edition, 1996.

[ST04] B. Schoenmakers and P. Tuyls. Practical two-party computation based on the conditional gate. In *Proc. of 10th Asiacrypt Conference*, number 3329 in LNCS, pages 119–136. Springer, 2004.

[TY98] Y. Tsiounis and M. Yung. On the security of ElGamal-based encryption. In *Proc. of 1st International Workshop on Practice and Theory in Public Key Cryptography (PKC)*, volume 1431 of *LNCS*, pages 117–134. Springer, 1998.

[Yao82] A. C. Yao. Protocols for secure computation. In *Proc. of 23th FOCS Symposium*, pages 160–164. IEEE Computer Society Press, 1982.

[Yao86] A. C. Yao. How to generate and exchange secrets. In *Proc. of 27th FOCS Symposium*, pages 162–167. IEEE Computer Society Press, 1986.

An Enhanced Estimation Algorithm for Reconstructing Fingerprint Strip Image

Woong-Sik Kim[1,2], Weon-Hee Yoo[1], Jang-Hyun Park[2], and Bok-Ki Kim[3]

[1] Department of Computer Science & Information Engineering Inha University, Yonghyun-dong, Nam-Ku, Incheon, Korea
wskim1@software.or.kr, whyoo@inha.ac.kr
[2] 11F KIPA Bldg., Garakbon-dong, Songpa-ku, Seoul, Korea
jhpark@software.or.kr
[3] Department of Electronics Engineering Kwangwoon Unversity, Wolgye-dong, Nowon-Gu, Seoul, Korea
bkkim@daisy.kw.ac.kr

Abstract. Fingerprint Recognition Technologies are becoming more popular to be used as a representative of biometrics nowadays. As all electronics devices tends to be smaller and smaller, people pays more attention to "linear sensor" method. The most important factor for securing the reliable Fingerprint Recognition Technology is pursuing the enhancement of the captured fingerprint image from sensors. This paper presents an algorithm that uses Image Estimation and Reconstruction Method in order to embody precisely the image captured from Linear Sensors. The proposed algorithm is verified theoretically and experimentally that the image can be perfectly reconstructed from the fingerprint on the linear sensor even though the finger position is displaced vertically, horizontally, and/or rotationally. Also this system demonstrates much higher recognition and authentication rates than any other existing systems.

1 Introduction

Fingerprint Linear Sensor captures a moving object on the sensor, generates consecutive image stripes and reconstructs the image by assorting and combining the stripes into one. Fax machine uses the same logic when scanning documents, in which the image of an object is reconstructed by combining the generated stripes from the moving object. Scanners or copy machines however obtain the image from the moving laser or sensor[4,5].

A clear difference between the logics that are used in the scanner and those in the fingerprint linear sensor is that the moving speed of the object in the scanner is uniform, but the speed of the finger is not constant because the force applied to the linear sensor is not uniform. These factors such as pressure, speed, and direction make the scanned image vertically, horizontally, and even slightly rotationally displaced. Therefore to acquire the image of the real fingers in the best quality, an algorithm for image estimating and reconstructing the image is necessary[3].

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 48 – 55, 2006.
© Springer-Verlag Berlin Heidelberg 2006

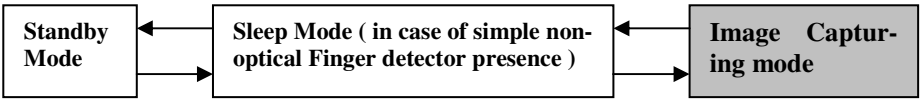
The algorithm only performs an vertical speed estimation and reconstructs the image since the moving speed of the finger is not uniform[7]. However distortion of the image can occurs in acquiring fingerprint image because the algorithm does not perform any correction to the possible horizontal and rotational displacements. Also, the algorithm limits the number of the stripes to four rows, and can not be applied to the images obtained from the Thomson-csf sensor.

2 Theory

An image capture optical sensor needs to have light source on. External device (PC or matching board) may make a solution to switch to the standby mode. In standby mode the sensor does not turn on the light and do not respond any users' actions. In image capturing mode the sensor turn on the light and try to capture the image. It might be useful to have one more (sleep) mode in the middle of standby and image capturing. The reason to do that is to avoid the light flushing before user places the finger onto the sensor and providing rough (non-optical) finger detection to switch sensor to the image-capturing mode to start grabbing the image. Estimation algorithm deals with the stripes capturing and used only in image capturing mode. Estimation algorithm estimate stripes shifts values to provide "Reconstructor" with the data to reconstruct the entire image.

Once decision is made to try capturing the fingerprint image, estimation algorithm will perform the following steps: wait for the image presence using "no image" test, then perform actual stripe-by-stripe image capturing and finally stop the image capturing (this is also based on "no image" test)[3].

< Chat 1 > Enroll of Fingerprint



Actual stripe-by-stripe image capturing is performed with adaptive time delay changes and with "no overlapping" tests. Coordinate shift values (dy1, dy2) of one stripe compare with another are calculated using minimum determination of difference function on each segment with parabolic three-points approximation and linear mean squire approximation with weight coefficients across the segments. Scale value is used to rescale dy1and dy2 to avoid of float point values data transfers. Amount of the segments is equal to the stripe's height (later: <m>). The reason is that for the maximum rotation angle, which we are able to capture (zero on one side and <m> on another), we will have the real minimum value line located within one discrete on each segment, so three point parabolic approximation is applicable nearby the minimum of the difference function.

Estimator acquires stripes form the Model and estimates the relative position of each next stripe compare with the previous one. The full set of the parameters include vertical, horizontal shifts and angle, also this parameters might be expressed in some other form. The value of this parameters are different, the most valuable is vertical shift (in sweep direction), then angle and then horizontal shift.

The estimation made for the ordinary set of the image parameters and usual kinds of finger sweeps shows that the average angle between two consecutive stripes is small, it's about 0.006 radians and that is the same as about 3-pixel vertical shift on a length of 512 pixels. We need to estimate that as soon as the sum of those small angles may lead to some significant value on a full sweep length[5].

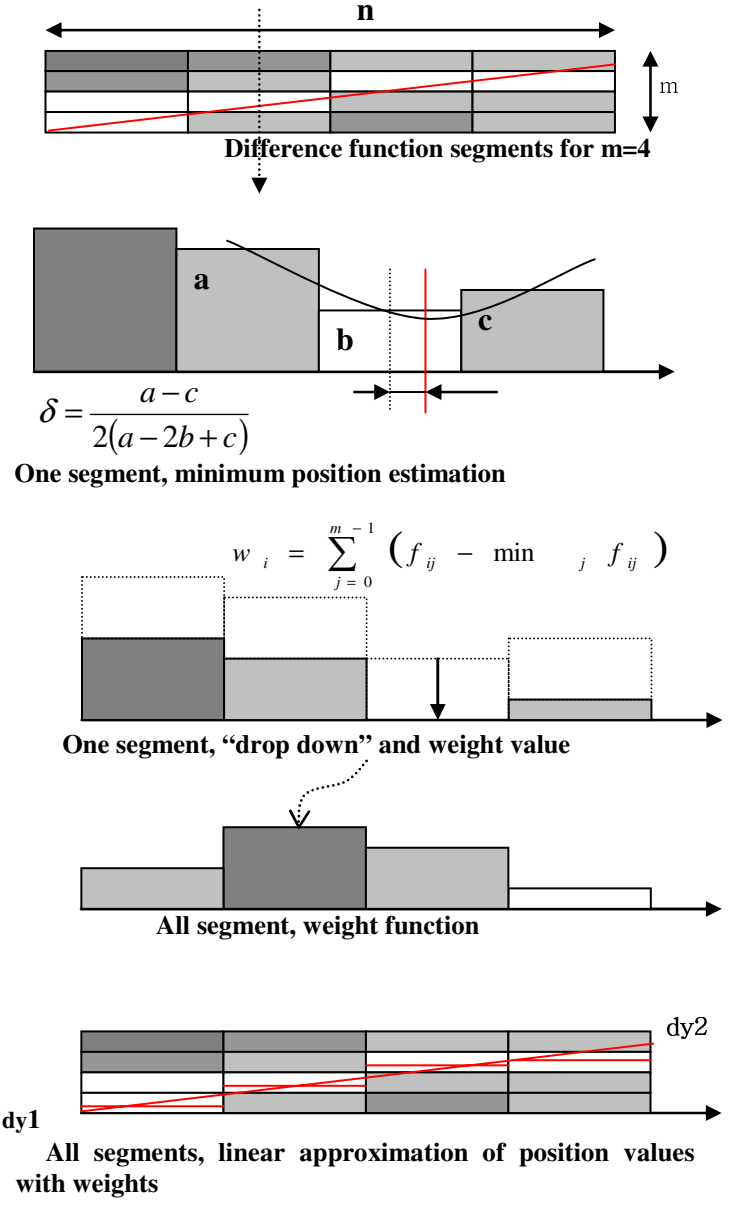


Fig. 1. Represent all of the algorithm's processes (1)

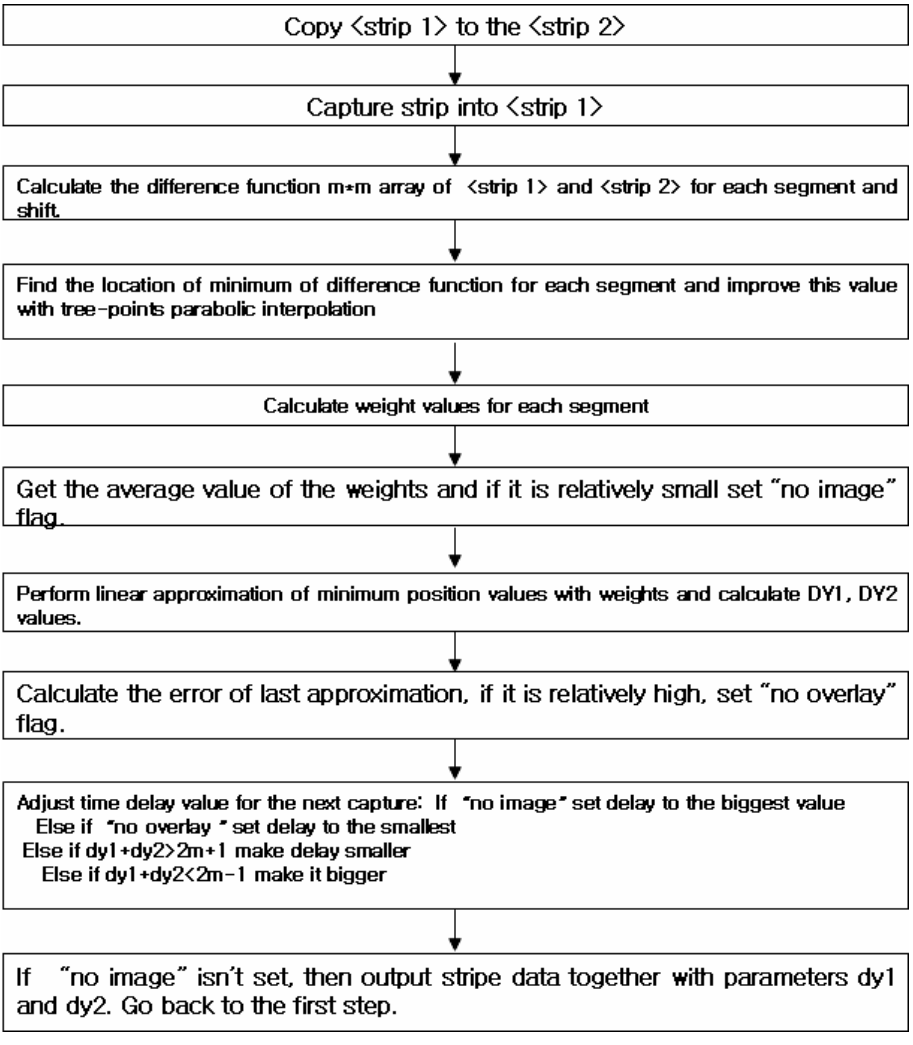


Fig. 2. Represent all of the algorithm's processes(2)

We have no precise estimation of horizontal shift value, just some rough analysis of the sweep process as the physical process (mechanics). To sweep the finger needs to use some force and this force might be irregular which will also lead to sweep speed irregularities both in absolute value and the angle (in certain limits). The direction of the force is about the same as the direction of the finger (just different sign). The closer those directions will be the less absolute force user will need to use for sweeping. That means that horizontal shifts might happen only if the user makes it intentionally.

Hence considering vertical translation and a small rotational angle, the movement of the finger can be estimated. When the finger moves rotation maximum 45 degrees,

2N stripes (when half of stripes are overlapped each other) are necessary to reconstruct the whole image. The angle of each strip[11].

$$\frac{n\Omega}{2N} \sim \Delta\alpha \quad (1)$$

where $\Omega = \pi/4$ (45°) , n is the length of the strip, and N is the length of the whole image. For example, when N=513 pixel and n=8 pixel, the rotational angle of the strip is approximately 0.006 radian.

Once decision is made to try capturing the fingerprint image, estimation algorithm will perform the following steps: wait for the image presence using "no image" test, then perform actual stripe-by-stripe image capturing and finally to stop the image capturing (this is also based on "No image" test). Actual stripe-by-stripe image capturing is performed with adaptive time delay changes and with "no overlapping" tests. Coordinate shift values (dy1, dy2) of one stripe compare with another are calculated using minimum determination of difference function on each segment with parabolic three-points approximation and linear mean squire approximation with weight coefficients across the segments. Scale value is used to rescale dy1 and dy2 to avoid of float point values data transfers. Amount of the segments is equal to the stripe's height (later: <m>). The reason is that for the maximum rotation angle, which we are able to capture (zero on one side and <m> on another), we will have the real minimum value line located within one discrete on each segment, so three point parabolic approximation is applicable nearby the minimum of the difference function.

3 Experiments

we proposed the technology which we can reconstruct the perfect image no matter the input image with various directions, vertical, horizontal and rotating. With this proposed technologies, we can reconstruct the image from estimation and compensation the scanned image through Fingerprint Sensor. And the recognition rate can be highly improved.

Term Model is used to indicate that this application simulates the linear sensor image capture process and assume some variations in sweeping object behavior, like speed or rotations. Also Model application panel provides the visualization of the process and provide the user with the controls to check the other applications for the variations of speed and rotations and even for some "sideway jitter".

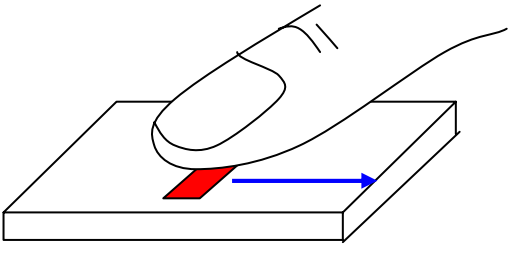


Fig. 3. Object moving of Linear Sensor

Linear sensor and strip comparison principle assume partial stripes overlapping, so in some extreme cases, when there is not such overlapping or the area of it is too small, or the data in overlapping area is "flat", there is no way to get the correct coordinate values and to place this stripe into the proper place of entire image. Partially to overcome that "Estimator" algorithm uses "No overlapping" check and proper reset after such an event. Also big angles in most of the cases just can not be estimated because of small stripe overlapping at this case.

Linear Type Sensor requires the reconstructing technologies with estimating the speed of finer object on the sensor (see Fig. 4). We have tested the algorithm with 'Reconstructor' which is generating the one combined image and 'Estimator' which is estimate the speed of constant outputs from sensor.

Fig. 3 shows the reconstructed image using stripes acquired from linear sensor by using vertical image estimation only, which is the image obtained using existing algorithms.

Fig. 4 shows the reconstructed image without any distortion using the proposed method that estimates vertically, horizontally, and rotationally.

Fig. 5 shows the reconstructed image using stripes acquired from linear sensor by using vertical image estimation only, which is the image obtained using existing algorithms.

Fig. 6 shows the reconstructed image without any distortion using the proposed method that estimates vertically, horizontally, and rotationally.

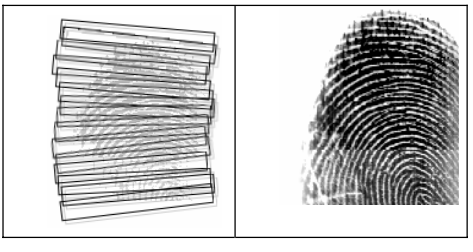


Fig. 4. Reconstructed fingerprint image through proposed algorithm

Fig. 5. **Fig. 6.**

The below Fig. 7 is example for captured fingerprint image on 515dpi through our algorithm.

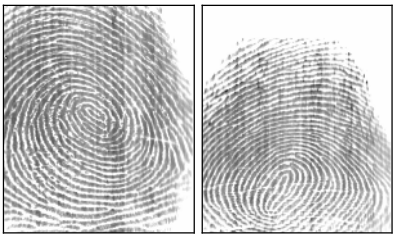


Fig. 7. Reconstructed fingerprint image through proposed algorithm

4 Conclusion

Most of linear sensors have inferior image restoration than the general optical sensor due to the skill of user and negligence. However, with the price and slim shape of the sensors, it has been applied in several fields such as mobile phone and others, and if the ensuing research could improve the image restoration for even more, there would be slim shape and the economic benefit to contribute to the development of bio-recognition industry[11].

On a final stage tests must be performed to make a conclusion about FAR/FRR values. Those tests might be performed on a large fingerprint database. Optimizing, speed and memory estimation needs to be performed for the hardware implementation (matching board). Depending on the desired fingerprint database size different algorithms might be used to achieve the balance between FAR/FRR, speed and memory requirements.

Acknowlegement

This work was supported by INHA UNIVERSITY Research Grant.

References

1. Dario Mario and David Maltoni, “Direct Gray-Scale Minutiae Detection In Fingerprints,” IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 19, No. 1, pp. 27-39, Jan., 1997.
2. Lin Hong, Yifei Wan and Anil Jain, “Fingerprint Image Enhancement : Algorithm and Performance Evaluation,” IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 20, No. 8, pp. 777-789, Aug., 1998.
3. A.K. Jain, L. Hong and R. Bolle, “On-Line Fingerprint Verification”, IEEE Trans., Pattern Analysis and Machine Intelligence, Vol. 19, No. 4, pp. 302-313, April., 1997.
4. A.K. Jain, S. Prabhakar, L. Hong and S. Pankanti, “Filterbank-based fingerprint matching,” IEEE Trans. on Image Processing, Vol. 9, No. 5, pp. 846-859, May 2000.
5. K. Nalini and A.K. Jain, “A Real-Time Matching System for Large Fingerprint Database,” IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 18, No. 18, pp. 799-813, Aug., 1996.

6. Xudong Jiang and Wei-Yun Yau, “Fingerprint Minutiae Matching Based on the Local and Global Structures,” International Conference on Pattern Recognition(ICPR’00), Vol. 2, No. 18, pp. 1038-1042, Sep., 2000.
7. T. MAEDA, M. Matsushida and K. Sasakawa, “Identification Algorithm Using a Matching Score Matrix,” IEICE Trans. INF. & SYST., Vol. E-84-D, No. 7, pp. 819-824, July 2001.
8. Xudong Luo, Jie Tian and Yan Wu, “A Minutia Matching algorithm in Fingerprint Verification,” International Conference on Pattern Recognition(ICPR’00), Vol. 4, No. 18, pp. 4833-4841, Sep., 2000.
9. J. D. Stosz and L. A. Alyea, "Automated Systems for Fingerprint Authentication Using Pores and Ridge Structure," Proceedings of SPIE, Automatic Systems for the Identification and Inspection of Humans (SPIE Vol. 2277), San Diego, 1994, pp. 210- 223.
10. W.-S.Kim, W.-H.Yoo, “A Layered Fingerprint Method”, LNCS3546, pp.702-709, 2005.
11. W.-S.Kim, W.-H.Yoo, “A Method For Acquiring Fingerprint by Linear Sensor”, LNAI 3684, pp. 410-416, 2005.

Trust Management for Resilient Wireless Sensor Networks

Junbeom Hur, Younho Lee, Seong-Min Hong, and Hyunsoo Yoon

Korea Advanced Institute of Science and Technology(KAIST)
{jbhur, yhlee, smhong, hyoon}@camars.kaist.ac.kr

Abstract. Utilities of wireless sensor networks are standing out in bold relief in various fields such as home environmental, industrial, and military applications. Compared with the vivid applications of the sensor networks, however, the security and privacy issues of the sensor networks are still in their infancy because unique features of the sensor networks make it difficult to adopt conventional security policies. Especially, false reports are critical threats because they can drain out the finite amount of energy resources in a battery-powered sensor networks; thus, a novel trust management scheme is necessary to make resilient wireless sensor networks. Cryptographic authentication mechanisms and key management schemes cannot suggest solutions for the real root of the problem. In this paper, we propose a trust management scheme which can identify trustworthiness of sensor nodes and suggest a defensible approach against insider attacks beyond conventional cryptographic approaches.

1 Introduction

Wireless sensor networks are emerging technologies that have a variety of potential applications such as battlefield surveillance and emergency response [1]. A major feature of these systems is that sensor nodes in networks assist each other by passing data, in-network process and control packets from one node to another. It is often termed an infrastructure-less, self-organized, or spontaneous network [2].

Research on sensor networks generally assumes a trusted environment, but in many sensor network applications, the network tends to be deployed in environments where an adversary may be motivated to disrupt the function of the network. An adversary may be able to position several intruder nodes within the network or compromise sensor nodes in the network in order to use them to transmit false messages [4]. So, to differentiate false data from legal ones is an essential process for a normal and effective function of the sensor network, because false reports can drain out the finite amount of energy resources in a battery-powered sensor networks and even a small amount of compromised nodes can influence the whole sensor networks critically [18].

Several researches have proposed authentication mechanisms for sensor networks to prevent false data injection by an outsider attacker [15], [16], [17]. Their basic approaches for security are to use Message Authentication Codes

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 56–68, 2006.
© Springer-Verlag Berlin Heidelberg 2006

(MACs) and probabilistic key pre-distribution schemes such as [12], [13]. These approaches prevent naive impersonation of a sensor node, however, they cannot prevent the injection of false data from malicious or compromised insider nodes which have already been authenticated as legal ones in the networks. Once authenticated as a legitimate node, broadcasting data from that node are also adjudicated to be trusted in the networks without any question. Therefore, a smart trust management scheme is necessary for wireless sensor networks to identify trustworthiness of sensor nodes.

Here, we propose a trust management scheme for resilient wireless sensor networks, which helps the networks to operate robustly although some nodes or data would be compromised. To be more specific, the best we can hope for in the presence of insider adversaries is graceful degradation of the wireless sensor networks. A direction for resilience of our scheme is to gather multiple and redundant sensed data, and crosscheck them for consistency. For a reasonable crosschecking, each sensor node compares sensed data of its neighbor nodes with the expected sensing results within the consistently-acceptable sensing range and estimates its neighbor nodes' trust values based on the result of that crosschecking.

The rest of the paper is organized as follows. Section 2 describes goals and assumptions of the trust management scheme. Section 3 details an overall process of the trust management scheme. Section 4 analyzes the performance evaluation. Section 5 shows the simulation results, and Section 6 remarks conclusion of the paper.

2 Goals and Assumptions

2.1 Threat Model

Sensor nodes are deployed in open areas and have many opportunities to interact closely with anonymous adversaries, so that deceitful data from them can be easily accepted as legal data in the networks. In addition, because each sensor node is confronting the added risk of physical attacks such as node capture, some private information for secure communication in the networks could be snatched by active attackers.

2.2 Goals

We focus on making resilient wireless sensor networks which work normally even though some sensor nodes might be compromised. For the purpose of making resilient networks, the adversaries should have only a limited influence on the result of the data aggregation. In other words, if x denotes the aggregated result of sensing data in the absence of an adversary and y denotes the result after attackers intervene, then we wish $|x - y|$ to be bounded by some small value. To make resilient wireless sensor networks, we direct our approaches to evaluate trustworthiness of sensor nodes accumulatively. As a result of the evaluated trust values of sensor nodes, we can filter out inconsistently-forged data from the malicious or compromised nodes in the sensor networks.

2.3 Assumptions

We have some assumptions in our trust management scheme as follows: (1) The network consists of a set of sensor nodes of unknown location and a set of specially equipped nodes, anchor nodes, with known location and orientation. Anchor nodes are trusted during localization step. (2) Sensor nodes are deployed densely enough to be able to sense some identical events redundantly with their neighbor nodes. (3) Malicious nodes do not collude with each other. That is, they do not manipulate or intentionally increase trust values for each other, but just try to inject spurious data into the networks.

3 Trust Management Scheme

We describe our trust management scheme which consists of four steps: First, we divide sensing areas into some logical grids and assign a unique identification to each grid (Section 3.1). Second, sensor nodes estimates their deployed location and a corresponding grid (Section 3.2). Third, each node evaluates trustworthiness of its neighbor nodes by crosschecking the neighbor nodes' redundant sensing data with its own result. Inconsistent data from malicious or compromised nodes can be detected in this step (Section 3.3). Fourth, special nodes, aggregators, aggregate sensing data from their grids and transmit the computed results to the destination node, or sink. Inconsistent data from malicious nodes can be excluded in this step (Section 3.4).

3.1 Step 1: Grid Definition

In this step, consider first sensing areas in which sensor nodes will be deployed and ready for some events. Then, we divide the sensing areas into several logical grids in proportion to the sensing range s of a sensor device so that one sensor device's sensing range can cover a grid entirely it belongs to regardless of its deployed location.

Although there can be so many choices, in our model, we intend to use as many redundant sensing data from multiple sensor nodes as possible to identify inconsistent data among them. So, for the higher correctness of the crosscheck, we set one grid size to $\frac{r}{\sqrt{2}} \times \frac{r}{\sqrt{2}}$ as in Figure 1.

After dividing sensing areas into some logical grids, we assign a unique identification to each grid.

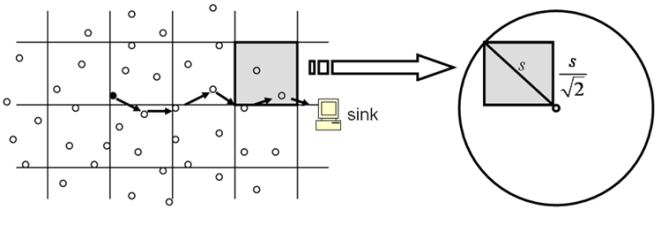


Fig. 1. Grid Definition

3.2 Step 2: Localization

In this localization step, each sensor node determines its own location and a corresponding grid in which it is deployed by adapting a localization scheme such as triangulation or trilateration using more than three anchor nodes.

After estimating its own deployed location, each sensor node broadcasts a HELLO message including location information to its neighbor nodes. Then, sensor nodes who received the HELLO messages can verify the location claims using a location verification protocol such as [6]. Although several secure localization schemes such as [20], [21], [22] have been proposed, in this paper, we would not propose an additional novel localization scheme for wireless sensor networks because it is out of our research focus.

3.3 Step 3: Trust Evaluation

In this step, we propose a trust evaluation process. The trust defined in our model is the confidence of a node on another node. The trust value means the level of trustworthiness of a node, which is computed based upon several trust evaluation factors. In our scheme, sensor nodes do not compute all the other nodes' trust values in the networks, but compute only their neighbor nodes' trust values accumulatively.

Trust Evaluation Factor. Each sensor node has k trust evaluation matrices which stores the trust evaluation factors for its k neighbor nodes. The trust evaluation matrix consists of several trust evaluation factors as follows:

1. Identification: This factor contains an unique identification of a node.
   - $ID_i =< GridID, Position_i >$, where $1 \leq i \leq k$
2. Distance: This factor contains distance information between two nodes. x_i means x coordinate and y_i means y coordinate of node i.
   - $D_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, where $0 \leq i, j \leq k$ and $i \neq j$
3. Sensing communication: This factor contains communication ratio information. This factor represents the level of selfishness and normality of a node.
   - S_i: sensing communication value of node i, where $1 \leq i \leq k$
   - ss_i: sensing success count of node i
   - sf_i: sensing failure count of node i
4. Sensing result: This factor represents sensing result information for detected events. This factor consists of sensing data and sensing time for the events.
   - $R_i =< sr_i, st_i >$: sensing result value of node i, where $1 \leq i \leq k$
   - sr_i: sensing data of node i
   - st_i: sensing time of node i
5. Consistency: This factor represents consistency level of a node. Based on this factor, each node can identify malicious nodes in the networks.
   - C_i: consistency value of node i, where $1 \leq i \leq k$
   - cs_i: consistent sensing count of node i
   - is_i: inconsistent sensing count of node i

6. Battery: This factor represents remained lifetime of a sensor node.
   - B_i: battery value of node i, where $1 \leq i \leq k$
7. Trust value: This factor represents a total trustworthiness of a node, which is evaluated based on the other trust evaluation factors.
   - T_i: trust value of node i, where $1 \leq i \leq k$

Next, we propose a consistency check mechanism, trust quantification method, and trust computation method.

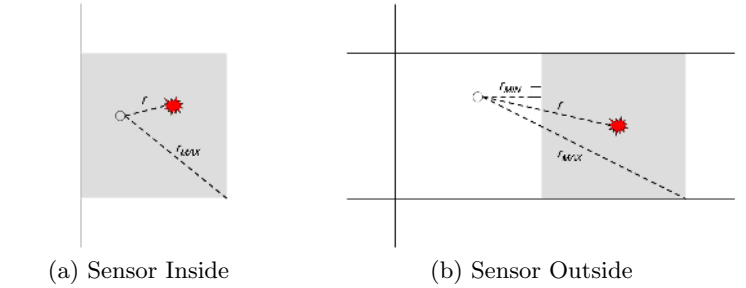
Consistency Check. In brief, to check consistency of neighbor nodes' broadcasted data, sensor nodes compute the acceptable region (AR) of an event first, and put boundary to the legally acceptable range of neighbor nodes' sensing data based on the AR and their own sensed data. When node j checks the consistency of its neighbor node i's sensing results, if the results are out of legally acceptable bound of node j, node j estimates the results to be inconsistent or deceitful data. Such an estimation for its neighbor, node i, affects the value of the consistency factor, C_i, in the trust evaluation matrix of node j.

To compute the legally acceptable range of the sensed data, sensor nodes should have a knowledge of the distance between an event and sensor nodes, and the distance between sensor nodes. Because each sensor node knows its neighbor nodes' location, it is enough to know the distance information between sensor nodes and an event to check the consistency of the neighbor nodes' sensed data. As even an event-sensed sensor node can hardly identify the location of the event exactly, however, we can just expect to limit the approximately possible AR of the event.

We can considerate two cases of event sensing of sensor nodes: inside the event-happened grid, and outside the event-happened grid as we described in Figure 2. When we define r as the distance between an event and a sensor node, r follows the condition:

$$r_{MIN} \leq r \leq r_{MAX},$$

where r_{MIN} and r_{MAX} are possible minimum and maximum range of r, respectively. To be more specific, r_{MIN} means the shortest distance to the AR of the event, and r_{MAX} means the longest distance to the AR of the event.



(a) Sensor Inside the Event-happened Grid

(b) Sensor Outside the Event-happened Grid

Fig. 2. Two Cases of Event Sensing

Next, we define a reverse sense function, RSF, which outputs the expected sensing value of a node based on the distance between a node and an event, the distance between sensor nodes, and a sensing result of a node. In our model, every sensor node knows this reverse sense function and makes use of it to check legality of its neighbor nodes' sensed data. Proposed reverse sense function is the following:

$$sr^*{}_j = RSF(sr_i, r, D_{i,j}), \tag{1}$$

where $sr^*{}_j$ is the expected sensing result of node j, which node i computed based on its own sensing result, sr_i. The $sr^*{}_j$ relies on the deployed location of sensor nodes.

When a sensor node senses an event, it broadcasts its identification and sensing data, $< ID, R >$, to its neighbor nodes. If a node receives information of the sensing data and sensing time from its neighbor nodes, it can check whether the received data from neighbor nodes can be acceptable as consistent data or unacceptable as inconsistent data based on its own sensing data for the same event.

We can consider two main cases in sensing environment from a local point of view. First case is that two neighbor nodes, n_i and n_j, succeed in sensing a same event. In this case, from a n_i's point of view, n_i increases sensing success count value for n_j by 1, that is $ss_j = ss_j + 1$, and checks consistency of the n_j's sensing data and assigns the checking result to its corresponding trust evaluation factor for n_j. If n_i is closer to the event than n_j, one of the possible AR of the events could be described pictorially like a Figure 3. In Figure 3, the gray region means the AR of the event. Of course, the AR of the events can be described variously based on the deployed location of the sensor nodes. If we define RSF_{MIN} and RSF_{MAX} as minimum and maximum RSF output of a node, respectively, the consistency check processes of n_i for n_j's data can be defined as follows:

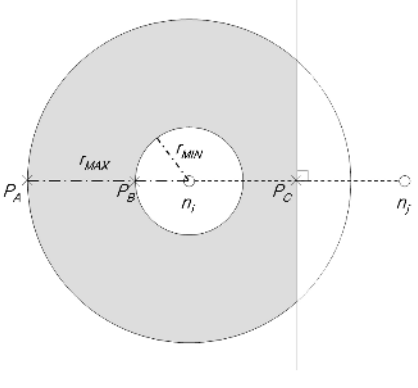


Fig. 3. Acceptable Region of an Event

$$\begin{cases} cs_j = cs_j + 1 & \text{if } RSF_{MIN}(sr_i, r, D_{i,j}) \le sr_j \text{ and} \\ & \quad sr_j \le RSF_{MAX}(sr_i, r, D_{i,j}), \\ ic_j = ic_j + 1 & \text{otherwise,} \end{cases}$$

where $st_i \le st_j$ and $r_{MIN} \le r \le r_{MAX}$.

On the other hand, when the n_j checks n_i's data for consistency in the same environment, the legally acceptable boundary of $sr^*{}_i$ would be $[RSF_{MIN}(sr_j, r, D_{i,j}), RSF_{MAX}(sr_i, r, D_{i,j})]$. For example, in the temperature sensing environment, as the receiving radiation energy is proportional to $\frac{1}{r^2}$, $RSF_{MIN}(sr_i, r, D_{i,j})$ in the above process would output the expected $sr^*{}_j$ based on the assumption that the position of the event is P_A in Figure 3. $RSF_{MAX}(sr_i, r, D_{i,j})$ would output the expected $sr^*{}_j$ based on the assumption that the position of the event is P_C. Likewise, the $RSF_{MIN}(sr_j, r, D_{i,j})$ and $RSF_{MAX}(sr_j, r, D_{i,j})$ would output the expected $sr^*{}_i$ based on the assumption that the position of the event is P_C and P_B, respectively.

Second case is that two neighbor nodes fail to sense a same event. In this case, on equal terms, n_i and n_j increase sensing failure count value for each other by 1, that is $sf_j = sf_j + 1$ and $sf_i = sf_i + 1$.

As we described a RSF in equation (1), we proposed the RSF as an abstract function. It is because the reverse sensing function may have to be dynamic for the applications it is adopted. Of course, parameters of the function could be changed dynamically according to its application.

Trust Quantification. Trust quantification process is to transform individually discrete values of trust evaluation factors into continuous values from -1 to +1. -1 and +1 mean complete distrust and complete trust, respectively. As a node communicates and revalues trust factor values for their neighbor nodes continuously, trust quantification process is imperative to impartial comparison among each node's trust values. Trust quantification processes for each trust evaluation factor are as follows:

1. Consistency value

$$C_i = \frac{cs_i - is_i}{cs_i + is_i}, \quad \text{where } -1 \le C_i \le 1. \tag{2}$$

2. Sensing communication value

$$S_i = \frac{ss_i - sf_i}{ss_i + sf_i}, \quad \text{where } -1 \le S_i \le 1. \tag{3}$$

3. Battery value

$$B_i : -1 \le B_i \le 1,$$

where each sensor node is supposed to broadcast quantification value of its own B_i.

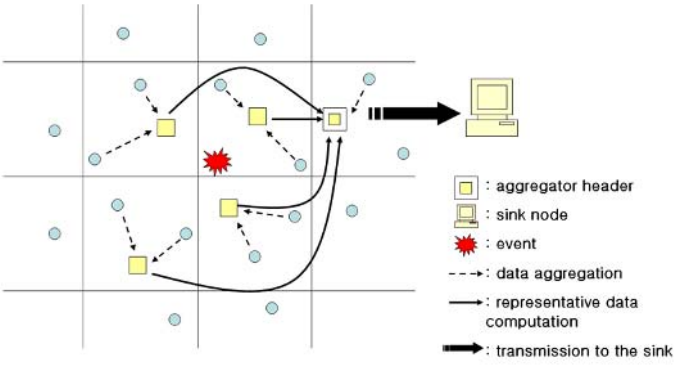


Fig. 4. Data Aggregation Process

Trust Computation. Trust computation involves an assignment of weights to the trust factors which were evaluated and quantified in trust quantification step. We define W_i as a weight which represents importance of a particular factor from 0, unimportant, to +1, most important. The weight is dynamic and dependent on the application.

When $B_i \neq -1$, trust value for node i is computed by the following equation:

$$T_i = \frac{W_1 C_i + W_2 S_i + W_3 B_i}{\sum_{i=1}^{3} W_i}, \tag{4}$$

where $0 \leq W_i \leq 1$ and W_is are not all zero. In case of $B_i = -1$, we just assign -1 to T_i and exclude the node from the networks because it totally cannot work in the networks.

3.4 Step 4: Data Aggregation

In this step, we propose a data aggregation scheme. Data aggregation is an essential process in wireless sensor networks to eliminate redundancy of sensing data, to minimize communication overhead, and to save energy. In addition to energy benefits, aggregation can help the networks to reduce the effects of error in sensor readings [19]. This aggregation scheme is best-suited to systems where there is plenty of redundancy in the data, so that the systems can cross-check sensor readings for consistency.

To aggregate data, sensor nodes elect one node as an aggregator per each grid, which has the highest trust value among all the nodes in an identical grid by the majority of vote. Then, the aggregator obtains sensing data from the other member nodes in its grid and aggregates them to a representative value in consideration of the trust values of member nodes by this equation:

$$SR_{GridID} = \frac{\sum_{i=1}^{m} (T_i + 1) sr_i}{\sum_{i=1}^{m} (T_i + 1)}, \tag{5}$$

where m is the number of nodes in a grid including the aggregator itself and T_is are not all -1.

After that, an aggregator header, node h, one of the aggregators which is the nearest aggregator among them to the sink node on the routing path as in Figure 4, calculates the median Med_h among the representative values from each grid and sends it to the sink node in the form of $< Med_h, (ID_a, ID_b, ..., ID_u) >$, where u is the number of participant grids.

4 Analysis

We analyze the resilience of our trust-based aggregation scheme compared with conventional aggregation schemes, especially median and average, under the k-node attack inspired from [18]. A k-node attack is an algorithm that try to forge k of the sensed data into false values.

When the sensor readings come from a Gaussian distribution $\mathcal{N}(\theta, \sigma^2)$, and n is the number of sensor nodes, Table 1 shows aggregation schemes, their mean square error (MSE) term in the absence of attacks, their resilience against k-node attack, and their breakdown point. The resilience factor means the deviation value which can be skewed by k compromised nodes, and the breakdown point means the fraction of nodes that can be compromised before security breaks down. The resilience and breakdown point of average and median were analyzed in [18].

Note that the breakdown point of the trust-based aggregation scheme ranges from $\epsilon^* = 1/2m$ to $\epsilon^* = (2m-1)/2m$ based on the estimated trust values of adversaries in the network, where m is the average number of nodes per grid. In the case of that the estimated trust values of adversaries is 1, that is to say that the system cannot evaluate trustworthiness of the nodes at all, the security of the trust-based scheme is worse than the median as long as $m > 1$. On the other hand, in the case of that the estimated trust values of adversaries is -1, that is the system can perfectly distinguish adversaries from normal nodes all the time, the security of the trust-based aggregation scheme is much better than the median as long as $m > 1$.

Table 1. Resilience of the Aggregation Schemes

Aggregation (f)	MSE(f)	Resilience	Breakdown point (ϵ^*)	Trust values of adversaries
average	$\frac{\sigma^2}{n}$	∞	0	
median	$\frac{\pi}{2} \cdot \frac{\sigma^2}{n}$	if $k < \frac{n}{2} : \frac{\pi}{2} + \frac{k^2}{2\pi}$ if $k > \frac{n}{2} : \infty$	1/2	
trust-based median	$\frac{m\pi}{2} \cdot \frac{\sigma^2}{n}$	if $k < \frac{n}{2m} : \frac{m\pi}{2} + \frac{mk^2}{2\pi}$ if $k > \frac{n}{2m} : \infty$	$1/2m$	1
		if $k < \frac{(2m-1)n}{2m} : \frac{m\pi}{2} + \frac{mk^2}{2\pi}$ if $k > \frac{(2m-1)n}{2m} : \infty$	$(2m-1)/2m$	-1

5 Simulation

We implement a temperature sensing system and simulate our trust evaluation scheme in C. The environments of the simulation system are as follows: 300 sensor nodes are uniformly distributed at the sensing area whose size is $500m \times 500m$. Sensing range of a sensor device is $70m$ and a grid size is $50m \times 50m$. In this simulation, we assume that all sensor nodes have a same amount of battery power and participate in communication positively regardless of their roles; thus, $W_2 = W_3 = 0$ in the equation (4). So, we consider only a consistency evaluation factor. To make a practical simulation, temperature sensing data in our simulation are modeled by Stefan-Boltzman Law whose property is that the receiving radiation energy is proportional to $\frac{1}{r^2}$, where r is the distance between two objects.

In this simulation, total number of sensor nodes who sensed the event is 39 and attackers notify their neighbor nodes of sensed temperature values 10 times as high as the original sensed one. Adversaries are randomly chosen among the 39 participant nodes, and the range of k is from 0 to total 39 under k-node attack. The adversaries' trust values range from -1 to 1, especially we choose three trust values -1, 0, and 1 for simulation.

The simulation result under k-node attack is shown in Figure 5. As we analyzed the resilience of the aggregation functions, average is not a robust estimator which cannot be computed meaningfully in the presence of a malicious sensor nodes. In this simulation environment, $n = 39$ and $m = 3$, so approximate breakdown numbers of k of each aggregation scheme can be derived from the analysis in Table 1: In the case of average, the breakdown number of k is 0. In the case of median, the breakdown number of k is 20. In the case of trust-based median, the breakdown number of k is 7 when the adversaries' trust values are 1, and 33 when the adversaries' trust values are -1. The Figure 5 shows that the analysis matches well with the simulation results.

Next simulation is to measure the resilience of the proposed trust-based aggregation scheme under k-node attack as the trust values of the nodes are evaluated

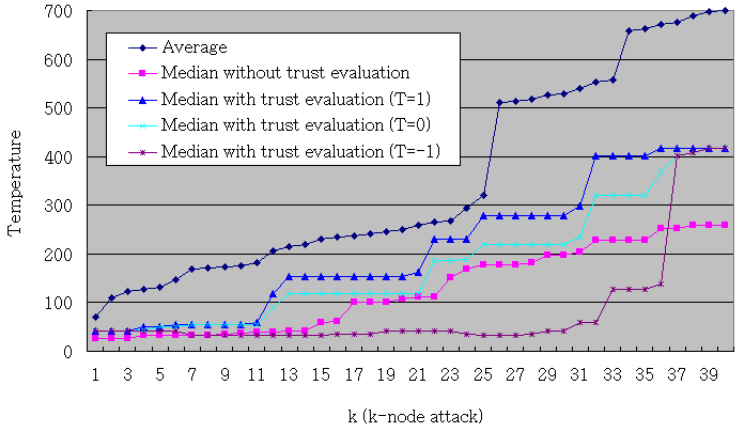
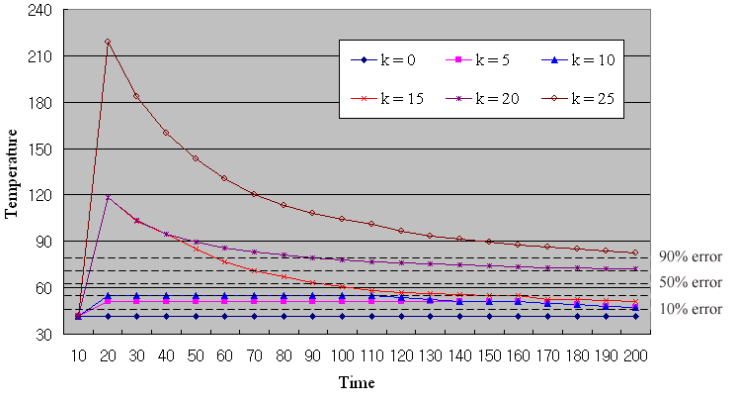
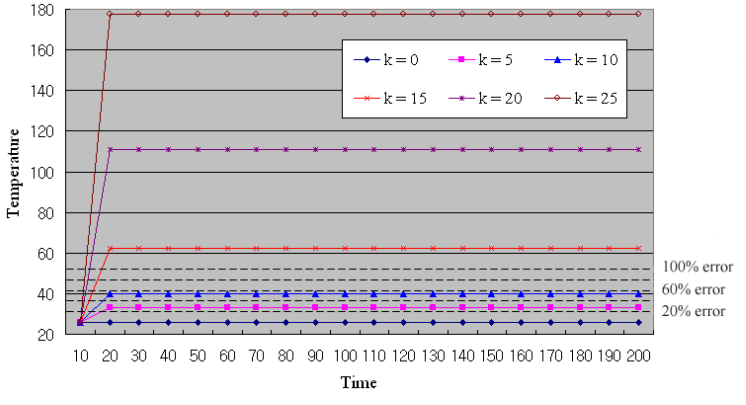


Fig. 5. Aggregation Results under k-node Attack with Static Trust Values



(a) Trust-based Aggregation Results



(b) Median

Fig. 6. Aggregation Results under k-node Attack with Dynamic Trust Values

dynamically by the system. Unlike the previous simulation, the same event occurs every 10 seconds. Attackers start to broadcast false data from 20 second and their trust values are evaluated continuously as the time elapses. Based on these trust values of attackers, aggregated data of the system can be computed as in Figure 6. The dotted lines represent the error rate from the original aggregated data to every increasing 20 percent.

In the case of the trust-based aggregation scheme of Figure 6(a), when k=0, the aggregated data represent 41.81895°C. Under 5-node and 10-node attack, the compromised data can converge to 47.51624°C whose errors from the original data are about 10 percent in 200 seconds. Under 15-node attack, the compromised data can converge to 51.44236°C whose errors are about 20 percent in 200 seconds. Under 20-node and 25-node attack, the compromised data can converge to 74.94018°C and 82.59066°C whose errors are approximately 70 percent and 90 percent, respectively, in 200 seconds. In the case of the median of Figure 6(b), when k=0, the original median values represent 26.014568°C. Under 5-node attack and 10-node attack, errors of the compromised medians are

larger than 30 percent and 50 percent from the original data, respectively. When k=15, the errors exceed 140 percent. Consequently, the trust-based aggregation scheme shows higher robustness than median in the same environment.

6 Conclusion

We proposed a trust management scheme for wireless sensor networks to make the sensor networks resilient against false data injection. Our trust evaluation scheme is best-suited to settings where there is plenty of redundancy in the data of sensor nodes, so that we can crosscheck sensor readings for consistency. Wireless sensor networks tend to consist of a large scale of cheap and crude sensors, which are exactly where our trust evaluation scheme is most appropriate. As the degree of redundancy in the sensing data increases, proposed trust management scheme would be more applicable in an increasing variety of applications in wireless sensor networks.

Our trust management scheme does not employ cryptographic approaches or certification mechanisms, so it is light enough to fit well with wireless sensor networks without great overheads. In addition, as we analyzed, under the condition that the trustworthiness of the sensor nodes are estimated precisely, the proposed trust-based aggregation scheme could be more resilient alternatives to median, which is known to be the most robust conventional aggregation scheme.

To the best of our knowledge, our approach is one of the incipient researches on the trust management scheme for wireless sensor networks. The proposed scheme can make the wireless sensor networks be able to detect malicious or compromised sensor nodes, and filter out the false data of them. We expect our trust management scheme to enhance the resilience of wireless sensor networks.

Acknowledgement

This work was supported by the Ministry of Science and Technology(MOST) / Korea Science and Engineering Foundation(KOSEF) through the Advanced Information Technology Research Center(AITrc) and the Ministry of Information and Communication(MIC), Korea, under the Information Technology Research Center(ITRC) support program supervised by the Institute of Information Technology Assessment(IITA).

References

1. H. Chan and A. Perrig, *Security and Privacy in Sensor Networks*, IEEE Computer 2003.
2. A. Pirzada, C. McDonald, *Establishing Trust In Pure Ad-hoc Networks*, Proceedings of the 27th conference on Australasian computer science, 2004.
3. A. Perrig, J. Stankovic, D. Wagner, *Security in Wireless Sensor Networks*, Communication of the ACM, June 2004.
4. C. Karlof, D. Wagner, *Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures*, NEST 2003.

5. Z. Yan, P. Zhang, T. Virtanen, *Trust Evaluation Based Security Solution in Ad Hoc Networks*, NordSec 2003, Proceedings of the Seventh Nordic Workshop on Secure IT Systems, 15th-17th October 2003.
6. N. Sastry, U. Shankar, D. Wagner, *Secure Verification of Location Claims*, Proceedings of the 2003 ACM workshop on Wireless security.
7. B. Przydatek, D. Song, A. Perrig, *SIA: Secure Information Aggregation in Sensor Networks*, SenSys 2003.
8. B. Krishnamachari, D. Estrin, S. Wicker, *The Impact of Data Aggregation in Wireless Sensor Networks*, ICDCSW, Proceedings of the 22nd International Conference on Distributed Computing Systems, pp. 575–578, 2002.
9. J. Deng, R. Han, S. Mishra, *Security Support for In-Network Processing in Wireless Sensor Networks*, Conference on Computer and Communications Security, Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks, pp. 83–93, 2003.
10. S.S. Doumit, D.P. Agrawal, *Self-Organized Criticality and Stochastic learning based intrusion detection system for wireless sensor networks*, Military Communications Conference, 2003. MILCOM '03. 2003 IEEE, pp.609–614.
11. L. Eschenauer, V. D. Gilgor, *A key-management scheme for distributed sensor networks*, Proceedings of the 9th ACM Conference on Computer and Communication Security, ACM Press, New York, 2002, pp. 41–47.
12. H. Chan, A. Perrig, D. Song, *Random key predistribution schemes for sensor networks*, IEEE Symposium on Security and Privacy, Berkely, California, May 11-14 2003, pp. 197–213.
13. W. Du, J. Deng, Y.S. Han, P.K. Varshney, *A pairwise key pre-distribution scheme for wireless sensor networks*, Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS), Washington, DC, USA, October 27-31 2003, pp. 42–51.
14. A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, *SPINS: Security Protocols for Sensor Networks*, Wireless Networks Journal (WINE), September 2002.
15. L. Hu and D. Evans, *Secure Aggregation for Wireless Networks*, In Workshop on Security and Assurance in Ad hoc Networks. January 2003.
16. S. Zhu, S. Setia, S. Jajodia, P. Ning, *An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks*, Proceedings of IEEE Symposium on Security and Privacy, Oakland, California, May 2004.
17. F. Ye, H. Luo, L. Zhang, *Statistical En-route Detection and Filtering of Injected False Data in Sensor Networks*, Proceedings of IEEE INFOCOM 2004.
18. David Wagner, *Resilient Aggregation in Sensor Networks*, ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04), October 25, 2004.
19. N. Shrivastava, C. Buragohain, D. Agrawal, S. Suri, *Medians and Beyond: New Aggregation Techniques for Sensor Networks*, Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004), August 16 2004.
20. Z. Li, W. Trappe, Y. Zhang, B. Nath, *Robust Statistical Methods for Securing Wireless Localization in Sensor Networks*, IPSN 2005, Los Angeles, April 2005.
21. X. Ji, H. Zha, *Robust Sensor Localization Algorithm in Wireless Ad-hoc Sensor Networks*, Proceedings of the 12th International Conference on Computer Communications and Networks (ICCCN03), 2003.
22. L. Lazos, R. Poovendran, *SeRLoc: Secure Range-Independent Localiztion for Wireless Sensor Networks*, Proceedings of the 2004 ACM Workshop on Wireless Security, pp. 21–30, 2004.

Improvements to Mitchell's Remote User Authentication Protocol

Vipul Goyal[1], Abhishek Jain[1], and Jean Jacques Quisquater[2]

[1] CSE Department, IT-BHU, India
vipul.goyal@cse04.itbhu.org,
a.jain@cse06.itbhu.org
[2] Crypto Group, UCL, Belgium
jjq@dice.ucl.ac.be

Abstract. A provably secure protocol for remote authentication is presented. Only public information is stored at the verifying host that makes our scheme resistant to server compromise. We use one time signatures coupled with offline transcripts for synchronization. Due to sole usage of fast cryptographic hash functions, our method is appropriate for low cost user authentication. Our construction improves over the previously proposed technique of Mitchell to overcome its problem of Denial of Service (DoS) attacks.

1 Introduction

Authentication is the process by which a system can determine whether or not a given user is who she claims to be. Authentication is the key for information security since if the authentication mechanism is compromised, the rest of the security measures are bypassed as well.

Authentication protocols using asymmetric key cryptography generally suffer from performance limitations. Recall that asymmetric key cryptography is typically hundreds of times slower than symmetric key cryptography. This increases the cost of authentication servers several times and renders the protocol impractical for clients with limited computational resources. On the other hand, authentication protocols using symmetric[1] key cryptography are vulnerable to either server compromise or network attacks (passive and active). One particularly widespread attack is to passively capture and replay passwords used for user authentication. A solution to this problem is to encode the passwords in such a way that it can be used exactly once and cannot be used to generate any other encoded password. Such an encoding is called a one-time password. A one-time password scheme was first designed by Lamport [16]. Later it was standardized [11,12,13] and implemented [10,17]. However, Lamport's scheme remains vulnerable to an active adversary who intercepts and traps (or impersonates the host in order to extract) an as yet unused one time password so that he can generate a list of valid unspent OTPs by using the hash chain's one-way property. This

[1] To avoid confusion, we collectively refer to protocols using (a) Symmetric key encryption (b) One-way Hash functions as symmetric key authentication protocols in this paper.

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 69 – 80, 2006.
© Springer-Verlag Berlin Heidelberg 2006

has been pointed out by a number of authors [8,14,18]. Besides, the number of times a user may authenticate to the server is limited.

A symmetric key-based unilateral authentication protocol was recently proposed by Mitchell [21,22] requiring only public information to be stored at the authentication server. Message Authentication codes (MAC) [1,2] were used to maintain storage and computational feasibility for low cost user authentication devices. The scheme has formed part of the Software Based Systems area of the Core 2 Research Programme of the Mobile VCE [24]. However, as acknowledged by the author, the scheme is vulnerable to a DoS attack. Besides, the protocol lacks proofs of security. We discuss this scheme and the associated vulnerabilities in detail in the next subsection.

We present a new key-based remote user authentication protocol using one-time signatures coupled with offline transcripts for synchronization. In our attempt to improve over Mitchell's scheme [22], the relevant goals we wish to achieve are –

- Provable security against server compromise.
- Provable security against passive and active adversaries.
- Practical solution for low cost user authentication devices.
- Unlimited authentications without system re-initialization.

The rest of the paper is organized as follows. Section 2 provides a background on one-time signatures. We present our scheme in Section 3. In section 4, we discuss the security and performance issues of the proposed scheme and provide a comparison with [22]. An enhancement to our scheme is discussed in Section 5. We conclude in Section 6.

Throughout the paper, A, B (*Alice*, *Bob*) are used to represent system principals, where *Alice* plays the role of client while *Bob* plays the role of server. We denote concatenation of two values by (X, Y). By $\{a_i\}_{i=1}^{n}$ we denote the set $\{a_1, ..., a_n\}$ of n entities (e.g. secret keys, MAC values etc). Notations specific to a system are introduced as and when required.

1.1 Related Works

Here we discuss the scheme proposed by Mitchell [21,22]. The scheme is divided into two phases, *Setup* and *Operation*. A MAC computation over a string X is denoted as $M_K(X)$ where K is the secret key.

Protocol. Mitchell's remote user authentication scheme

I. *Setup*

1. A performs the following operations.
   - Choose an integer constant t, key set $\{K_i\}_{i=1}^{t}$ and a random data string X.
   - Compute $V_i = M_{K_i}(X), \forall i \in [1, t]$.

- Transfer X, $\{V_i\}_{i=1}^{t}$ to B in a manner guaranteeing their authenticity.

2. B stores the received data. Note that data integrity is the only assumption.

II. *Operation*

1. A sends login request to B.
2. B sends X to A.
3. A performs the following operations.
   - Verify X against the stored value (A mismatch signifies a loss of synchronization between A and B. We discuss this situation later on).
   - Compute a new key set $\{K_i'\}_{i=1}^{t}$, random string X', MAC values $V_i' = M_{K_i'}(X')$ and $W_i' = M_{K_i}(V_1', ..., V_t')$, $\forall i \in [1, t]$.
   - Send $\{W_i'\}_{i=1}^{t}$ to B.
4. B sends a random r subset of $\{i\}_{i=1}^{t}$, say $\{c_i\}_{i=1}^{r}$.
5. A sends $\{K_{c_i}\}_{i=1}^{r}, X', \{V_i'\}_{i=1}^{t}$ to B and replaces $X, \{K_i\}_{i=1}^{t}$ with $X', \{K_i'\}_{i=1}^{t}$ respectively. (In some cases, the *old* values may be retained as discussed below).
6. B first verifies the stored MAC subset $\{V_{c_i}\}_{i=1}^{r}$ using key subset $\{K_{c_i}\}_{i=1}^{r}$ and the stored value X. If the entire MAC subset is correct, then B verifies the MAC subset $\{W_{c_i}'\}_{i=1}^{r}$. If this is also correct, B authenticates A and replaces $X, \{V_i\}_{i=1}^{t}$ with $X', \{V_i'\}_{i=1}^{t}$.

As acknowledged in [22], the scheme is vulnerable to a simple and effective DoS attack. An adversary may first capture the random string X sent by A to B during a protocol execution (step 5 in the above protocol) and then later impersonate B to A using X. Although the login attempt would fail, a single protocol execution between the adversary and A would result in a new random string at A for the next login protocol while that stored at B is still the old one. This results in a loss of synchronization between A and B. Further, this protocol execution would also enable the adversary to learn a chosen subset of r keys sent out by the user in step 4. The adversary may possibly use these keys later to impersonate A to B. Thus, the scheme requires careful choice of t and r so that the probability of an adversary successfully guessing the subset $\{c_i\}_{i=1}^{r}$ in advance is negligible (see Section 3.1 in [22] for details).

When A later tries to authenticate herself to B, step 2 of the protocol would enable her to learn about the loss of synchronization. Here two options are possible, re-initialize the system manually or use the same key set that was used with the

adversary, again for authentication to B. However, there is a potential danger in using the same key set again as the adversary may again impersonate B to A to learn another chosen set of keys. The adversary may choose the new r subset such that it is disjoint with the old key set she already has; she may keep impersonating B to A until she has a significant portion of keys out of t thus enabling her to impersonate A to B.

[22] proposes to increase t to enable the user to securely use the same key set a number of times. For example, to use the same key set s times, t may be set to $64 \times s$ with $r = 32$ to ensure that the probability of a successful attack remains less than 2^{-32}. Thus, security may be increased at the cost of performance. However, it is clear that irrespective of how big s may be, the adversary may keep impersonating B to A up to s times so that the key set expires and re-synchronization becomes impossible by using the same key set. In cases where it is possible, the system may be re-initialized by manual means. In other cases, there is no way for the user to authenticate any further. Thus, the DoS attack succeeds.

2 Preliminaries and Notations

Hash Functions. Throughout the paper, $H : \{0,1\}^* \rightarrow \{0,1\}^k$ is a k-bit cryptographic hash function (informally known as a one-way hash function [3]) that is hard to invert and collision resistant.

One time Signatures. The concept of one time signatures was initially proposed by Lamport [15]. It was subsequently enhanced by Merkle [19,20], Winternitz [21], Bicakci et. al. [4] and Bleichenbacher et. al. [5,6,7].

Signing a one bit message. The signer chooses as the secret key two values X_1 and X_2 (representing '0' and '1') and publishes their images under a one-way function $Y_1 = H(X_1)$ and $Y_2 = H(X_2)$ as the public key. These *X*'s and *Y*'s are called the secret key components and the public key components, respectively. To sign a single bit message, reveal the pre-image corresponding to the actual '0' or '1' i.e., reveal X_1 or X_2 based upon whether the message to be signed is '0' or '1'.

For signing longer messages, several instances of this basic scheme may be used. Thus we note that to sign an N bit message, $2N$ *X*'s and thus $2N$ *Y*'s are required and the size of signatures generated is equal to N times the size of secret key components.

There are several improvements to this basic scheme. Merkle [19,20] proposed an improvement which reduces the number of public as well as secret key components in Lamport's method by almost two-fold. Instead of generating two *X*'s and two *Y*'s for each bit of the message, the signer generates only one *X* and one *Y* for each bit of the message to be signed. When one of the bits in the message to be signed is a '1', the signer releases the corresponding value of *X*; but when the bit to be signed is a '0', the signer releases nothing. Because this allows the receiver to pretend that he did not receive some of the *X*'s and therefore to pretend that some of the '1' bits in the signed message were '0', the signer must also sign the count of the '0' bits in the message.

Now, when the receiver pretends that a '1' bit was actually a '0' bit, he must also increase the value of the count field, which can't be done. Because the count field has only $log_2 N$ bits in it, the number of public and secret key components is decreased by almost a factor of two i.e., from $2N$ to $N + log_2 N$ (or to $N + log_2 N + 1$ if N is not a power of 2). This also results in the decrease of signature size by almost a factor of two.

Winternitz [20] proposed an improvement that reduces the signature size by several folds at the expense of increased computational effort. In Winternitz's method, the one-way function is applied to two secret key components iteratively for a fixed number of times, resulting in a two-component public key.

We now introduce some basic notations associated with one-time signatures used throughout the paper.

m	Number of public/secret key components used in the OTS scheme; equal to $k + log_2(k)$ for Merkle's construction if the output of the hash function is k-bit.
p	Average number of secret key components revealed in an OTS; usually equal to $m/2$.
sk	A one-time secret key; equal to the collection of m secret key components.
pk	A one-time public key; equal to the collection of m public key components.
$S(sk, M)$	One-time signature computed over message M with sk; equal to the collection of the relevant secret key components required to sign M.

Calculation of a one-time key pair. Given a secret key K^A and an integer counter value i, a one-time key pair $\left(sk_i^A, pk_i^A\right)$ for a system principal A is computed as per the following definition.

$$sk_i^A = \left\{H\left(K^A, i, 1\right), ..., H\left(K^A, i, m\right)\right\}$$

$$pk_i^A = \left\{H^2\left(K^A, i, 1\right), ..., H^2\left(K^A, i, m\right)\right\}$$

For a given integer value i, the corresponding one-time key pair for a system principal may be computed whenever required.

3 The Proposed Scheme

Our scheme is divided into two phases, *Setup* and *Operation*. Offline transcripts are computed during the operation phase to restore synchronization whenever required. We use an integer i as a counter for protocol executions.

Protocol. Remote Authentication using One-Time Signatures

I. *Setup*

1. A initializes a counter $i^A \leftarrow 1$ and performs the following operations.
   - Choose a secret key K^A, compute a one-time key pair $\left(sk_1^A, pk_1^A\right)$.
   - Transfer pk_1^A to B in a manner guaranteeing its authenticity. This may be done using a certificate issued by a trusted authority.
2. B stores the received value and initializes his counter $i^B \leftarrow 1$.

II. *Operation*

1. A identifies herself to B with her login name.
2. B computes a random string r_i^B and sends $\left(i^B, r_i^B\right)$ to A.
3. A performs the following operations.
   - Verify i^B against i^A (In case of a mismatch, compute offline transcript)
   - Compute sk_i^A, a new one-time key pair $\left(sk_{i+1}^A, pk_{i+1}^A\right)$, hash value $H\left(r_i^B, pk_{i+1}^A\right)$ and the OTS $S\left(sk_i^A, H\left(r_i^B, pk_{i+1}^A\right)\right)$.
   - Send $pk_{i+1}^A, S\left(sk_i^A, H\left(r_i^B, pk_{i+1}^A\right)\right)$ to B. Store r_i^B and set counter $i^A \leftarrow i^A + 1$.
4. B verifies the correctness of the received OTS. If the OTS is correct, A is authenticated. B replaces pk_i^A with pk_{i+1}^A and sets the counter $i^B \leftarrow i^B + 1$.

At any point of time, A stores an integer counter and a set of challenge strings sent to her by B, while B stores an integer counter and a one-time public key.

Offline Transcript. In case of a mismatch between the counter values in step 3 of the operation phase, A computes an offline transcript. Further actions remain the same as explained in the protocol. We suppose that the counter value at A is $(i+d)$ while that received from B is i. Then the offline transcript is the set $\left\{r_j^B, pk_{j+1}^A, S\left(sk_j^A, H\left(r_j^B, pk_{j+1}^A\right)\right)\right\}_{j=i}^{i+d-1}$ consisting of d entries. To compute the offline transcript, A performs the following operations.

- Compute the one-time key pair $\left(sk_j^A, pk_j^A\right)$, $\forall j \in [i, i+d]$.
- Locate challenge string r_j^B, $\forall j \in [i, i+d)$ from the database.
- Compute the OTS $S\left(sk_j^A, H\left(r_j^B, pk_{j+1}^A\right)\right)$, $\forall j \in [i, i+d)$.

Now to authenticate herself to B, A sends the offline transcript along with the *current* OTS $S\left(sk_{i+d}^A, H\left(r_{i+d}^B, pk_{i+d+1}^A\right)\right)$ and one-time public key pk_{i+d+1}^A. B uses

the offline transcript to first update his state from pk_i^A to pk_{i+d}^A. Now he is in a position to verify the c*urrent* OTS and hence allow A to log in. It is worth noting here that the offline transcript itself is insufficient for authentication. We stress that the c*urrent* OTS be sent along with it. This is because the OTS in the transcript are based on old challenge strings that are more likely to have been generated by an adversary impersonating B to A during earlier protocol executions (This will be more clear after we discuss DoS attack in Section 4.1). Further, the content of the offline transcript is just a collection of the regular authentication data sent during previous authentication sessions and may be known to an adversary. Hence the transcripts are just a method of *catch-up* for the host.

4 Security Analysis

In the security proofs given in this section, we refer to the pair $\left\{pk_{i+1}, S\left(sk_i, H\left(r_i, pk_{i+1}\right)\right)\right\}$ as a one-time password (OTP). The superscripts referring to a system principal have been omitted in both the security proofs.

Theorem 1. *If the underlying one-time signature scheme is secure, then for any integer counter value i, given access to all the one-time passwords sent by the client up to login $(i-1)$ and the current server state pk_i, an adversary cannot compute the correct i^{th} one-time password.*

Proof. Assume that the thesis is false i.e., there is an algorithm F that succeeds in computing the correct one-time signature for some login (without the knowledge of the one-time secret key) with non-negligible probability. That is, for 'some' integer value i, F runs on the current server state pk_i and the OTP set $\left\{pk_{j+1}, S\left(sk_j, H\left(r_j, pk_{j+1}\right)\right)\right\}_{j=1}^{i-1}$, takes a random string r_i and one-time public key pk_{i+1} as input and outputs a valid one-time signature over their hashed value. (On all other inputs, it outputs zero). We show how F can be used to build an algorithm F_1 which forges signatures for the underlying one-time signature scheme.

F_1 is allowed to query an oracle O which works in the following manner. On being queried with an integer value i as input, it computes a one-time key pair (sk_i, pk_i) and outputs pk_i; when queried with an integer i and some string as input, it outputs the one-time signature computed over the input string using sk_i. F_1 executes the following algorithm.

Algorithm 1

1. Set counter $i = 1$
2. Query O with input i. Receive and store pk_i
3. Query O with input $i+1$. Receive and store pk_{i+1}

4. Compute a random string r_i and the hash value $H(r_i, pk_{i+1})$
5. Transfer the OTP set $\left\{pk_{j+1}, S\left(sk_j, H\left(r_j, pk_{j+1}\right)\right)\right\}_{j=1}^{i-1}$ and current server state pk_i to F and execute it on input $H(r_i, pk_{i+1})$. If F outputs non-zero then Stop.
   Else Query O with input $\{i, H(r_i, pk_{i+1})\}$. Receive and store $S\left(sk_i, H\left(r_i, pk_{i+1}\right)\right)$
6. Set $i \leftarrow i+1$, go to step 3

With non-negligible probability, for some integer counter value i, F outputs a valid signature S' over the given input. Now, F_1 stops and outputs S' as a valid one-time signature without the knowledge of the one-time secret key and without having made a query to the oracle O for the new one-time signature. This contradicts the assumption that the underlying one-time signature scheme is secure. Hence our theorem is proved.

Denial of Service Attacks. We now discuss the effect of DoS attack on our protocol. Observe that an unsuccessful authentication is likely to be caused due to an adversary impersonating B to A. We suppose that *since* the last successful login, the adversary has engaged in d protocol executions with A impersonating B to A each time. This implies that d is the difference between the counter value stored by the system principals. Hence, if the counter value at A was i when the adversary started the attack, it must be $(i+d)$ at this point.

Assume that during $(i+d)^{th}$ protocol execution, A receives $(i+d)^B$, r_{i+d}^B from B. After the failure of the counter value verification, A computes an offline transcript which is a set of cardinality d as explained in Section 3. Now A sends this offline transcript to B along with the usual generated *current* OTS and one-time public key. B verifies each OTS in the transcript one by one and keeps updating its state. If the verification of all the one-time signatures in the transcript is successful, the state of B is updated to pk_{i+d}^A, the counter value being $(i+d)$. Now he is in a position to verify the regular OTS sent along with the offline transcript. If it is found correct, A is logged in and counter is updated to $(i+d+1)$.

Again, it may be the adversary impersonating B to A and exchanging messages with A. In that case, during the next login attempt, A will add the current OTS, one-time public key and the challenge string to the offline transcript to be sent to B. Hence it is clear that though, higher the number of consecutive attacks, higher the size of offline transcript; despite any number of attacks, synchronization can still be restored between A and B. Further, A may not need to store all the challenge strings sent to him so far. When she is sure that the login attempt was successful and she is logged into the genuine server (e.g. on seeing her files or e-mails), she may delete all the challenge strings stored until that point. In section 6, we propose an enhancement to the scheme which does not require such manual intervention on the client side.

Security of offline transcripts. Recall that though both Message Authentication Codes [1,2] and Digital Signatures [9,23] are used for authentication purposes, digital signatures provide non-repudiation whereas MACs are repudiable. The reason being that *the verification data released to the verifier in case of digital signatures does not give him the power to modify or create a signature while the secret key released to the verifier in case of MAC gives him modification power.* We now discuss the security of the offline transcript mechanism. We claim that this mechanism which employs only one-time signatures is secure and possesses the property of offline verification due to the same reason as discussed above i.e., the verification data does not give modification power to the verifier.

Theorem 2. *If the underlying one-time signature scheme is secure and* H *is a collision resistant hash function, then given access to the current server state* pk_i*, an adversary cannot modify an offline transcript and the attached current one-time password to his advantage.*

Proof. Assume that the thesis is false i.e., there is an algorithm F that succeeds in modifying an offline transcript with non-negligible probability. That is, for some integer values i and d, F runs on current server state pk_i, takes as input set $Z = \left\{ r_j, pk_{j+1}, S\left(sk_j, H\left(r_j, pk_{j+1}\right)\right)\right\}_{j=i}^{i+d}$ (the current one time password being $\left\{ pk_{i+d+1}, S\left(sk_{i+d}, H\left(r_{i+d}, pk_{i+d+1}\right)\right)\right\}$) and outputs a modified *valid* set Z'. We show how F can be used to build an algorithm F_1 which either forges signatures for the underlying one-time signature scheme or finds collisions in the hash function.

F_1 is allowed to query an oracle O which works in the same manner as discussed in the Proof of Theorem 1. F_1 first executes Algorithm 2 with input (i,d).

Algorithm 2. *Prepare Transcript (i, d)*

1. Set $j = i$
2. Query O with input j. Receive and store pk_j
3. Query O with input $j+1$. Receive and store pk_{j+1}
4. Compute a random string r_j and the hash value $H\left(r_j, pk_{j+1}\right)$
5. If $j = i+d$ Stop
   Else Query O with input $\left\{ j, H\left(r_j, pk_{j+1}\right)\right\}$. Receive and store $S\left(sk_j, H\left(r_j, pk_{j+1}\right)\right)$
6. Set $j \leftarrow j+1$, go to step 3

It now runs F on current server state pk_i with the offline transcript $Z = \left\{ r_j, pk_{j+1}, S\left(sk_j, H\left(r_j, pk_{j+1}\right)\right)\right\}_{j=i}^{i+d}$ as input. With non-negligible probability, F

outputs a modified *valid* set Z'. F_1 now compares Z and Z' for each corresponding entry. On finding the first mismatch, it stops. Suppose that the mismatch occurs at some j. Now, if $H\left(r_j, pk_{j+1}\right)^Z = H\left(r_j, pk_{j+1}\right)^{Z'}$ and $\left(r_j, pk_{j+1}\right)^Z \neq \left(r_j, pk_{j+1}\right)^{Z'}$; this means F_1 has found a collision in the hash function. Otherwise, F_1 outputs the one-time signature from the j^{th} entry in Z' as a valid one-time signature, thus contradicting the assumption that the underlying one-time signature scheme is secure. Hence our theorem is proved.

5 An Enhancement to the Scheme

Mutual Authentication is a desired feature of a two-party authentication protocol. Both A and B must authenticate each other to prevent an adversary from impersonating either of them.

In the proposed scheme, an adversary is able to disturb the synchronization between the system principals due to lack of mutual authentication. Though offline transcripts successfully counter such attacks, it would be appropriate to eradicate such an attack in the first place. To support mutual authentication, the basic protocol can be extended by making A authenticate B in exactly the same manner as B authenticates A. The extended scheme consists of two phases, *Setup* and *Operation*. Offline transcripts are no longer required since loss of synchronization is no longer possible.

Protocol. Remote Authentication using One-Time Signatures (Extended Version)

I. *Setup*

1. A initializes a counter $i^A \leftarrow 1$ and performs the following operations.
   a. Choose a secret key K^A, compute a one-time key pair $\left(sk_1^A, pk_1^A\right)$.
   b. Transfer pk_1^A to B in a manner guaranteeing its authenticity. This may be done using a certificate issued by a trusted authority.
2. B stores the received value and initializes his counter $i^B \leftarrow 1$. He then performs the following operations.
   a. Choose a secret key K^B, compute a one-time key pair $\left(sk_1^B, pk_1^B\right)$.
   b. Transfer pk_1^B to A in a manner guaranteeing its authenticity.

II. *Operation*

1. A computes a random string r_i^A and sends $\left\{A, i^A, r_i^A\right\}$ to B.
2. B performs the following operations.
   - Verify i^A against i^B. Stop if mismatch occurs.
   - Compute sk_i^B, a new one-time key pair $\left(sk_{i+1}^B, pk_{i+1}^B\right)$, hash value $H\left(r_i^A, pk_{i+1}^B\right)$ and the OTS $S\left(sk_i^B, H\left(r_i^A, pk_{i+1}^B\right)\right)$.

- Compute a random string r_i^B and send $r_i^B, pk_{i+1}^B, S\left(sk_i^B, H\left(r_i^A, pk_{i+1}^B\right)\right)$ to A.

3. A performs the following operations.
   - Verify the received OTS. Stop if verification fails.
   - Compute sk_i^A, a new one-time key pair $\left(sk_{i+1}^A, pk_{i+1}^A\right)$, hash value $H\left(r_i^B, pk_{i+1}^A\right)$ and the OTS $S\left(sk_i^A, H\left(r_i^B, pk_{i+1}^A\right)\right)$.
   - Send $pk_{i+1}^A, S\left(sk_i^A, H\left(r_i^B, pk_{i+1}^A\right)\right)$ to B. Replace pk_i^B with pk_{i+1}^B and set counter $i^A \leftarrow i^A + 1$.
4. B verifies the correctness of the received OTS. If it is correct, A is authenticated. B replaces pk_i^A with pk_{i+1}^A and sets the counter $i^B \leftarrow i^B + 1$.

6 Conclusions

A new key-based authentication protocol has been presented that uses one-time signatures coupled with offline transcripts for synchronization. The proposed scheme is provably secure against server compromise as well as both passive and active adversaries. The mechanism of offline transcripts provides immunity against DoS attacks and thus improves over Mitchell's scheme [21,22]. The extended protocol supports mutual authentication and eliminates any possibility of loss of synchronization in the first place. Due to the sole usage of fast cryptographic hash functions, the proposed scheme maintains trivial computational and storage requirements to be a practical solution for low cost user authentication devices where the complexity of implementing public key cryptography should be avoided.

References

[1] M. Bellare, R. Canetti, H. Krawczyk, "Keying Hash Functions for Message Authentication", *CRYPTO 96*, LNCS Vol. 1109.
[2] M. Bellare, R. Canetti, H. Krawczyk, "HMAC: Keyed-Hashing for Message Authentication", *RFC 2104,* February 1997.
[3] T.A. Berson, L. Gong and T.M.A Lomas, "Secure, Keyed and Collisionful Hash Functions", *Technical Report SRI-CSL-94-08,* May 1994.
[4] K. Bicakci, G. Tsudik, B. Tung, "How to construct optimal one-time signatures", *Computer Networks (Elsevier),* Vol.43 (3), pp.339-349, 2003.
[5] D. Bleichenbacher and U. M. Maurer, "Directed Acyclic Graphs, One-way Functions and Digital Signatures", *Proc. CRYPTO 94*, LNCS 839, Springer Verlag, 1994, pp 75-82.
[6] D. Bleichenbacher, U. M. Maurer, "Optimal Tree-Based One-time Digital Signature Schemes", *Proc. STACS 96*, LNCS 1046, Springer Verlag, pp 363-374.
[7] D. Bleichenbacher, U. M. Maurer, "On the efficiency of one-time digital signatures", *Proc. ASIACRYPT 96*, LNCS 1163, Springer-Verlag, pp 145-158, 1996.

[8] L. Chen, C.J. Mitchell, "Comments on the S/Key user authentication scheme", *ACM Operating Systems Review*, 30(4), 12-16, October 1996.
[9] W. Diffie, M. Hellman. "New Directions in Cryptography", *IEEE Transactions on Information Theory*, IT-22 (6), 74-84, 1976.
[10] N. Haller, "The S/Key One-Time Password System", *Proceedings of the ISOC Symposium on Network and Distributed System Security*, pp 151-157, February 1994.
[11] N Haller, "The S/KEY One-Time Password System", *RFC 1760*, 1995.
[12] N. Haller, "A One-Time Password System", *RFC 1938*, May 1996.
[13] N Haller, C. Metz, P. Nesser and M. Straw, "A One-Time Password System", *RFC 2289*, Feb 1998.
[14] C. Kaufman, R. Perlman, M. Speciner, "Network Security, Private Communication in a Public World", Prentice Hall Series, 2002.
[15] L. Lamport, "Constructing Digital Signatures from a One-Way Function", *Technical Report CSL-98*, SRI International 1978.
[16] L. Lamport, "Password Authentication with Insecure Communication", *Communications of the ACM* 24.11 (November 1981), pp 770-772.
[17] D.L. McDonald, R.J. Atkinson, C. Metz, "One-Time Passwords in Everything (OPIE): Experiences with Building and Using Strong Authentication", *In Proc. of the 5th USENIX UNIX Security Symposium*, June 1995.
[18] A. J. Menzees, P. C. Van Oorschot and S. A. Vanstone, "Handbook of Applied Cryptography", CRC Press, Boca Raton, 1997.
[19] R. C. Merkle, "A Digital Signature Based on a Conventional Encryption Function", *Proc. CRYPTO 87*, LNCS 293, Springer Verlag, pp 369-378.
[20] R. C. Merkle, "A Certified Digital Signature", *Proc. CRYPTO 89*, LNCS 435, Springer Verlag, 1990, pp 218-238.
[21] C. J. Mitchell, "Authentication of a remote user to a host in a data communication system", *UK patent application* filed 3rd November 2001.
[22] C. J. Mitchell, "Remote user authentication using public information", *9th IMA International Conference on Cryptography and Coding*, Cirencester, UK, December 2003, Springer-Verlag (LNCS 2898), pp.360-369.
[23] R. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Comm. of the ACM*, 21(2), 120-126, 1978.
[24] Mobile VCE, www.mobilevce.com.

Efficient Authenticators with Application to Key Exchange

Shaoquan Jiang[1,2] and Guang Gong[2]

[1] Department of Computer Science,
University of Electronic Science and Technology of China, ChengDu, China
[2] Department of Electrical and Computer Engineering,
University of Waterloo, ON, N2L 3G1, Canada
{jiangshq, ggong}@calliope.uwaterloo.ca

Abstract. The notion of authenticator, proposed by Bellare *et al.*, is to transform a protocol secure in the authenticated-link model to a new one secure in the unauthenticated-link model. This notion admits a modular design and analysis of cryptographic protocols and thus greatly simplifies the underlying tasks. However, all previous authenticators are constructed via a so called MT-authenticator. This kind of authenticator authenticates each message independently. Thus, the round complexity of the resulting protocol is amplified by a multiplicative factor. In this paper, we propose two efficient authenticators which authenticate the protocol as a whole and the round complexity of the resulting protocol increases only by at most an additively small number. We also construct a very efficient key exchange protocol. Our protocol is provably secure under the general cryptographic assumption (especially without a concrete hardness assumption such as DDH or RSA). Of an independent interest, our security proof lies in the emulation based ideal-real model, instead of the widely adopted (seemingly weaker) SK-security. To our knowledge, this is the first protocol of its kind. It is worth mentioning that all our constructions are obtained by improving the related protocols of Bellare *et al.* [1].

1 Introduction

Authentication is one of the most important issues in the area of secure communication. It requires that all the messages in transmission should not be modified or changed improperly. This is not an easy cryptographic task since protocols typically are executed concurrently. In addition, an adversary could compromise a party at any time. He also could arbitrarily delete, block, inject, modify the messages over the channel. The topic of authentication has been extensively studied in the literature [2, 16]. Authentication is closely related to the problem of authenticated key exchange. Authenticated key exchange is a procedure that enables two or more parties to jointly compute a common secret in an authenticated and secure manner. The rigorous treatment of key exchange protocol was first due to Bellare *al.* [2, 3] and was later developed by many authors [21, 7, 6].

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 81–91, 2006.
© Springer-Verlag Berlin Heidelberg 2006

1.1 Related Work

Bellare *et al.* [1] formalized the authenticated-link model (AM) and the unauthenticated-link model (UM). The former essentially requires the messages in the channel be faithfully delivered while the latter does not have this requirement. They introduced a notation of *authenticator*, which transforms a protocol secure in the AM to a new one secure in the UM. This notion admits a modular design and analysis of cryptographic protocols. They then constructed several authenticators. In their authenticator, each message is authenticated independently via a MT-authenticator. Since their work, many authors work in this direction. Canetti and Krawczyk [5] slightly extended AM and UM so that the security (called SK-security) of a key exchange protocol can be defined directly (i.e., without the real-ideal emulation paradigm), and then proposed several secure key exchange protocols following the modular approach. Boyd *et al.* [4] constructed two static Diffie-Hellman MT-authenticators and then proposed two static Diffie-Hellman based key exchange protocols using these authenticators. Boyd *et al.* [22] applied the modular approach to the key exchange protocol in the wireless network. Raimondo and Gennaro [18] constructed deniable authenticators, in which the sender can later deny the fact of authentication. Although the modular approach greatly simplifies both the design and analysis of protocols, all the previous authenticators are constructed by simply applying an MT-authenticator to each message independently. As a result, the transformed protocol is very inefficient as its round complexity is amplified by a multiplicative factor.

1.2 Contribution

In this paper, we first construct two efficient authenticators, which authenticate the protocol as a whole (instead of the message-by-message manner). As a result, the round complexity of the transformed protocol increases by at most a small additive number. One authenticator preserves the round complexity, which is technically similar to the compiler designated for the group key exchange [15]. Then, we design two very efficient and secure key exchange protocols by first designing an AM-secure protocol and then applying our authenticators to it. Our protocols are only based on the existence of trapdoor permutation (especially, not assuming any *concrete* harness assumption, e.g. DDH or RSA). We stress that this is important as no one knows when a hardness assumption will be broken later. Our protocols are provably secure in the ideal-real emulation paradigm instead of SK-security [5]. SK-security seems weaker than the security in the real-ideal emulation paradigm. A sound evidence is, the plain Diffie-Hellman key exchange protocol is SK-secure in the AM while it can not be proven secure in the real-ideal emulation paradigm (as mentioned in Appendix A of [5], the security proof for plain DH key exchange protocol in [1] is invalid). To our best of knowledge, our protocols are the first that are provably secure in the emulation paradigm and without a concrete hardness assumption (Note that as mentioned

in Appendix A of [5], the proof of encryption-based key exchange protocol in [1] is invalid). It is worth mentioning our constructions are obtained by improving the related protocols of Bellare *et al.* [1].

2 Model

Bellare *et al.* [1, 5] formalized two security models for n-party protocols: unauthenticated-link model (UM) and authenticated-link model (AM). These models are quite useful in security analysis of protocols. In the following, we overview them. For details, please refer to [5].

Assume $P_1, \cdots, P_n$ are n-parties. π is an arbitrary cryptographic protocol. We consider the execution of π among these parties. We model each party as a probabilistic polynomial time interactive Turning machine. Initially, P_i is invoked with secret input, identity and random input. Then he waits for an **activation**. P_i can be activated by either incoming messages from other parties or external request from other programs within P_i itself. Once activated, P_i follows the specification of π by computing

$$\pi(\text{secret input, internal state, incoming message, external request})$$
$$= (\text{new state, outgoing messages, external requests, output}).$$

Initial internal state is the party's identity and random input. After each activation, the internal state is updated by new state. The internal state does not contain the secret input. Each activation could generate outgoing messages for other parties, external requests for other programs in the same party. It also generates a **local output** and labels the sensitive part as 'secret'.

Each P_i could invoke many copies of π. An invocation is called a **session.** Each session has a session ID. The only requirement for a session ID is its uniqueness in P_i. In order of delivery (also for security), each message sent into the channel is assumed to contain information (sender, sender session ID, receiver, receiver session ID). Note that in a multiparty protocol, a session in P_i might need to communicate with many parties. If session sid_i in P_i is interacting with session sid_j in P_j, we call sid_i in P_i and sid_j in P_j is a paired session.

Unauthenticated-link Model. To introduce the security model, we must include an adversary. We first consider the unauthenticated-link model. In this model, the scheduling of events are determined by adversary $\mathcal{U}$. Such scheduling consists of a sequence of activations in different parties. $\mathcal{U}$ can invoke a party with arbitrary incoming message or external request. Especially, he is not assumed to deliver an message faithfully. He can delete, block, modify and insert any message in the channel. Once a party completes an activation, the outgoing message, outgoing request as well as the local output, except the part labelled as 'secret', are available to $\mathcal{U}$. $\mathcal{U}$ can corrupt a party at any time. When one party gets corrupted, the secret input and the whole internal state within this party is available to $\mathcal{U}$. A special note "corrupted" is appended to the output of this party. Later, this party will not produce an output any more. In addition, this

party's future action is fully taken by $\mathcal{U}$. $\mathcal{U}$ can also corrupt a particular session in P_i. In this case, he obtains the current internal state for this session. A special note of corruption is appended to this session's output. Later, it will not produce an output any more. In the future, the execution of this session is fully taken by $\mathcal{U}$. A session can produce an output at any time (according to the protocol specification). We define an party's final output to be the concatenation of his output history from all sessions.

Let $\boldsymbol{x} = (x_1, \cdots, x_n)$, where x_i is the input for P_i. Let $\boldsymbol{r} = (r_0, \cdots, r_n)$ be the random input, where r_0 for $\mathcal{U}$ and r_i for P_i. We use $\mathrm{Adv}_{\pi,\mathcal{U}}(\boldsymbol{x}, \boldsymbol{r})$ to denote the output of $\mathcal{U}$, and use $\mathrm{UnAuth}_{\pi,\mathcal{U}}(\boldsymbol{x}, \boldsymbol{r})_i$ to denote the output of P_i. Let

$$\mathrm{UnAuth}_{\pi,\mathcal{U}}(\boldsymbol{x}, \boldsymbol{r}) = \mathrm{Adv}_{\pi,\mathcal{U}}(\boldsymbol{x}, \boldsymbol{r}), \mathrm{UnAuth}_{\pi,\mathcal{U}}(\boldsymbol{x}, \boldsymbol{r})_1, \cdots, \mathrm{UnAuth}_{\pi,\mathcal{U}}(\boldsymbol{x}, \boldsymbol{r})_n.$$

Let $\mathrm{UnAuth}_{\pi,\mathcal{U}}(\boldsymbol{x})$ be the random variable describing $\mathrm{UnAuth}_{\pi,\mathcal{U}}(\boldsymbol{x}, \boldsymbol{r})$.

Authenticated-link Model. Authenticated-link model is similar to unauthenticated-link model but with some restrictions. Any outgoing message sent by an uncorrupted session will be faithfully delivered. Each message will be delivered only *at most* once (note that the messages may be delayed or even undelivered. This reflects the nature of an asynchronous network). *This implicitly assumes that a sender never sends an identical message twice.* By default, if the receiver is corrupted, we assume the message is delivered faithfully (recall the delivery is controlled by the adversary himself and thus the delivery of this kind of message is essentially the adversary's internal computation). In addition, the delivery order of messages to an uncorrupted session must obey the protocol specification. Whenever a message is received by an uncorrupted session, it must be indeed sent from the indicated sender session. Note here by default, if the indicated sender session is corrupted, we assume the message is indeed sent from this sender session. Let $\boldsymbol{x} = (x_1, \cdots, x_n)$, where x_i is the input for P_i. Let $\boldsymbol{r} = (r_0, \cdots, r_n)$ be the random input, where r_0 for $\mathcal{A}$ and r_i for P_i. Analogous to UM, we can define $\mathrm{Adv}_{\pi,\mathcal{A}}(\boldsymbol{x}, \boldsymbol{r})$, $\mathrm{Auth}_{\pi,\mathcal{A}}(\boldsymbol{x}, \boldsymbol{r})_i$, $\mathrm{Auth}_{\pi,\mathcal{A}}(\boldsymbol{x}, \boldsymbol{r})$, and $\mathrm{Auth}_{\pi,\mathcal{A}}(\boldsymbol{x})$.

With the above models, we can define the notion of *emulation* as follows.

Definition 1. *Let π and π' be two protocols. We say π'* **emulates** *π in the* UM*, if for any* UM*-adversary $\mathcal{U}$ there exists an* AM*-adversary $\mathcal{A}$ such that for all input $\boldsymbol{x}$,*

$$\mathrm{Auth}_{\pi,\mathcal{A}}(\boldsymbol{x}) \stackrel{c}{\equiv} \mathrm{UnAuth}_{\pi',\mathcal{U}}(\boldsymbol{x}), \tag{1}$$

where $\stackrel{c}{\equiv}$ denotes computational indistinguishability.

3 Protocol Authenticators

Bellare *et al.* [1] proposed a notion of *authenticator*. Essentially, an authenticator is a transformation, which, given a protocol π secure in the AM, outputs a protocol π' secure in the UM. Formally,

Definition 2. *Let $\mathcal{C}$ be a transformation on cryptographic protocols. We say $\mathcal{C}$ is an* **authenticator** *if for any protocol π, $\pi' = \mathcal{C}(\pi)$ emulates π in the* UM.

Given an authenticator, one can construct a UM-secure protocol as follows: first construct an AM-secure protocol, then apply the authenticator to it. Bellare *et al.* constructed authenticators as follows: first construct a MT-*authenticator*, which essentially is a protocol emulating a one-flow protocol in the UM; then an authenticator is defined by applying the MT-authenticator to each message independently. Although the authenticators constructed in this way is secure, it is not efficient since the round complexity of a transformed protocol will increases by a multiplicative factor. In the following two sections, we will introduce our efficient authenticators overcoming this problem.

4 An Encryption-Based Authenticator E-Auth

In this section, we propose an efficient authenticator, denoted by E-Auth. Our authenticator makes use of the modified encryption based MT-authenticator [1]. The original MT-authenticator is flawed [8] but can be easily fixed by adding the message into the encryption. Essentially, since we require the encryption scheme is IND-CCA2 secure, the adversary is unable to mall the ciphertext to a new one with an identical MAC key but a different plaintext. Thus, the receiver's reply ciphertext is tied to a unique session. Notice usually the message is length variable thus the encryption has to be length variable. So, this solution seems inefficient since it might require to compute a ciphertext for a long message. We remark that this is not a problem. Indeed, we can choose to use the hash value of the message into the encryption instead of the message itself. As long as the hash function is collision resistant, the security remains. This technique has been previously employed in [13]. For simplicity, in this work, we do not bother to consider this issue. Our protocol E-Auth only applies this MT-authenticator to the first two messages between a pair of parties while the authenticator in [1] applied the MT-authenticator to every message in the protocol. Now we formally describe E-Auth.

Let π be any protocol. Assume $P_1, \cdots, P_n$ are n parties. Let $m_1, m_2, \cdots$ be the message flows in π exchanged between P_i and P_j. Without loss of generality, suppose m_1 is sent from P_i to P_j. Let $\pi' = \mathsf{E\text{-}Auth}(\pi)$. We use P_i' to denote Party i in protocol π'. π' is augmented with an initialization stage, in which each P_i' received a public/private key pair (e_i, d_i) for encryption scheme E. As in π, P_i' will be provided an initial secret input I_i. Let the public input I_0' be the public input in π concatenated with $\{e_i\}_1^n$. Let MAC be a message authentication code with a key space $\mathcal{K}$. In π', the communication between P_i' and P_j' is defined as follows. P_i' first sends m_1 to P_j', who takes $N_j \leftarrow \mathcal{K}$, responds with message m_1 and $C_1 = E_{e_i}(N_j | m_1)$. Next, P_i' verifies whether C_1 is decrypted to $N_j | m_1$ for some $N_j \in \mathcal{K}$. If no, he rejects; otherwise, P_i' sends back $m_1 || MAC_{N_j}(1, m_1, P_j')$ and in addition generates a local output 'P_i' sent message m_1 to P_j''. When receiving this message, P_j' verifies if the authentication tag is valid. If yes, he generates a local output 'P_j' received message m_1 from

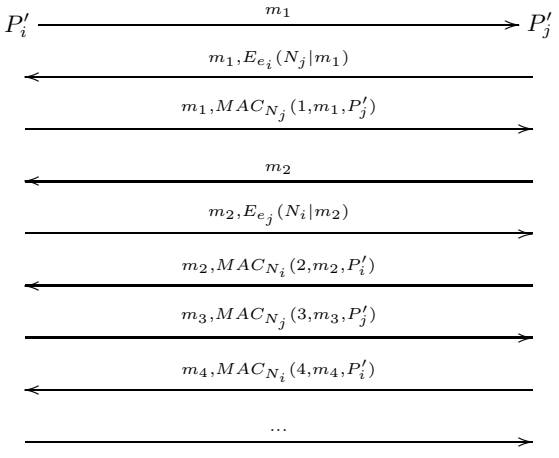


Fig. 1. An encryption-based authenticator E-Auth

P_i'', and follows the execution of P_j in π with incoming message m_1 which in turn generates the outgoing message m_2, and he then uses a similar procedure (as for m_1) to send m_2 to P_i'. Assume P_i' chooses the challenge number N_i in this process. Then for $t > 1$, whenever P_i wishes to send m_{2t-1} in π to P_j, P_i' appends a tag $MAC_{N_j}(2t-1, m_{2t-1}, P_j')$ to m_{2t-1}, and sends the resultant to P_j'. In addition, he generates a local output 'P_i' sent message m_{2t-1} to P_j'''. Receiving the message m_{2t-1} and τ_{2t-1}, P_j' verifies whether the tag τ_{2t-1} is valid. If no, he rejects; otherwise, he outputs 'P_j' received message m_{2t-1} from P_i''' and follows a similar procedure to send m_{2t} (if any) in π to P_i'. This completes the description of π'.

For the description, E-Auth only increases the round complexity by 4 for each pair of parties. In the original authenticator of Bellare *et al.*, each message is applied the encryption-based MT-authenticator independently. We observes that as long as the paired sessions are both uncorrupted, the receiver-chosen number N_i (resp. N_j) will remain cryptographically secure. Thus, each sender only need to invoke the MT-authenticator once and later re-use the receiver-chosen number to authenticate the subsequent messages. To disable reordering and repaying attacks, we include the flow id into the mac tag. In the following theorem, we will show that E-Auth is indeed an authenticator. The proof can be found in the full version [14].

Theorem 1. *Let E be IND-CCA2 secure public key cryptosystem, and MAC be existentially unforgeable message authentication code. Then* E-Auth *is an authenticator.*

5 A Signature-Based Authenticator: Sig-Auth

In this section, we construct our signature-based authenticator (denoted by Sig-Auth) without MT-authenticator.

Sig-Auth authenticator is constructed as follows. Let $P_1, \cdots, P_n$ be n parties and π be any protocol. Let $\pi' = \mathsf{Sig\text{-}Auth}(\pi)$. Let P'_i denote Party i in π'. Let $m_1, m_2, \cdots$ be the messages exchanged between P_i and P_j in π, where w.l.o.g, assume P_i sends m_1. Then the execution between P'_i and P'_j is as follows. Initially, P'_i gets the secret input I_i of P_i in π and signing key s_i of digital signature scheme (G, S, V) (i.e., Key Generation, Signing and Verification Algorithms). In addition, each participant obtains the public-key I_0 in π as well as $\{v_i\}_{i=1}^n$, where v_i is the verification key corresponding to the signing key s_i. The communication between P'_i and P'_j is as follows. Whenever P_i in π wishes to send message m_t to P_j, P'_i sends m_t together with $S_{s_i}(t, m_t, P'_j)$ and then generates a local output 'P'_i sent m_t to P'_j'. Whenever P'_j receives a message $m_t||\tau_t$ from P'_i, he verifies if $\tau_t = V_{v_i}(t, m_t, P'_j, \tau_t)$ (note the flow index t is indicated in the internal state of P'_j). If the verification is successful, then P'_j generates a local output 'P'_j received m_t from P'_i' and then activates π with incoming message m_t to simulate the outgoing message m_{t+1} (if any). If this is the case, then P'_j prepares and sends out $m_{t+1}||S_{s_j}(t+1, m_{t+1}, P'_i)$. If the activation generates any output, P'_j outputs it directly. This completes the description of π'. A graphic interpretation is presented in Figure 2.

Essentially, by signature on message m and flow id, we are guaranteed that the message is authentic and as the correct flow from the right sender session. In the protocol of Bellare *et al.*, correctness of the flow is guaranteed by using a receiver's challenge. Here we authenticate π as a whole and the state of flow id suffices for this purpose since the flow id for the protocol between two parties is labeled sequentially thus is unique. In the following theorem, we show that Sig-Auth is an authenticator. The proof can be found in the full paper [14].

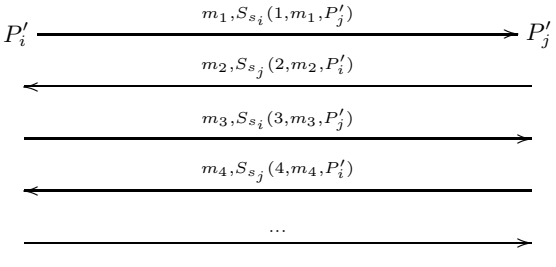


Fig. 2. A digital signature-based authenticator Sig-Auth

Theorem 2. *Let (G, V, S) be a digital signature scheme secure against chosen message attack. Then* Sig-Auth *is an authenticator.*

6 Key Exchange

In this section, we apply our authenticators to the key exchange protocol. We only need to consider the AM-secure construction. In the following, we will first

introduce the ideal process of key exchange. Then, we will introduce our new protocol. Our protocol does not assume any concrete hardness assumption (e.g. Diffie-Hellman [10] or RSA [19]).

6.1 Ideal Process

In the ideal process (See Table 1), we formalize an idealized world for key exchange. The model is concerned with an adaptive malicious adversary under concurrent executions. In this process, parties $P_1, \cdots, P_n$, adversary $\mathcal{S}$ and a trustee T are involved. The output of ideal process, denoted by $\text{IDEAL}_{\mathcal{S}}(r_S, r_T)$, is defined to be the joint output of $P_1, \cdots, P_n$ and the output of $\mathcal{S}$, where r_S and r_T are the random input for $\mathcal{S}$ and T respectively. In the future, we use $\text{Adv}_{\mathcal{S}}(r_S, r_T)$, $\text{IDEAL}_{\mathcal{S}}(r_S, r_T)_i$ to denote the output of $\mathcal{S}$ and P_i respectively. With the ideal process, we define the security of a key exchange protocol as follows.

Table 1. Ideal Process of 2-Party Key Exchange

Participants: $P_1, \cdots, P_n$, trustee T and adversary $\mathcal{S}$.

1. Invoked with input (P_i, P_j, s, I), P_i forwards it to T, where s is the session id which is locally unique and determined by the higher level calling program, I means initiator.
2. Upon receiving (P_i, P_j, s, I), T checks if it was recorded before. If yes, he ignores it; otherwise, he records it.
3. Invoked with (P_j, P_i, s, R), P_j forwards it to T, where R means responder.
4. Upon receiving (P_j, P_i, s, R), if (P_i, P_j, s, I) was not recorded or if (P_j, P_i, s, R) was previously received, T ignores it. Otherwise, if P_i, P_j, session s in P_i, or session s in P_j, is corrupted, he asks $\mathcal{S}$ to choose a session key $sk \in \mathcal{K}$. If no such corruption happened, T takes $sk \leftarrow \mathcal{K}$ (session key domain). Finally, T requests $\mathcal{S}$ to send (P_i, P_j, s, sk) to both P_i and P_j (but sk is invisible to $\mathcal{S}$).
5. Receiving (P_i, P_j, s, sk) from T, P_i (resp. P_j) outputs '(P_i, P_j, s, sk)' directly and labels sk as 'secret'.
6. Upon corruption to session s in P_i, the underlying session key (if any) is provided to $\mathcal{S}$. In addition, a note '**session s in P_i is corrupted**' is appended to the output of P_i. Since that time, this session will not generate an output any more and its future action will be fully taken by $\mathcal{S}$.
7. Upon corruption to P_i, $\mathcal{S}$ learns the entire internal state of P_i including those labelled as 'secret'. A note 'P_i **is corrupted**' is appended to the output of P_i. Since that time, P_i will not generate an output any more and his future action will be fully taken by $\mathcal{S}$.

Definition 3. *Let π be a 2-party key exchange protocol. We say that π is secure in the* UM *if for any* UM*-adversary $\mathcal{U}$, there exists an ideal adversary $\mathcal{S}$ such that*

$$\mathrm{UnAuth}_{\pi,\mathcal{U}}() \stackrel{c}{\equiv} \mathrm{IDEAL}_{\mathcal{S}}().$$

We say that π *is secure in the* AM *if for any* AM *adversary* $\mathcal{A}$*, there exists an ideal adversary* $\mathcal{S}$ *such that*

$$\mathrm{Auth}_{\pi,\mathcal{A}}() \stackrel{c}{\equiv} \mathrm{IDEAL}_{\mathcal{S}}().$$

6.2 Our Protocol Encr-KE

In this subsection, we introduce our new key exchange protocol. Our construction only assumes the existence of trapdoor permutation (i.e., without a concrete hardness assumption such as DDH or RSA).

Let $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a public key cryptosystem, where given input 1^κ, $\mathcal{G}$ outputs a public/private key pair (ek, dk). The protocol is specified as follows. The graphic interpretation is presented in Figure 3.

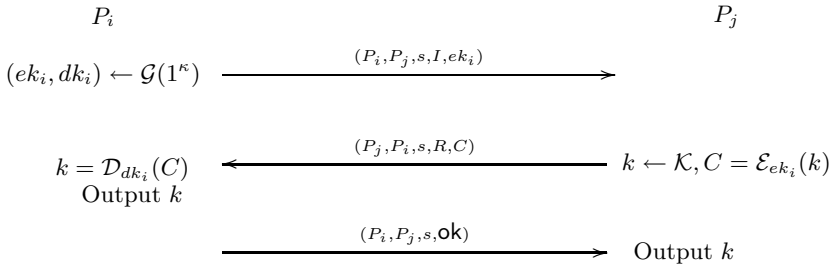


Fig. 3. Our Key Exchange Protocol Encr-KE in the AM
Note: Erasing the intermediate data is important but not presented in the figure

1. Initiator P_i takes a *temporary* key pair $(ek_i, dk_i) \leftarrow \mathcal{G}(1^\kappa)$ and sends (P_i, P_j, s, I, ek_i) to the responder P_j, where s is the session id and I means initiator.
2. P_j takes $k \leftarrow \mathcal{K}$ and computes $C = \mathcal{E}_{ek_i}(k)$, where $\mathcal{K}$ is the key domain. Then he sends (P_j, P_i, s, R, C) to P_i, where R means responder. Finally, P_j defines the session state to be (P_i, P_j, s, k) and *erases* all the other intermediate data (e.g., random bits in computing C).
3. P_i decrypts C and obtains k. If $k = \perp$, he rejects. Otherwise, he defines the session state to be (P_i, P_j, s, k) and *erases* all the other intermediate data (e.g., dk_i). Finally, he outputs (P_i, P_j, s, k) and sends $(P_i, P_j, s, \mathsf{ok})$ to P_j.
4. When P_j receives $(P_i, P_j, s, \mathsf{ok})$, he accepts and outputs (P_i, P_j, s, k).

In the following theorem, we show our Encr-KE is secure. The proof can be found in the full paper [14].

Theorem 3. *Assume* $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ *is a semantically secure public-key encryption scheme. Then* Encr-KE *is secure in the* AM*.*

Let EE-KE=E-Auth(Encr-KE) and SigE-KE=Sig-Auth(Encr-KE). Note IND-CCA2 secure public key encryption [17] exists if trapdoor permutation [11] exists, See [9, 11]. Unforgeable message authentication code and digital signature scheme both exist, assuming the existence of one-way function, See [12] for the former and [20] for the latter. Note that the existence of trapdoor permutation implies the existence of one-way function. From Theorem 1 and Theorem 2, we immediately have the following result.

Corollary 1. EE-KE *and* SigE-KE *are both secure in the* UM. *Furthermore, they can be implemented if the trapdoor permutation exists.*

References

1. M. Bellare, R. Canetti, and H. Krawczyk, a modular approach to the design and analysis of authentication and key exchange protocols, *STOC'98*, pp. 419-428.
2. M. Bellare and P.Rogaway, Entity authentication and key distribution, *Advances in Cryptology-CRYPTO'93*, D. Stinson (Ed.), LNCS 773, Springer-Verlag, 1993.
3. M. Bellare and P. Rogaway, Provably secure session key distribution - the three party case, *STOC'95*.
4. C. Boyd, W. Mao, K. Paterson, Key Agreement Using Statically Keyed Authenticators, *ACNS 2004*, M. Jakobsson *et al.* (Eds.), LNCS 3809, Spinger-Verlag, pp. 248-262, 2004.
5. R. Canetti and H. Krawczyk, analysis of key-exchange protocols and their use for building secure channels, *Advances in Cryptology-EUROCRYPT 2001*, B. Pfitzmann (Ed.), LNCS 2045, Springer-Verlag, pp. 453-474, 2001.
6. R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai, universally composable two-party and multi-party secure computation, *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pp. 494-503, May 19-21, 2002, Montreal, Quebec, Canada.
7. R. Canetti, universally composable security: a new paradigm for cryptographic protocols, *42th Symposium on Foundations of Computer Science, FOCS'01*, pp. 136-145, October 14-17, 2001, Las Vegas, Nevada, USA.
8. K. Choo, C. Boyd and Y. Hitchcock, Errors in Computational Complexity Proofs for Protocols, *Advances in Cryptology-ASIACRYPT'05*, B. Roy (Ed.), LNCS 3788, Springer-Verlag, pp. 624-643, 2005.
9. D. Dolev, C. Dwork and M. Naor, Non-malleable Cryptography, *STOC'91*. Full version appeared in *SIAM J. on Computing* 30(2), 2000, pp. 391-437.
10. W. Diffie and M. Hellman, new directions in cryptography, *IEEE Transactions on Information Theory*, Vol. 22, pp. 644-654, Nov. 1976.
11. O. Goldreich, *Foundations of Cryptography: Basic Applications,* Cambridge University Press, 2004.
12. O. Goldreich, S. Goldwasser amd S. Micali, On the cryptographic applications of random functions, *CRYPTO'84*, LNCS 263, pp. 276-288, 1985.
13. S. Jiang and G. Gong, "Password Based Key Exchange with Mutual Authentication", *Selected Area in Cryptography 2004*, H. Handschuh and A. Hasan (Eds.), LNCS 3357, pp. 271-283, 2005.
14. S. Jiang and G. Gong, Efficient Authenticators with Application to Key Exchange. Full paper of this work, available at `http://calliope.uwaterloo.ca/~jiangshq`

15. J. Katz, M. Yung, Scalable Protocols for Authenticated Group Key Exchange, *Advances in Cryptology-CRYPTO'03*, D. Boneh (Ed.), LNCS 2729, Springer-Verlag, pp. 110-125, 2003.
16. C. Rackoff, Some definitions, protocols, proofs about secure authentications, *IBM CASCON'92*.
17. C. Rackoff and D. Simon, non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack, *Advances in Cryptology-CRYPTO 1991*, J. Feigenbaum (Ed.), LNCS 576, Springer-Verlag, pp. 433-444, 1992.
18. M. Raimondo and R. Gennaro, New Approaches for Deniable Authentication, *IACR eprint 2005/046.* Available at http://eprint.iacr.org/2005/046.
19. R. Rivest, A. Shamir and L. Adleman, a method for obtaining digital signatures and public-key cryptosystems, *Communications of ACM*, Vol. 2, pp. 120-126, February 1978.
20. J. Rompel, One-way functions are necessary and sufficient for secure signatures, *STOC'90*, pp. 387-394.
21. V. Shoup, On Formal Models for Secure Key Exchange, *Theory of Cryptography Library*, 1999. Available at http://philby.ucsd.edu/cryptolib/1999.html.
22. Y. Tin, H. Vasanta, C. Boyd and J. Nieto, Protocols with Security Proofs for Mobile Applications, *ACISP 2004*, H. Wang *et al.* (Eds.), LNCS 3108, Springer-Verlag, pp. 358-369, 2004.

Benes and Butterfly Schemes Revisited

Jacques Patarin and Audrey Montreuil

Université de Versailles, 45 avenue des Etats-Unis,
78035 Versailles Cedex, France

Abstract. In [1], W. Aiello and R. Venkatesan have shown how to construct pseudo-random functions of $2n$ bits $\rightarrow 2n$ bits from pseudo-random functions of n bits $\rightarrow n$ bits. They claimed that their construction, called "Benes", reaches the optimal bound ($m \ll 2^n$) of security against adversaries with unlimited computing power but limited by m queries in an Adaptive Chosen Plaintext Attack (CPA-2). However a complete proof of this result is not given in [1] since one of the assertions of [1] is wrong. Due to this, the proof given in [1] is valid for most attacks, but not for all the possible Chosen Plaintext Attacks. In this paper we will in a way fix this problem since for all $\varepsilon > 0$, we will prove CPA-2 security when $m \ll 2^{n(1-\varepsilon)}$. However we will also see that the probability to distinguish Benes functions from random functions is sometime larger than the term in $\frac{m^2}{2^{2n}}$ given in [1]. One of the key idea in our proof will be to notice that, when $m \gg 2^{2n/3}$ and $m \ll 2^n$, for large number of variables linked with some critical equalities, the average number of solutions may be large (i.e. $\gg 1$) while, at the same time, the probability to have at least one such critical equalities is negligible (i.e. $\ll 1$).

Keywords: Pseudo-random functions, unconditional security, information-theoretic primitive, design of keyed hash functions. (An extended version of this paper is available from the authors).

1 Introduction

In [6], M. Luby and C. Rackoff have published their famous theorem: a 3-round Feistel scheme with three independent random round functions f_1, f_2, f_3 of n bits $\rightarrow n$ bits gives a pseudo-random function of $2n$ bits $\rightarrow 2n$ bits with security against all Adaptive Chosen Plaintext Attacks (CPA-2) when the number m of cleartext/ciphertext pairs chosen by the adversary satisfies $m \ll 2^{n/2}$ (even if the adversary has unbounded computing power). Since this paper [6], these constructions, or similar constructions, have inspired a considerable amount of research. In [11] a summary of existing works on this topic is given. The bound $m \ll 2^{n/2}$ is called the "birthday bound", i.e. it is about the square root of the optimal bound against an adversary with unbounded computing power. In [12] and [1] it was proved that for 3 or 4-round Feistel schemes this bound $m \ll 2^{n/2}$ is the best we can get. One direction of research is to design or study various schemes where we have a better proved security than the birthday bound. This is

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 92–116, 2006.
© Springer-Verlag Berlin Heidelberg 2006

what W. Aiello and R. Venkatesan have done in [1]: they have found a construction of locally random functions, called "Benes" (back-to-back Butterfly), where the optimal bound ($m \ll 2^n$) is obtained instead of the birthday bound. Here the functions are not permutations. Similarly, in [7] U. Maurer has found some other constructions of locally random functions (not permutations) where he can get as close as wanted to the optimal bound (i.e. $m \ll 2^{n(1-\varepsilon)}$ and for all $\varepsilon > 0$ he has a construction). In [11] the security of unbalanced Feistel schemes is studied and a security proof in $2^{n(1-\varepsilon)}$ is obtained, instead of $2^{n/2}$, but for much larger round functions (from $2n$ bits to ε bits, instead of n bits to n bits). However here this bound is basically again the birthday bound for this functions. In [15] J. Patarin obtained a security when $m \ll 2^n$ for 5-round Feistel scheme against all CPA-2. But even though Benes needs 8 pseudo-random functions application and 5-round Feistel needs 5 of them, Benes can be done in two parallel rounds while the Feistel will need 5 rounds, so in many cases Benes will still be actually 2.5 times better. In [2], Bellare, Goldreich and Krawczyk's construction is similar to the Butterfly construction and provide length-doubling for the input. The difference is that their construction is secure only against random queries and not adaptively chosen queries. Benes, in contrast, is actually a pseudo-random function, so the Bellare, Goldreich and Krawczyk's construction does not supercede Benes unless for all purposes a KPA (Known Plaintext Attack) secure random function is enough and length-doubling pseudo-random function is never needed.

In this paper we will study again the "Benes" schemes of [1]. First, we will notice that the proof of security given in [1] is valid for most chosen plaintext attacks, but is not valid for all chosen plaintext attacks. We will then in a way fix this problem. For known plaintext attacks (KPA), we will see that one Butterfly is enough to get security when $m \ll 2^n$ (Benes schemes and Butterfly schemes are defined in section 2). Then, for adaptive chosen plaintext attacks and for all $\varepsilon > 0$, we will prove CPA-2 security when $m \ll 2^{n(1-\varepsilon)}$ for sufficiently large n. However our proved security bound in this case will be larger than the term given in [1]. We will also mention what appears for a variant of Benes called "modified Benes", and we will give some examples of applications.

2 Notations

- $I_n = \{0,1\}^n$ is the set of the 2^n binary strings of length n.
- F_n is the set of all functions $f : I_n \to I_n$. Thus $|F_n| = 2^{n \cdot 2^n}$.
- For a, $b \in I_n$, $a \oplus b$ stands for bit by bit exclusive or of a and b.
- For a, $b \in I_n$, $a||b$ stands for the concatenation of a and b.
- For $a, b \in I_n$, we also denote by $[a, b]$ the concatenation $a||b$ of a and b.
- Given four functions from n bits to n bits, $f_1, \ldots, f_4$, we use them to define the **Butterfly transformation** (see [1]) from $2n$ bits to $2n$ bits. On input $[L_i, R_i]$, the output is given by $[X_i, Y_i]$, with:

$$X_i = f_1(L_i) \oplus f_2(R_i) \text{ and } Y_i = f_3(L_i) \oplus f_4(R_i).$$

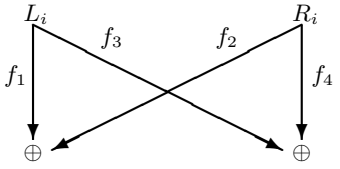


Fig. 1. Butterfly transformation

- Given eight functions from n bits to n bits, $f_1, \ldots, f_8$, we use them to define the **Benes transformation** (see [1]) (back-to-back Butterfly) as the composition of two Butterfly transformations. On input $[L_i, R_i]$, the output is given by $[S_i, T_i]$, with:

$$S_i = f_5(f_1(L_i) \oplus f_2(R_i)) \oplus f_6(f_3(L_i) \oplus f_4(R_i)) = f_5(X_i) \oplus f_6(Y_i)$$
$$T_i = f_7(f_1(L_i) \oplus f_2(R_i)) \oplus f_8(f_3(L_i) \oplus f_4(R_i)) = f_7(X_i) \oplus f_8(Y_i).$$

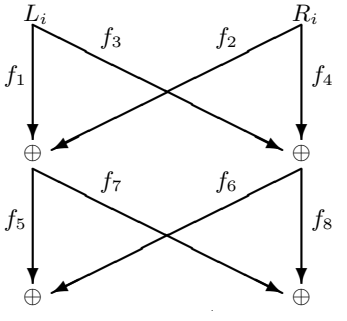


Fig. 2. Benes transformation (back-to-back Butterfly)

3 A Problem in the Proof of [1]

Definition 1. *We will say that we have "a circle in X, Y of length k" if we have k pairwise distinct indices such that $X_{i_1} = X_{i_2}$, $Y_{i_2} = Y_{i_3}$, $X_{i_3} = X_{i_4}$,..., $X_{i_{k-1}} = X_{i_k}$, $Y_{i_k} = Y_{i_1}$. We will say that we have "a circle in X, Y" if there is an even integer k, $k \geq 2$, such that we have a circle in X, Y of length k.*

Let $[L_1, R_1]$, $[L_2, R_2]$, $[L_3, R_3]$ and $[L_4, R_4]$ be four chosen inputs such that $L_1 = L_2$, $R_2 = R_3$, $L_3 = L_4$ and $R_4 = R_1$ (and $R_1 \neq R_2$ and $L_1 \neq L_3$). (Here we will say that we have "a circle in L, R" of length 4). Let p be the probability for these inputs to produce "a circle in X, Y" (or, in the language of [1], an "alternating cycle") after a Butterfly. In [1], page 318, it is claimed that "the probability that the top Butterfly produces an alternating cycle of length $2j$ is $\leq 2^{-2jn}$". So here this means $p \leq \frac{1}{2^{4n}}$. However we will see that $p \geq \frac{1}{2^{2n}}$. We have:

$$\begin{array}{ll} X_1 = f_1(L_1) \oplus f_2(R_1) & Y_1 = f_3(L_1) \oplus f_4(R_1) \\ X_2 = f_1(L_2) \oplus f_2(R_2) = f_1(L_1) \oplus f_2(R_2) & Y_2 = f_3(L_2) \oplus f_4(R_2) = f_3(L_1) \oplus f_4(R_2) \\ X_3 = f_1(L_3) \oplus f_2(R_3) = f_1(L_3) \oplus f_2(R_2) & Y_3 = f_3(L_3) \oplus f_4(R_3) = f_3(L_3) \oplus f_4(R_2) \\ X_4 = f_1(L_4) \oplus f_2(R_4) = f_1(L_3) \oplus f_2(R_1) & Y_4 = f_3(L_4) \oplus f_4(R_4) = f_3(L_3) \oplus f_4(R_1) \end{array}$$

First possible circle in X, Y. We will get the circle $X_1 = X_2$, $Y_2 = Y_3$, $X_3 = X_4$ and $Y_4 = Y_1$ if and only if $f_2(R_1) = f_2(R_2)$ and $f_3(L_1) = f_3(L_3)$ and the probability for this is exactly $\frac{1}{2^{2n}}$ (since $R_1 \neq R_2$ and $L_1 \neq L_3$).

Conclusion. The probability p to have a circle in X, Y of length 4 (i.e. the probability that the top Butterfly produces an alternating cycle of length 4 in the language of [1]) is $\geq \frac{1}{2^{2n}}$, so it is not $\leq \frac{1}{2^{4n}}$ as claimed in [1]. As we will see in this paper, this problem is not easily solved: a precise analysis will be needed in order to prove the security result $m \ll 2^{n(1-\varepsilon)}$ for all $\varepsilon > 0$.

Remark. It is possible to show that 6 different circles in X, Y are possible here, with a probability $\frac{1}{2^{2n}}$. So the probability p to have a circle in X, Y of length 4 will be between $\frac{1}{2^{2n}}$ and $\frac{6}{2^{2n}}$ here. One part of the work of this paper will be to see if $\frac{1}{2^{2n}}$ instead of $\frac{1}{2^{4n}}$ can create a problem or not, and another part of the work will be to evaluate the effect of the number of cases, 6 here, and the analysis will have to be done for circles of any length, not only length 4 as here.

4 One Butterfly: Proof of KPA Security When $m \ll 2^n$

Here we will prove KPA (Known Plaintext Attack) security by using the "coefficient H technique" of [14] (more precisely theorem 3.1 p.516 of [14]). Let $[L_i, R_i]$, $1 \leq i \leq m$, be the inputs. With one round of Butterfly, the outputs are $[X_i, Y_i]$ with:

$$\forall i, 1 \leq i \leq m, \begin{cases} X_i = f_1(L_i) \oplus f_2(R_i) \\ Y_i = f_3(L_i) \oplus f_4(R_i) \end{cases} \qquad (\#)$$

Now when the values L_i, R_i, X_i, Y_i are given, $1 \leq i \leq m$, let H be the number of f_1, f_2, f_3, f_4 of F_n such that we have (#). If we have no circle in L, R (cf. Definition 1) then each new equation (#) fixes f_1 (or f_2) and f_3 (or f_4) in a new point. So if we have no circle in L, R we will have exactly: $H = \frac{|F_n|^4}{2^{2nm}}$. Moreover, since we are in KPA, with m random cleartext/ciphertext pairs, we will now see that we can indeed assume, when $m \ll 2^n$, that there are no circle in L, R.

1. Circle on 2 indices i, j, $i \neq j$, are impossible because $L_i = L_j$ and $R_i = R_j$ implies that $i = j$.
2. Without loosing generality, we can study only circles on r indices, with r even ($L_1 = L_2$, $L_2 = L_3$ and $R_3 = R_1$ for example gives the circle $L_1 = L_3$, $R_3 = R_1$). We will first study the case of circle on 4 indices. Here, we have some pairwise distinct indices i, j, k, l such that: $L_i = L_j$, $L_k = L_l$, $R_i = R_k$ and $R_j = R_l$. The probability to have such a circle, when the L_α, R_α are randomly chosen, is $\leq \frac{m^4}{4 \cdot 2^{4n}}$ (Proof: we have here $\frac{m^4}{4}$ possible choices for

i, j, k, l since we can start the circle in i, j, k or l and each of the 4 equations have a probability $\frac{1}{2^n}$ to be satisfied if the L_α, R_α are randomly chosen).

3. More generally the probability to have a circle on r indices, when the L_i, R_i are randomly chosen, is $\leq \frac{m^r}{r \cdot 2^{nr}}$. So the probability p to have at least one circle in L, R, when the values L_α and R_α are randomly chosen satisfies: $p \leq \frac{1}{4} \sum_{i=2}^{\infty} \frac{m^{2i}}{2^{2in}} = \frac{1}{4} \cdot \frac{m^4}{2^{4n}} \cdot \frac{1}{1-\frac{m^2}{2^{2n}}}$.

Conclusion. By using theorem 3.1 p.516 of [14], we obtain (with $\alpha = 0$ and $\beta = \frac{1}{4} \cdot \frac{m^4}{2^{4n}} \cdot \frac{1}{1-\frac{m^2}{2^{2n}}}$): for all algorithm A taking m values $[L_i, R_i]$ on input, $|E(P_1 - P_1^*)| \leq \frac{1}{4} \cdot \frac{m^4}{2^{4n}} \cdot \frac{1}{1-\frac{m^2}{2^{2n}}}$ (where E is the expectancy on the $[L_i, R_i]$ randomly chosen).

So one Butterfly resist all KPA attacks when $m \ll 2^n$.

Remark 1. For Benes (i.e. two independent rounds of Butterfly), a similar KPA analysis would give security in $\mathcal{O}(\frac{m^2}{2^{2n}})$ instead of $\mathcal{O}(\frac{m^4}{2^{4n}})$ here for only one round of Butterfly. As we will see in appendix C for Benes, and more generally for λ rounds of Benes, $\lambda \geq 1$, the KPA security in $\mathcal{O}(\frac{m^2}{2^{2n}})$ is tight: there is an explicit ciphertext only attack in $\mathcal{O}(\frac{m^2}{2^{2n}})$. So for KPA security and for ciphertext only security one round of Butterfly is slightly better than two rounds (or λ rounds) when $m \ll 2^n$. This is due to the fact that for two rounds of Butterfly we can have $X_i = X_j$ and $Y_i = Y_j$ with $i < j$, and for one round we cannot have $L_i = L_j$ and $R_i = R_j$ with $i < j$ (two rounds of independent pseudo-random permutations cannot be less secure than one, but with pseudo-random functions, as here, it can be).

Remark 2. For CPA-1 (non-Adaptive Chosen Plaintext Attack) security however, unlike KPA security or ciphertext only attacks, Benes (i.e. two independent rounds of Butterfly) is clearly much better than one. We will see that when $m \ll 2^{n(1-\varepsilon)}$, $\varepsilon > 0$, Benes is secure against all CPA-2 (so also CPA-1). For one round of Butterfly there is a CPA-1 with $m = 4$: just choose two values L_i and L_j, $L_i \neq L_j$ and two values R_i and R_j, $R_i \neq R_j$, and ask for the outputs of $[L_i, R_i]$, $[L_j, R_j]$, $[L_i, R_j]$ and $[L_j, R_i]$. With Benes we will have $X_1 \oplus X_2 \oplus X_3 \oplus X_4 = 0$ and $Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_4 = 0$ with probability 1, and for random function this occurs with probability only $\frac{1}{2^{2n}}$.

5 Benes: First Results on Circles in X, Y

5.1 Circles in X, Y and CPA-2 Security

With Benes, we have:

$$\forall i,\ 1 \leq i \leq m,\ Benes(f_1, \ldots, f_8)[L_i, R_i] = [S_i, T_i] \Leftrightarrow \begin{cases} S_i = f_5(X_i) \oplus f_6(Y_i) \\ T_i = f_7(X_i) \oplus f_8(Y_i) \end{cases} \quad (1)$$

$$\text{with } \begin{cases} X_i = f_1(L_i) \oplus f_2(R_i) \\ Y_i = f_3(L_i) \oplus f_4(R_i) \end{cases}$$

When some L_i, R_i, S_i, T_i values are given, $1 \leq i \leq m$, let H be the number of $f_1, \ldots, f_8$ such that: $\forall i$, $1 \leq i \leq m$, $Benes(f_1, \ldots, f_8)[L_i, R_i] = [S_i, T_i]$.

Theorem 1. *If the X_i and Y_i values are such that there are no "circles in X, Y" then the number of f_5, f_6, f_7, f_8 solution of (1) is exactly $\frac{|F_n|^4}{2^{2nm}}$.*

Proof. Let $\mathcal{A}$ be a set of equations $S_i = f_5(X_i) \oplus f_6(Y_i)$. So, if the equations of $\mathcal{A}$ are not all independent, then we have a subset of λ equations in $\mathcal{A}$ where all the X_i values and all the Y_i values are identical two by two (Proof: if this property does not occur we can associate a new variable $f_5(X_\alpha)$ or $f_6(Y_\beta)$ to each new equation $S_i = f_5(X_i) \oplus f_6(Y_i)$, where α and β are found by looking the equations where we have $f_5(X_i)$ or $f_6(Y_i)$). We will have some indices $i_1, \ldots, i_k$, k even, where all the X_i values and all the Y_i values are identical two by two, $i \in \{i_1, \ldots, i_k\}$. There is at least one index $j_2 \in \{i_1, \ldots, i_k\}$, $j_2 \neq i_1$, such that $X_{i_1} = X_{j_2}$. There is at least one index j_3, $j_3 \neq j_2$, such that $Y_{j_2} = Y_{j_3}$. If $j_3 = i_1$ we have a circle in X, Y. If not, we can continue: there is at least one index j_4, $j_4 \neq j_3$, such that $X_{j_4} = X_{j_3}$. If $j_4 \in \{i_1, j_2\}$ we have a circle in X, Y. If not, we can continue. Like this we will obtain a circle in X, Y of length $< k$, or at the end we have an index j_k, $j_k \neq j_{k-1}$, such that $Y_{j_k} = Y_{i_1}$ and this gives a circle in X, Y of length k. So, if we have no circle in X, Y, then all the equations $S_i = f_5(X_i) \oplus f_6(Y_i)$ of (1) are independent, so we have exactly $\frac{|F_n|^2}{2^{nm}}$ functions f_5, f_6 solution. Similarly, we have exactly $\frac{|F_n|^2}{2^{nm}}$ functions f_7, f_8 solution of the equation $T_i = f_7(X_i) \oplus f_8(Y_i)$ of (1). So if we have no circle in X, Y we have exactly $\frac{|F_n|^4}{2^{2nm}}$ functions f_5, f_6, f_7, f_8 solution of (1), as claimed.

Let p be the probability to get at least one circle in X, Y in a CPA-2 (when f_1, f_2, f_3, f_4 are randomly chosen). From theorem 1, we have $H \geq (1-p)\frac{|F_n|^8}{2^{2nm}}$. So with theorem 3.2 p.517 of [14] (with $\alpha = 0$ and $\beta = p$) we get:

Theorem 2. *The probability to distinguish Benes functions from random functions of $2n$ bits $\to 2n$ bits in a CPA-2 is always $\leq p$, when $f_1, \ldots, f_8$ are randomly and independently chosen in F_n, and where p is the probability to have a circle in X, Y.*

Remark 1. This result was already in [1], written in the language of "alternating cycles". In fact, this result can be obtained directly, without using theorem 3.2 of [14]: when there are no circles in X, Y in each equation (1), we have a new variable $f_5(X_i)$ or $f_6(Y_i)$, and a new variable $f_7(X_i)$ or $f_8(Y_i)$, so if f_5, f_6, f_7, f_8 are random functions, the outputs S_i and T_i are perfectly random and independent from the previous S_j, T_j, $i < j$.

Remark 2. In this paper we will evaluate p. One difficulty is the fact that in a CPA-2 we cannot assume that the variables X_i and Y_i are random, so we cannot use the same proof as we did in section 4 for KPA security. For example if we choose $L_1 = L_2$, $L_3 = L_4$, $R_1 = R_3$ and $R_2 = R_4$, then we will have: $X_4 = X_1 \oplus X_2 \oplus X_3$ and $Y_4 = Y_1 \oplus Y_2 \oplus Y_3$, so the X_i (and Y_i) variables are not independent random variables.

Remark 3. In this paper we will analyze when p is small, since p small is a sufficient condition for CPA-2 security. We can notice, however, that this is not a necessary condition. Let us assume that we can, with a non negligible probability p generate A circles in X, Y with k variables. For each such circles we will have:

$$S_{i_1} \oplus \dots S_{i_k} = 0 \text{ and } T_{i_1} \oplus \dots T_{i_k} = 0. \tag{2}$$

For random functions, we will have about $\frac{m^k}{k!2^{2n}}$ indices $i_1, \dots, i_k$ such (2), with a standard deviation of about $\sqrt{\frac{m^k}{k!2^{2n}}}$ for this number. So even if p is not negligible, we may not be able to distinguish Benes functions from random functions if the probability to have $A \geq \sqrt{\frac{m^k}{k!2^{2n}}}$ is negligible (instead of $A \neq 0$). Now we will study the probability to have a circle in X, Y of length k. We will first study the cases $k = 2$ and $k = 4$ to better understand the proof of the general case.

5.2 Circles in X, Y with $k = 2$

Theorem 3. *The probability p_2 to have a circle in X, Y of length 2, when f_1, f_2, f_3, f_4 are randomly chosen in F_n satisfies: $p_2 \leq \frac{m(m-1)}{2 \cdot 2^{2n}}$. So p_2 is negligible when $m \ll 2^n$.*

Proof. Here we want $i < j$ such that $X_i = X_j$ and $Y_j = Y_i$, i.e. such that:

$$\begin{cases} f_1(L_i) \oplus f_2(R_i) = f_1(L_j) \oplus f_2(R_j) & (3) \\ f_3(L_i) \oplus f_4(R_i) = f_3(L_j) \oplus f_4(R_j) & (4) \end{cases}$$

First case: $R_i \neq R_j$. Then when f_1 is fixed, we have exactly $\frac{|F_n|}{2^n}$ functions f_2 such that (3) is satisfied, and when f_3 is fixed, we have exactly $\frac{|F_n|}{2^n}$ functions f_4 such that (4) is satisfied.

Second case: $R_i = R_j$. Then we have $L_i \neq L_j$ (since $i < j$ so $i \neq j$), so we have exactly $\frac{|F_n|}{2^n}$ functions f_1 such that (3) is satisfied and exactly $\frac{|F_n|}{2^n}$ functions f_3 such that (4) is satisfied.

Conclusion. Whatever L_i, L_j, R_i, R_j are, when i and j are fixed, we have exactly $\frac{|F_n|^4}{2^{2n}}$ functions f_1, f_2, f_3, f_4 such that (3) and (4) are satisfied. So, since we have $\frac{m(m-1)}{2}$ indices i, j, $i < j$, we have $p_2 \leq \frac{m(m-1)}{2 \cdot 2^{2n}}$, as claimed.

5.3 Circles in X, Y with $k = 4$

(As already said in section 4, without loosing generality we can study only circles with k even. $X_1 = X_2$, $X_2 = X_3$ and $Y_3 = Y_1$ for example gives the circle $X_1 = X_3$, $Y_3 = Y_1$ with $k = 2$).

Theorem 4. *The probability p_4 to have a circle in X, Y of length 4, when f_1, f_2, f_3, f_4 are randomly chosen in F_n satisfies: $p_4 \leq \frac{m^4}{4 \cdot 2^{4n}} + \frac{3m^2}{2^{2n}}$. So p_4 is negligible when $m \ll 2^n$.*

Proof. Here we want 4 pairwise distinct i, j, k, l such that: $X_i = X_j$, $Y_j = Y_k$, $X_k = X_l$ and $Y_l = Y_i$, i.e. such that:

$$(5)\begin{cases} f_1(L_i) \oplus f_2(R_i) = f_1(L_j) \oplus f_2(R_j) \\ f_1(L_k) \oplus f_2(R_k) = f_1(L_l) \oplus f_2(R_l) \\ f_3(L_j) \oplus f_4(R_j) = f_3(L_k) \oplus f_4(R_k) \\ f_3(L_i) \oplus f_4(R_i) = f_3(L_l) \oplus f_4(R_l) \end{cases}$$

For i, j, k, l, we have $\leq \frac{m(m-1)(m-2)(m-3)}{4}$ possibilities (since when i, j, k, l is a solution we can start the circle in X, Y with i, j, k or l. For example, we can decide that i is the smallest index: $i < j$ and $i < k$ and $i < l$).

- If in (5) the four equations are independent, the probability to obtain (5) will be $\leq \frac{m(m-1)(m-2)(m-3)}{4 \cdot 2^{4n}}$.
- Now in (5), as we will see, there are 6 cases where the four equations are not independent (they come from only two independent equations). For example (case 1), if $L_i = L_j$, $L_k = L_l$, $R_i = R_k$ and $R_j = R_l$, then

$$(5) \Leftrightarrow \begin{cases} f_2(R_i) = f_2(R_j) \\ f_3(L_i) = f_3(L_k) \end{cases}$$

  The equations number 1 and 3 of (5) are always independent , since f_3 and f_4 are randomly chosen independently from f_1 and f_2. However the equation number 2 can be equivalent with the equation number 1 if $L_i = L_j, L_k = L_l, R_i = R_k, R_j = R_l$ or $L_i = L_j, L_k = L_l, R_i = R_l, R_j = R_k$ or $L_i = L_k, L_j = L_l, R_i = R_j, R_k = R_l$ or $L_i = L_k, L_j = L_l, R_i = R_l, R_j = R_k$ or $L_i = L_l, L_j = L_k, R_i = R_j, R_k = R_l$ or $L_i = L_l, L_j = L_k, R_i = R_k, R_j = R_l$.

  These 6 cases are also the conditions for the equations number 5 and 6 to be equivalent. However, in all of these 6 cases, 2 indices are fixed when two other indices are given. For example with case 1, if i and l are given, then j and k are fixed, since $L_j = L_i$ and $R_j = R_l$ (this fixes at most one j), and since $L_k = L_l$ and $R_k = R_i$ (this fixes at most one k).

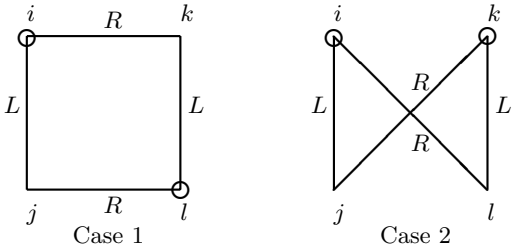


Fig. 3. Case 1: if i and l are given, then j and k are fixed. As $i < l$, here we have $\leq \frac{m(m-1)}{2}$ possibilities for i, l. Case 2: if i and k are given, then j and l are fixed. As $i < k$, here we have $\leq \frac{m(m-1)}{2}$ possibilities for i, k too. (And we do the same thing for the other cases.)

Conclusion. $p_4 \leq \frac{m(m-1)(m-2)(m-3)}{4 \cdot 2^{4n}} + 6 \cdot \frac{m(m-1)}{2 \cdot 2^{2n}}$, so $p_4 \leq \frac{m^4}{4 \cdot 2^{4n}} + \frac{3m^2}{2^{2n}}$, as claimed.

5.4 Circles in X, Y, the General Case

Theorem 5. *Let k be an even integer. The probability p_k to have a circle in X, Y of length k, when f_1, f_2, f_3, f_4 are randomly chosen in F_n satisfies: $p_k \leq k^{2k} \frac{m^2}{2^{2n}}$.*

Proof. We have a circle of length k in X, Y if and only if there are some pairwise distinct indices $i_1, \ldots, i_k$ such that $X_{i_1} = X_{i_2}$, $Y_{i_2} = Y_{i_3}$, $X_{i_3} = X_{i_4}$, ..., $Y_{i_k} = Y_{i_1}$, i.e. such that:

$$(8) \begin{cases} f_1(L_{i_1}) \oplus f_2(R_{i_1}) = f_1(L_{i_2}) \oplus f_2(R_{i_2}) \\ f_1(L_{i_3}) \oplus f_2(R_{i_3}) = f_1(L_{i_4}) \oplus f_2(R_{i_4}) \\ \vdots \\ f_1(L_{i_{k-1}}) \oplus f_2(R_{i_{k-1}}) = f_1(L_{i_k}) \oplus f_2(R_{i_k}) \end{cases} \text{ and } (9) \begin{cases} f_3(L_{i_2}) \oplus f_4(R_{i_2}) = f_3(L_{i_3}) \oplus f_4(R_{i_3}) \\ f_3(L_{i_4}) \oplus f_4(R_{i_4}) = f_3(L_{i_5}) \oplus f_4(R_{i_5}) \\ \vdots \\ f_3(L_{i_k}) \oplus f_4(R_{i_k}) = f_3(L_{i_1}) \oplus f_4(R_{i_1}) \end{cases}$$

For $\{i_1, \ldots, i_k\}$ we have $\leq \frac{m^k}{k}$ possibilities (since we can start the circle with i_1, or i_2 ,..., or i_k). If in (8) and (9) the $\frac{k}{2} + \frac{k}{2} = k$ equations are independent, the probability to obtain (8) and (9) will be $\leq \frac{m^k}{k \cdot 2^{kn}}$.

To study the general case, where the equations may be dependent, we will introduce the equations of the circle in X, Y one by one.

- The first equation is: $X_{i_1} = X_{i_2}$. Here we have $m(m-1)$ possible choices for i_1 and i_2, and when i_1 and i_2 are given, the probability to have $X_{i_1} = X_{i_2}$ is exactly $\frac{1}{2^n}$.
- The second equation is: $Y_{i_2} = Y_{i_3}$. Here, when i_1 and i_2 are given, we have $\leq m-2$ possible choices for i_3, and this equation $Y_{i_2} = Y_{i_3}$ is independent from the equation $X_{i_1} = X_{i_2}$ (since with Y we use f_3 and f_4 and with X we use f_1 and f_2), so the probability to have $Y_{i_2} = Y_{i_3}$ when $X_{i_1} = X_{i_2}$ and when i_1, i_2 and i_3 are given is exactly $\frac{1}{2^n}$.
- The equation number 3 is: $X_{i_3} = X_{i_4}$. Here, there are two cases.

Case 1. $X_{i_3} = X_{i_4}$ is independent from $X_{i_1} = X_{i_2}$. Then for i_4 we have $m-3$ possible choices (when i_1, i_2 and i_3 are given), and the probability to have $X_{i_3} = X_{i_4}$ when $X_{i_1} = X_{i_2}$ and $Y_{i_2} = Y_{i_3}$ is $\frac{1}{2^n}$.

Case 2. $X_{i_3} = X_{i_4}$ is dependent from $X_{i_1} = X_{i_2}$. Then the indices i_1, i_2, i_3 and i_4 can be associated two by two with equalities in R, and two by two with equalities in L (for example $L_{i_4} = L_{i_3}$, $L_{i_1} = L_{i_2}$, $R_{i_3} = R_{i_2}$, and $R_{i_4} = R_{i_1}$). So we have $L_{i_4} = L_{i_\alpha}$, $\alpha = 1, 2$ or 3, and $R_{i_4} = R_{i_\beta}$, $\beta = 1, 2$ or 3, and $\alpha \neq \beta$ (since $\alpha < 4$, and since $L_{i_4} = L_{i_\alpha}$ and $R_{i_4} = R_{i_\beta}$ would imply $i_4 = i_\alpha$ and $\alpha = 4$). So in this case 2, i_4 is fixed when i_1, i_2 and i_3 are fixed, and when the equalities in L and R are given. For these equalities in L and R we have here $\leq 3 \times 2 = 6$ possibilities (3 for α and 2 for β when α is fixed).

- For equation number μ, $3 \leq \mu < k$, we have two cases.

Case 1. This equation number μ is independent from the other equations. Then for $i_{\mu+1}$ we have $m - \mu$ possible choices (when i_1, i_2, ..., i_μ are given), and the probability to have this equation number μ when the other equations are satisfied is $\frac{1}{2^n}$.

Case 2. This equation number μ is dependent from the other equations. Then $\exists \alpha, \beta$, $\alpha \neq \beta$, $\alpha \leq \mu$, $\beta \leq \mu$, such that $L_{i_{\mu+1}} = L_{i_\alpha}$ and $R_{i_{\mu+1}} = R_{i_\beta}$. So (since the $[L_i, R_i]$, $1 \leq i \leq m$ are pairwise distinct), $i_{\mu+1}$ is fixed when i_1, i_2, ..., i_μ are fixed, and when the equalities in L and R are given. For α and β we have $\leq \mu(\mu - 1)$ possibilities.

If equation number μ is the first in this case 2. If μ is the first integer such that equation number μ is in this case 2 (i.e. such that equation number μ is dependent from the other equations) then we will see now that not only one but at least two indices can be fixed from the other indices.

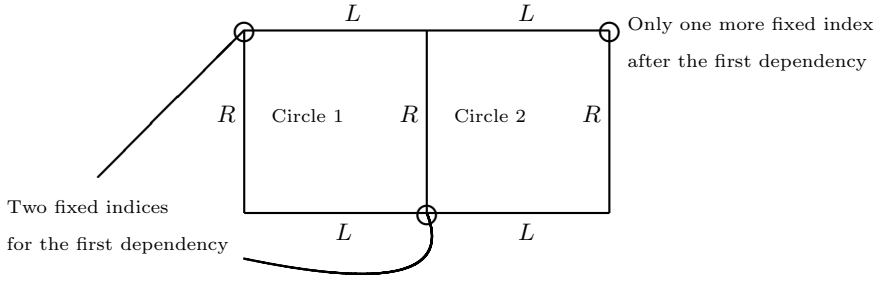


Fig. 4. Example: after the first time one more dependency can fix only one more index

Proof: since equation number μ is dependent from the previous equations, there is a subset S of the equations such that all the equations in S have a number $\leq \mu$, such that all the L_i variables in the equations of S can be associated two by two with equalities, and such that all the R_i variables in the equations of S can be associated two by two with equalities, and such that equation number μ is in S. So we have an index α, $\alpha \leq \mu$ such that $L_{i_{\mu+1}} = L_{i_\alpha}$ and an index β, $\beta \leq \mu$, $\alpha \neq \beta$, such that $R_{i_{\mu+1}} = R_{i_\beta}$, as said above, but we also have an index γ, $\gamma \leq \mu$, $\gamma \neq \alpha$ and an index δ, $\delta \leq \mu$, $\delta \neq \gamma$ such that $R_{i_\alpha} = R_{i_\gamma}$ and $L_{i_\gamma} = L_{i_\delta}$ (see figure 5). Here we see that $i_{\mu+1}$ **and** i_γ can be fixed from the other indices $\leq \mu$ (since $L_{i_\gamma} = L_{i_\delta}$ and $R_{i_\gamma} = R_{i_\alpha}$, and $L_{i_{\mu+1}} = L_{i_\alpha}$ and $R_{i_{\mu+1}} = R_{i_\beta}$) and here for $\alpha, \beta, \gamma, \delta$,

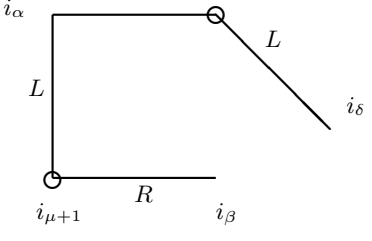


Fig. 5. First time dependency: at least two indices can be fixed

Remark. Alternatively it is also possible to show that since equation number μ is dependent from the previous equations, we will have one, or more than

one circles in L, R (there is an index α_1 such that $L_{i_{\mu+1}} = L_{i_{\alpha_1}}$, and an index $\alpha_2 \neq \alpha_1$ such that $R_{i_{\alpha_1}} = R_{i_{\alpha_2}}$, and an index $\alpha_3 \neq \alpha_2$ such that $L_{i_{\alpha_2}} = L_{i_{\alpha_3}}$, etc. and since the number of indices is finite, we get like this a circle in L, R. If not all the indices of the dependencies are covered, we can continue to get some other circles in L, R). Since in a circle in L, R, 50% of the indices can be fixed from the other indices, and since a circle in L, R has a length ≥ 4 (since $L_i = L_j$ and $R_i = R_j$ imply $i = j$), we see again that at least 2 indices will be fixed with the first dependency. For the second dependency it may occur, however, that only one new index is fixed. For example if $L_1 = L_2 = L_3$, $L_4 = L_5 = L_6$, $R_1 = R_4$, $R_2 = R_5$ and $R_3 = R_6$, then $X_1 = X_4$ implies $X_2 = X_5$ (this fixes 2 indices) and $X_3 = X_6$ (this fixes only one more index).

• For equation number k, the last equation, it can be dependent or not from the previous equations, but here, unlike before, we do not introduce a new index (since we have a circle in the indices).

Conclusion for p_k. If we have no dependencies in the equations, the probability to have all the equations is $\leq \frac{m^k}{k \cdot 2^{nk}}$. If we have a dependency, for equations number 1 and 2 and for indices i_1, i_2, i_3 we have a probability $\leq \frac{m^3}{2^{2n}}$. Then each new equation different from the last one, say equation number μ, $\mu < k$, either introduces a new index $i_{\mu+1}$ with a condition in $\frac{1}{2^n}$ (it gives a term $\leq \frac{m}{2^n}$) or we have less than $\mu(\mu-1)$ possibilities for this index $i_{\mu+1}$. Moreover, the first time that we have a dependency, say with equation number μ, $\mu < k$, then we can fix two indices, $i_{\mu+1}$ and i_γ for $\gamma \leq \mu$, from the other indices $i_j, j \leq \mu$, and after less than $\mu(\mu-1)(\mu-1)^2$ possibilities for the equalities in L and R these two indices will be fixed. So we get: $p_k \leq \frac{m^k}{k \cdot 2^{nk}} + \left[\frac{m(m-1)}{2^n} \cdot \frac{(m-2)}{2^n} \cdot \left(\frac{(k-1)^2}{m} \cdot \prod_{\mu=3}^{k-1} (\mu(\mu-1)) + \frac{m-\mu}{2^n} \right) \right]$. (Note: the term $\frac{(k-1)^2}{m}$ comes from the second fixed index when we get the first dependency). $p_k \leq \frac{m^k}{k \cdot 2^{nk}} + \frac{m^3}{2^{2n}} \frac{(k-1)^2}{m} \cdot (2 \cdot 3 + \frac{m}{2^n}) \cdot (4 \cdot 5 + \frac{m}{2^n}) \ldots ((k-2)(k-1) + \frac{m}{2^n})$. So if $m \leq 2^n$, $p_k \leq \frac{m^k}{k \cdot 2^{nk}} + \frac{m^2 (k-1)^2}{2^{2n}} \cdot (2 \cdot 3 + 1) \cdot (4 \cdot 5 + 1) \ldots ((k-2)(k-1) + 1)$ so $p_k \leq \frac{m^k}{k \cdot 2^{nk}} + \frac{m^2}{2^{2n}} \cdot (k-1)^2 (k^2)^{k-2}$ so since $\frac{m^k}{k \cdot 2^{nk}} \leq \frac{m^2}{k \cdot 2^{2n}}$ if $m \leq 2^n$ and $k \geq 2$, we get $p_k \leq k^{2k} \cdot \frac{m^2}{2^{2n}}$ as claimed.

Remark. In appendix F we will show a slightly different way to prove this theorem 5 (by looking differently at all the possible equalities in L and R). In appendix F we will see that instead of the coefficient k^{2k}, we can get a coefficient near k^k. We can notice however that this coefficient can really be very large. For example, if we start from a fixed circle of length k in L, R:

- For equalities in X, Y such that we have a circle of length k in X, Y, we have potentially $(k-1)!$ possibilities.
- For equalities in X, Y such that all the indices can be associated two by two with equalities in X, and associated two by two with equalities in Y, we have potentially $(3 \cdot 5 \cdot 7 \ldots (k-1))^2$ possibilities (this is $\leq (k^{k/2})^2$).

6 Benes: Proof of CPA-2 Security When $m \ll 2^{n(1-\varepsilon)}$

In order to introduce the complexity of the proofs progressively, we will first prove security when $m \ll 2^{n/2}$, when $m \ll 2^{2n/3}$, when $m \ll 2^{3n/4}$ and finally when $m \ll 2^{n(1-\varepsilon)}$, for all $\varepsilon > 0$.

6.1 Security When $m \ll 2^{n/2}$

When f_1, f_2, f_3, f_4 are randomly and independently chosen in F_n, the probability q_1 to have i, j, $1 \leq i < j \leq m$, such that $X_i = X_j$ satisfies $q_1 \leq \frac{m(m-1)}{2 \cdot 2^n}$. So the probability p to have a circle in X, Y (of any length) satisfies $p \leq q_1 \leq \frac{m(m-1)}{2 \cdot 2^n}$ (since in any circle in X, Y we will have $i < j$ such that $X_i = X_j$). So from theorem 2 we get:

Theorem 6. *The probability to distinguish Benes functions from random functions of $2n$ bits $\rightarrow$ $2n$ bits in any CPA-2 with m chosen messages is always $\leq \frac{m(m-1)}{2 \cdot 2^n}$ (when $f_1, \ldots, f_8$ are randomly and independently chosen in F_n). This gives security when $m^2 \ll 2^n$, i.e. when $m \ll 2^{n/2}$.*

6.2 Security When $m \ll 2^{2n/3}$

When f_1, f_2, f_3, f_4 are randomly and independently chosen in F_n, the probability q_2 to have 3 pairwise distinct indices i, j, k, such that $X_i = X_j$ and $Y_j = Y_k$ satisfies $q_2 \leq \frac{m(m-1)(m-2)}{2^{2n}}$ (Proof: when i, j, k are fixed $X_i = X_j$ is a condition with probability $\frac{1}{2^n}$ on f_1 and f_2 and $Y_j = Y_k$ is a condition with probability $\frac{1}{2^n}$ on f_3 and f_4, and f_1, f_2, f_3, f_4 are independently chosen). So the probability p to have a circle in X, Y (of any length) satisfies $p \leq q_2 \leq \frac{m^3}{2^{2n}}$. So from theorem 2 we get:

Theorem 7. *The probability to distinguish Benes functions from random functions of $2n$ bits $\rightarrow$ $2n$ bits in any CPA-2 with m chosen messages is always $\leq \frac{m^3}{2^{2n}}$ (when $f_1, \ldots, f_8$ are randomly and independently chosen in F_n).*

6.3 Security When $m \ll 2^{3n/4}$

Theorem 8. *When f_1, f_2, f_3, f_4 are randomly and independently chosen in F_n, the probability q_3 to have 4 pairwise distinct indices i, j, k, l, such that $X_i = X_j$, $Y_j = Y_k$, $X_k = X_l$ satisfies $q_3 \leq \frac{m^4}{2^{3n}} + \frac{3m^2}{2^{2n}}$.*

Proof. If the two equations in X are independent, the probability to obtain these 3 independent equations on 4 indices is $\leq \frac{m^4}{2^{3n}}$. If $f_1(L_i) \oplus f_2(R_i) = f_1(L_j) \oplus f_2(R_j)$ and $f_1(L_k) \oplus f_2(R_k) = f_1(L_l) \oplus f_2(R_l)$ are dependent, then the values L_i, L_j, L_k, L_l can be linked two by two with equalities, and the valuesR_i, R_j, R_k, R_l can be linked two by two with equalities (for example: $L_i = L_j$, $L_k = L_l$, $R_i = R_k$ and $R_j = R_l$). These equations in L, R can be written as some circles of equalities in L, R (in the example above we have the circle: $L_i = L_j$, $R_j = R_l$, $L_l = L_k$, $R_k = R_i$). (Remark: here, with 2 equations in X, we can have only one circle, of length 4, since circles in L, R of length 2 cannot exist since

$L_i = L_j$ and $R_i = R_j$ implies $i = j$). So two indices will be fixed from the two other indices from these equations in L, R of the circle (in the example above, j and k are fixed when i and l are given, since $L_j = L_i$, $R_j = R_l$, $L_k = L_l$ and $R_k = R_i$). Moreover, for the equalities in L and R we have here ≤ 6 possibilities (for α such that $R_k = R_\alpha$ we can take $\alpha = i, j$ or l, then for β such that $L_k = L_\beta$ we can take $\beta = i, j$ or l and we need $\alpha \neq \beta$). Conclusion: $q_3 \leq \frac{m^4}{2^{3n}} + \frac{3m^2}{2^{2n}}$, as claimed.

Theorem 9. *The probability p to have a circle in X, Y (of any length) satisfies $p \leq \frac{m^2}{2 \cdot 2^{2n}} + \frac{m^4}{2^{3n}} + \frac{3m^2}{2^{2n}}$.*

Proof. The probability to have a circle in X, Y is $\leq$ [the probability to have a circle in X, Y of length 2 + the probability to have a circle in X, Y of length > 2].

We have seen in section 5.2 that the probability to have a circle in X, Y of length 2 is $\leq \frac{m^2}{2 \cdot 2^{2n}}$. Circles in X, Y of length > 2 always have 4 pairwise distinct indices i, j, k, l such that $X_i = X_j$, $Y_j = Y_k$ and $X_k = X_l$. So from theorem 8, we have $p \leq \frac{m^2}{2 \cdot 2^{2n}} + (\frac{m^4}{2^{3n}} + \frac{3m^2}{2^{2n}})$, as claimed.

Now from theorem 2 we get immediately the CPA-2 security of Benes schemes when $m \ll 2^{3n/4}$:

Theorem 10. *The probability to distinguish Benes functions from random functions of $2n$ bits $\rightarrow 2n$ bits in any CPA-2 with m chosen messages is always $\leq \frac{m^4}{2^{3n}}$ (when $f_1, \ldots, f_8$ are randomly and independently chosen in F_n).*

6.4 Security When $m \ll 2^{n(1-\varepsilon)}$

The probability to have a circle in X, Y is $\leq$ [the probability to have a circle in X, Y of length 2 + the probability to have a circle in X, Y of length 4 + the probability to have a circle in X, Y of length between 6 and k + the probability to have a circle in X, Y of length $> k$].

We have here an infinite sum. The trick to have just to calculate a finite sum is to fix k and to use the fact that the probability to have a circle in X, Y of length $> k$ is $\leq$ the probability to have a line in X, Y of length k. We give here the definition of a line in X, Y:

Definition 2. *If k is odd, we will say that we have "a line in X, Y of length k" if we have $k + 1$ pairwise distinct indices such that $X_{i_1} = X_{i_2}$, $Y_{i_2} = Y_{i_3}$, $X_{i_3} = X_{i_4}$, ..., $Y_{i_{k-1}} = Y_{i_k}$, $X_{i_k} = X_{i_{k+1}}$. Similarly, if k is even, we will say that we have "a line in X, Y of length k" if we have $k + 1$ pairwise distinct indices such that $X_{i_1} = X_{i_2}$, $Y_{i_2} = Y_{i_3}$, $X_{i_3} = X_{i_4}$, ..., $X_{i_{k-1}} = X_{i_k}$, $Y_{i_k} = Y_{i_{k+1}}$. So in a line in X, Y we have $k + 1$ indices, and k equations, in X or in Y, and these equations can be written "in a line" from the indices.*

Let k be an integer, $k \geq 1$.

Theorem 11. *When f_1, f_2, f_3, f_4 are randomly and independently chosen in F_n, the probability q_k to have a line in X, Y of length k satisfies $q_k \leq \frac{m^{k+1}}{2^{nk}} + \frac{k^{2k} m^2}{2^{2n}}$, when $k \geq 4$ (or $q_k \leq \frac{m^{k+1}}{2^{nk}} + \frac{k^{2k} m^4}{2^{4n}}$, when $k \geq 6$).*

Remark. This decrease can be improved in different ways, for example if $k \geq 6$ we can get $\frac{k^{2k}m^4}{2^{4n}}$ instead of $\frac{k^{2k}m^2}{2^{2n}}$, and the value k^{2k} can be improved (see appendix F). However, this result will be enough to get security in $2^{n(1-\varepsilon)}$ for all fixed $\varepsilon > 0$.

Proof of theorem 11. The proof is exactly the same as the proof of theorem 5 of section 5.4. The proof is even slightly simpler here, since the last equation does not have to be treated differently from the other equations (each equation in X or Y introduces a new index and a new equation).

Theorem 12. *For all fixed integer k, $k \geq 1$, when $f_1, \ldots, f_8$ are randomly and independently chosen in F_n, the probability p to have a circle in X, Y (of any length) satisfies $p \leq \frac{m^2}{2 \cdot 2^{2n}} + \left(\frac{m^4}{4 \cdot 2^{4n}} + \frac{3m^2}{2^{2n}}\right) + \sum_{\lambda=6}^{k} \frac{\lambda^{2\lambda} m^2}{2^{2n}} + \left(\frac{m^{k+1}}{2^{nk}} + \frac{k^{2k}m^2}{2^{2n}}\right)$.*

Proof. If we have a circle in X, Y, then we have a circle of length 2 (probability $\leq \frac{m^2}{2 \cdot 2^{2n}}$ from theorem 3), or a circle of length 4 (probability $\leq \frac{m^4}{4 \cdot 2^{4n}} + \frac{3m^2}{2^{2n}}$ from theorem 4), or we have a circle of length between 6 and k (probability $\leq \sum_{\lambda=6}^{k} \frac{\lambda^{2\lambda} m^2}{2^{2n}}$ from theorem 5), or we have a line in X, Y of length k (probability $\leq \frac{m^{k+1}}{2^{nk}} + \frac{k^{2k}m^2}{2^{2n}}$ from theorem 11).

Theorem 13. *For all fixed integer k, $k \geq 1$, the probability p to distinguish Benes functions from random functions of $2n$ bits $\rightarrow 2n$ bits in any CPA-2 with m chosen messages always satisfies $p \leq (3 + \frac{1}{2} + \sum_{\lambda=6}^{k} \lambda^{2\lambda} + k^{2k}) \frac{m^2}{2^{2n}} + \frac{1}{4} \frac{m^4}{2^{4n}} + \frac{m^{k+1}}{2^{nk}}$ (when $f_1, \ldots, f_8$ are randomly and independently chosen in F_n). So if k is fixed, $n \rightarrow \infty$ and $m^{k+1} \ll 2^{nk}$, then p will be $\ll 1$. So, for any k, for sufficiently large n, $m \ll 2^{nk/(k+1)}$ gives CPA-2 security for Benes. So, for any $\varepsilon > 0$, for sufficiently large n, $m \ll 2^{n(1-\varepsilon)}$ gives CPA-2 security for Benes.*

Proof. Theorem 13 follows immediately from theorem 12 and theorem 2.

7 Examples of Applications

Keyed hash functions. In [1] it is explained how Benes schemes can be used for the design of keyed hash function. From an input $[L_i, R_i]$ of $2n$ bits, the Benes transformation gives a keyed hash function of n bits (the key is the functions f_1, f_2, f_3, f_4). By combining this construction with the scheme of [3] it is also possible to obtain a keyed hash function where the inputs can have any length and the outputs will have n bits.

Information-theoretic application. Let us first describe a problem. Alice wants to send many encrypted messages to Bob (for example 1 million messages) with a stream cipher. Charlie is an adversary of Alice and Bob. He has "dynamic" access to m messages. "Dynamic" means that he can choose a message, get the corresponding ciphertext and then adaptively choose the next message and get the corresponding ciphertext, etc., m times. Moreover, Charlie has unlimited computing power: he has access to only m messages but he can perform an infinite number of computations. Alice and Bob know a secret K. We want to

design an encryption function that will "resist" Charlie's attacks. "Resist" means that, even if Charlie uses the m cleartext/ciphertext pairs he has access to, he has no practical information on the other cleartexts. Benes functions offer a solution for these problems with a length of the key (i.e. the functions f_1, f_2, f_3, f_4) not far from the optimal. The idea is to use the Benes functions to create a stream cipher like this: the message number i, says m_i, will be encrypted as $c_i = Benes(i) \oplus m_i$ (since this is a stream cipher, we do not need here Benes to be invertible). Here i can be any value between 0 and $2^{2n} - 1$, so we can encrypt $N = 2^{2n}$ messages. So here Charlie can choose m values i between 0 and $2^{2n} - 1$ and he will get $Benes(i)$. The length of the key is here $K = 4 \cdot n \cdot 2^n$ bits and the scheme will be secure as long as $m \ll 2^{n(1-\varepsilon)}$ for any fixed ε and sufficiently large n.

8 Conclusion

William Aiello and Ramarathnam Venkatesan did a wonderful work by pointing out the great potentialities of the Benes schemes and by giving some very important parts of a possible proof. Unfortunately, the complete proof of security when $m \ll 2^n$ for CPA-2 is more complex than what they published in [1] due to some possible attacks with circles in L, R. However, a careful analysis of these attacks shows that $\forall \varepsilon > 0$, for large values n the probability p to distinguish Benes schemes from truly random functions satisfies for all CPA-2: $p \ll 1$ when $m \ll 2^{n(1-\varepsilon)}$ (but we do not have always $p \leq \frac{m^2}{2^{2n}}$ as claimed in [1]), so the final security is in a way similar, at least for large n. One of the key point in our proof was to notice the fact that the expectancy of the number of circles in X, Y may be large (when $m \gg 2^{2n/3}$) while the probability to have at least one such circle is generally negligible (when $m \ll 2^n$). The security bound in $m \ll 2^n$ is also the security bound for the complexity, since we have shown in this paper how to distinguish Benes (and more generally λ rounds of independent Benes schemes for all integer $\lambda \geq 1$) from random functions with a cyphertext only attack of about 2^n messages with about 2^n computations (for Feistel schemes we do not have a similar result).

References

1. W. Aiello and R. Venkatesan, *Foiling Birthday Attacks in Lenght-Doubling Transformations - Benes: a non-reversible alternative to Feistel*, Eurocrypt '96, LNCS 1070, pp. 307–320, Springer.
2. M. Bellare, O. Goldreich and H. Krawczyk *Stateless evaluation of pseudorandom functions: Security beyond the birthday barrier*, Crypto '99, LNCS 1666, Springer-Verlag.
3. I. Damgård, *Design Principles of Hash Functions*, Crypto '89, Springer-Verlag.
4. O. Goldreich, S. Goldwasser and S. Micali, *How to Construct Random Functions*, *JACM*, 33, pp 792–807, 1986.
5. M. Luby. *Pseudorandomness and Its Cryptographic Applications, Princeton Computer Science Notes, Princeton University Press.*
6. M. Luby and C. Rackoff. *How to construct pseudorandom permutations from pseudorandom functions, SIAM Journal on Computing*, vol. 17, nb 2, pp. 373–386, April 1988.

7. U. Maurer. *A simplified and Generalized Treatment of Luby-Rackoff Pseudorandom Permutation Generators, Eurocrypt '92*, Lecture Notes in Computer Science 658, pp. 239–255, Springer-Verlag.
8. U. Maurer. *Information-Theoretic Cryptography, Crypto '99*, Lecture Notes in Computer Science 1666, pp. 47–64, Springer.
9. U. Maurer. *Indistinguishability of Random Systems, Eurocrypt '02*, Lecture Notes in Computer Science 2332, pp. 110–132, Springer.
10. U. Maurer and K. Pietrzak. *The security of Many-Round Luby-Rackoff Pseudo-Random Permutations,Eurocrypt '03*, pp. 544–561, Springer.
11. M. Naor and O. Reingold, *On the construction of pseudo-random permutations: Luby-Rackoff revisited, Journal of Cryptology*, vol. 12, 1999, pp. 29–66. Extended abstract was published in *Proc. 29th ACM Symp. on Theory of Computing*, 1997, pp. 189–199.
12. J. Patarin, *New results on pseudo-random permutation generators based on the DES scheme, Crypto '91*, Lecture Notes in Computer Science 576, pp. 301–312, Springer-Verlag.
13. J. Patarin, *Improved security bounds for pseudorandom permutations*, 4th ACM Conference on Computer and Communications Security, April 1-4, 1997, Zurich, ACM Press, pp. 142–150.
14. J. Patarin, *Luby-Rackoff: 7 rounds are Enough for $2^{n(1-\varepsilon)}$ Security, Crypto '03*, Lecture Notes in Computer Science 2729, pp. 513–529, Springer.
15. J. Patarin, *Security of Random Feistel Scemes with 5 or more rounds, Crypto '04*, Lecture Notes in Computer Science 3152, pp. 106–122, Springer.

Appendices

A Summary of the Security Results for Benes and Butterfly Schemes

We summarize the security results on Benes and Butterfly schemes on figure 3 and figure 4 below. We also compare them with Feistel schemes. For large values of n the minimum number of computations is always $\geq m$, where m is the number of messages used in the attack.

	Random Ciphertext only attack	KPA	CPA-2
One round of Butterfly	2^n	2^n	4
Benes	2^n	2^n	$\geq 2^{n(1-\varepsilon)}$
λ rounds of Benes $\lambda \geq 1$	2^n	2^n	$\geq 2^{n(1-\varepsilon)}$
3 rounds of Feistel	Does not exist *	$2^{n/2}$ **	$2^{n/2}$ **
λ rounds of Feistel $\lambda \geq 6$	Does not exist *	2^n **	2^n **

Fig. 6. Minimum number m of queries needed to distinguish the schemes from random functions of $2n$ bits $\rightarrow$ $2n$ bits (or from random permutations for Feistel schemes), even if we have access to unbounded computing power. For simplicity we denote 2^α for $\mathcal{O}(2^\alpha)$, i.e. we have security if $m \ll 2^\alpha$.

	Random Ciphertext only attack	KPA	CPA-2
One round of Butterfly	2^n	2^n	4
Benes	2^n	2^n	$\geq 2^{n(1-\varepsilon)} \leq 2^n$
λ rounds of Benes $\lambda \geq 1$	2^n	2^n	$\geq 2^{n(1-\varepsilon)} \leq 2^n$
3 rounds of Feistel	Does not exist *	$2^{n/2}$ **	$2^{n/2}$ **
λ rounds of Feistel $\lambda \geq 6$	Does not exist *	$\geq 2^n \leq 2^{(\lambda-4)n}$ **	$\geq 2^n \leq 2^{(\lambda-4)n}$ **

Fig. 7. Minimum number of computations needed to distinguish the schemes from random functions of $2n$ bits $\rightarrow$ $2n$ bits (or from random permutations for Feistel schemes)

$\geq$: best proved security. $\leq$: best known attack.
* Feistel schemes are permutations, so the ciphertext of m random messages gives m random values. So there are no ciphertext only attacks from random cleartexts.
**cf [15].

B Benes: Example of CPA-1 with $k = 2$ Where $p \simeq \frac{m^2}{4 \cdot 2^{2n}}$

Here we will see a simple ciphertext only attack and simple CPA-1 with $m \simeq 2 \cdot 2^n$ messages, and about $2 \cdot 2^n$ computations. The CPA-1 is not better than the ciphertext only attack that we will see in appendix C, but we will improve this CPA-1 in appendix D with $k = 2$ and $k = 4$. These attacks illustrate the fact that for Benes the security with the number of computations is not larger than the security with the number of messages: these attacks are with $2 \cdot 2^n$ computations. These attacks will also illustrates a difference (by a factor only 2 here) between the expectancy of the number of critical "circles" and the probability that at least such circles exist.

- We choose m such that $\sqrt{m}$ is an integer.
- We choose a set L of $\sqrt{m}$ possible values for the L_i values.
- We choose a set R of $\sqrt{m}$ possible values for the R_i values.
- So our messages are all the $\sqrt{m} \times \sqrt{m} = m$ values $[L_i, R_i]$, where $L_i \in L$ and $R_i \in R$.

(This is a non adaptive chosen plaintext attack, i.e. CPA-1, with m messages). Now we count the number N of (i, j), $i < j$, such that: $S_i = S_j$ and $T_i = T_j$.

First case: Random functions. For random functions, the average value of N is $\frac{m(m-1)}{2 \cdot 2^{2n}}$, since we have $\frac{m(m-1)}{2}$ values (i, j), $i < j$, and when i and j are fixed, we have a probability $\frac{1}{2^{2n}}$ to have $S_i = S_j$ and $T_i = T_j$.

Remark: It can also be shown that the standard deviation from the average value is about $\sqrt{\frac{m(m-1)}{2 \cdot 2^{2n}}}$, i.e. about $\frac{m}{\sqrt{2} \cdot 2^n}$.

Second case: Benes functions. For Benes functions,

$$\begin{cases} S_i = S_j \\ T_i = T_j \end{cases} \Leftrightarrow \begin{cases} f_5(X_i) \oplus f_6(Y_i) = f_5(X_j) \oplus f_6(Y_j) \\ f_7(X_i) \oplus f_8(Y_i) = f_7(X_j) \oplus f_8(Y_j) \end{cases} \quad (1)$$

This can occur either if ($X_i \neq X_j$ or $Y_i \neq Y_j$) and (1) is satisfied with probability $\frac{1}{2^{2n}}$, or if ($X_i = X_j$) and ($Y_i = Y_j$), i.e. if

$$\begin{cases} f_1(L_i) \oplus f_2(R_i) = f_1(L_j) \oplus f_2(R_j) \\ f_3(L_i) \oplus f_4(R_i) = f_3(L_j) \oplus f_4(R_j) \end{cases} \quad (2)$$

since $i < j$, we have $L_i \neq L_j$ or $R_i \neq R_j$, so (2) occurs with probability $\frac{1}{2^{2n}}$. So for Benes functions, the average value of N is about $2 \cdot \frac{m(m-1)}{2 \cdot 2^{2n}}$, instead of about $\frac{m(m-1)}{2 \cdot 2^{2n}}$ for random functions.

Probability to get at least one such value Now let i, j, $i < j$, be two indices such that $X_i = X_j$ and $Y_i = Y_j$.

Case a: $L_i \neq L_j$ **and** $R_i \neq R_j$**.** Then let i' be the index such that: $[L_{i'}, R_{i'}] = [L_i, R_j]$, and let j' be the index such that: $[L_{j'}, R_{j'}] = [L_j, R_i]$. Then $\{i', j'\} \neq \{i, j\}$ (because $L_i \neq L_j$ and $R_i \neq R_j$) and $X_{i'} = X_{j'}$, $Y_{i'} = Y_{j'}$ (this comes immediately from (2)), so we will have: $S_{i'} = S_{j'}$ and $T_{i'} = T_{j'}$. So, if (i, j), $i < j$, are such that $S_i = S_j$ and $T_i = T_j$ and $L_i \neq L_j$ and $R_i \neq R_j$, we will have $S_{i'} = S_{j'}$ and $T_{i'} = T_{j'}$ with probability about $\frac{1}{2}$ if we have a Benes function, and with probability $\frac{1}{2^{2n}}$ if we have a random function.

Case b: $L_i = L_j$ **(we can analyze similarly** $R_i = R_j$**).** Then (2) becomes:

$$\begin{cases} f_2(R_i) = f_2(R_j) \\ f_4(R_i) = f_4(R_j) \end{cases} \quad (3)$$

Now, let i', j' be two indices such that $R_{i'} = R_i$ and $R_{j'} = R_j$. For i', j', we have m possibilities (since for $L_{i'}$ we have $\sqrt{m}$ choices and for $L_{j'}$ we have $\sqrt{m}$ choices).

From (3), we get $X_{i'} = X_{j'}$ and $Y_{i'} = Y_{j'}$, so $S_{i'} = S_{j'}$ and $T_{i'} = T_{j'}$. So if (i, j), $i < j$, is such that $S_i = S_j$ and $T_i = T_j$ and $L_i = L_j$, we will have $S_{i'} = S_{j'}$ and $T_{i'} = T_{j'}$ for all these (i', j') values with probability about $\frac{1}{2}$ if we have a Benes function, and with probability $\frac{1}{2^{2n}}$ for each (i', j') if we have a random function.

Conclusion. While analyzing a Benes function, if we get two indices i, j, $i < j$ such that $X_i = X_j$ and $Y_i = Y_j$, we will easily be able to certify with a very high probability that we have a Benes function by testing the i' and j' inputs/outputs. We can notice that the probability to obtain such indices is $\leq \frac{m(m-1)}{4 \cdot 2^n}$ since each time we get one such index we have in fact immediately 2 or m such indices, and since the average number of such indices is exactly $\frac{m(m-1)}{2 \cdot 2^n}$ for Benes functions.

Remark. For large values of m such that $m(m-1) < 4 \cdot 2^n$, the probability to obtain such indices can be as near as wanted to $\frac{m(m-1)}{4 \cdot 2^n}$, since for large values of m, the number of (i, j), $i < j$, such that $L_i = L_j$ or $R_i = R_j$ becomes negligible compared with the number of (i, j), $i < j$, such that $L_i \neq L_j$ and $R_i \neq R_j$ (i.e. $m\sqrt{m}$ becomes negligible compared with $\frac{m(m-1)}{2} - m\sqrt{m}$).

Conclusion. With $k = 2$, we have obtained here an attack with a probability to distinguish Benes functions from random functions of about $\frac{m(m-1)}{4 \cdot 2^{2n}}$, and an average number of critical values i, j, $i < j$, with $X_i = X_j$ and $Y_i = Y_j$ (i.e. an average number of circles in X, Y of length 2) of about $\frac{m(m-1)}{2 \cdot 2^{2n}}$.

C Benes and λ Rounds of Benes: Example of Ciphertext Only Attack with About $2 \cdot 2^n$ Computations

Let $[L_i, R_i]$ be m random messages. Let N be the number of i, j, $i < j$, such that: $S_i = S_j$ and $T_i = T_j$. With a similar analysis as done in appendix B for the CPA-1, we can easily show that for random functions $N \simeq \frac{m(m-1)}{2 \cdot 2^{2n}}$, and for Benes functions the average value of N is about $N \simeq 2 \cdot \frac{m(m-1)}{2 \cdot 2^{2n}}$ (since $S_i = S_j$ and $T_i = T_j$ can occur if $X_i = X_j$ and $Y_i = Y_j$ with probability $\simeq \frac{1}{2^{2n}}$, or if $X_i \neq X_j$ or $Y_i \neq Y_j$ with probability $\simeq \frac{1}{2^n}$, when i and j are fixed). The probability to distinguish Benes functions from random functions with this ciphertext only attack is about $\frac{m(m-1)}{2 \cdot 2^{2n}}$ (when this value is < 1). (So this ciphertext only attack is slightly better than the CPA-1 of appendix B. We have introduced the CPA-1 because it illustrates what we will do in appendix D).

Remark. More generally, for all integer $\lambda \geq 1$, this ciphertext only attack (i.e. counting the number N of i, j such that $S_i = S_j$ and $T_i = T_j$) distinguishes λ independent rounds of Butterfly from random functions with about m random messages and about 2^n complexity. So, unlike what appears with Feistel schemes (see [15] or appendix A), the number of computations to be done to distinguish λ Butterfly from random functions with our best known attacks do not increase with λ. For some applications Benes schemes may therefore be less useful than Feistel schemes, even if the permutations are not required.

Complexity. The number of computations needed in these attacks (KPA or CPA-1) is about $2 \cdot 2^n$ (with the same memory) since we can store the $[S_i, T_i]$ values and look for collisions.

Remark. It is also possible to need only $\frac{2 \cdot 2^n}{\lambda}$ memory with $\lambda(2 \cdot 2^n)$ computations with the usual time/memory tradeoff algorithm (storing $\frac{2 \cdot 2^n}{\lambda}$ values $[S_i, T_i]$ at each time).

D Benes: Example of CPA-1 with $k = 2$ and $k = 4$ Where $p \simeq \frac{7m^2}{4 \cdot 2^{2n}}$

Here we will see an attack where the probability p to distinguish a random function from a Benes function can be as near as wanted to $\frac{7m(m-1)}{4 \cdot 2^{2n}}$ (for large values m and when $\frac{7m(m-1)}{4 \cdot 2^{2n}}$ is < 1). This shows that the result claimed in [1] (page 318 it is written: $p \leq \frac{m^2}{2^{2n}}$) is not always true (since $\frac{7}{4} > 1$). Moreover this example illustrates with $k = 4$ many things that we consider in this paper for general values of k.

The beginning of the attack is similar with the attack given in appendix B:

- We choose m such that $\sqrt{m}$ is an integer.
- We choose a set L of $\sqrt{m}$ possible values for the L_i values.
- We choose a set R of $\sqrt{m}$ possible values for the R_i values.
- Our messages are all the $\sqrt{m} \times \sqrt{m} = m$ values $[L_i, R_i]$, where $L_i \in L$ and $R_i \in R$.

(This is a CPA-1 with m messages).Now we count the number N of $\{i,j,k,l\}$, i,j,k,l pairwise distinct, such that: $L_i = L_j$, $L_k = L_l$, $R_i = R_k$, $R_j = R_l$, $S_i \oplus S_j \oplus S_k \oplus S_l = 0$, $T_i \oplus T_j \oplus T_k \oplus T_l = 0$ and such that we do not have two indices α and β, $\alpha \neq \beta$, such that $\alpha, \beta \in \{i,j,k,l\}$ and: $S_\alpha = S_\beta$ $T_\alpha = T_\beta$ $(*)$ This extra condition $(*)$ is here to guarantee that this attack of appendix D is really different from the attack of appendix B. At the end we will be able, if we want, to combine the two attacks.

First case: random functions. For random functions, the average value of N is about $\frac{A}{2^{2n}}$, where A is the number of circles in L, R of length 4, i.e. the number of $\{i,j,k,l\}$, i,j,k,l pairwise distinct, such that $L_i = L_j$, $L_k = L_l$, $R_i = R_k$, $R_j = R_l$. We can find the exact value of A: we have $A = \frac{m}{4}(\sqrt{m}-1)^2$ (Proof: For i we have m possibilities.Then for j such that $L_i = L_j$ and $i \neq j$ we have $\sqrt{m}-1$ possibilities. Then for k such that $R_k = R_i$ and $k \neq i$ (i.e. $L_k \neq L_i$) we have $\sqrt{m}-1$ possibilities. Then for l such that $L_l = L_k$ and $R_l = R_j$ we have exactly one possibility when i,j,k are fixed. Like this we have counted all the circles exactly 4 times (we can start the circle with i,j,k or l), so $A = \frac{m}{4}(\sqrt{m}-1)^2$as claimed).

Second case: Benes functions. For Benes functions,

$$\begin{cases} S_i \oplus S_j \oplus S_k \oplus S_l = 0 \\ T_i \oplus T_j \oplus T_k \oplus T_l = 0 \end{cases} \Leftrightarrow$$
$$\begin{cases} f_5(X_i) \oplus f_5(X_j) \oplus f_5(X_k) \oplus f_5(X_l) = f_6(Y_i) \oplus f_6(Y_j) \oplus f_6(Y_k) \oplus f_6(Y_l) \\ f_7(X_i) \oplus f_7(X_j) \oplus f_7(X_k) \oplus f_7(X_l) = f_8(Y_i) \oplus f_8(Y_j) \oplus f_8(Y_k) \oplus f_8(Y_l) \end{cases} \quad (1)$$

This can occur either if the X_i values and the Y_i values can be eliminated two by two (for example if $X_i = X_j$, $X_k = X_l$, $Y_i = Y_k$ and $Y_j = Y_l$), or with probability $\frac{1}{2^{2n}}$ when i,j,k,l are fixed if the X_i values and the Y_i values cannot be eliminated two by two. However we do not want to have two indices α and β, $\alpha \neq \beta$, such that $\alpha, \beta \in \{i,j,k,l\}$ and $X_\alpha = X_\beta$ and $Y_\alpha = Y_\beta$ because this would imply $S_\alpha = S_\beta$ and $T_\alpha = T_\beta$ in contradiction with the condition $(*)$ above. So the X_i values and the Y_i values can be eliminated two by two if and only if we have a circle in X, Y of length 4 (i.e. if we can chose i_1, i_2, i_3, i_4 pairwise distinct in $\{i,j,k,l\}$ with $X_{i_1} = X_{i_2}$, $Y_{i_2} = Y_{i_3}$, $X_{i_3} = X_{i_4}$ and $Y_{i_4} = Y_{i_1}$). We have here 6 possible circles:

1. $X_i = X_j, Y_j = Y_k, X_k = X_l$ and $Y_l = Y_i$
2. $X_i = X_j, Y_j = Y_l, X_l = X_k$ and $Y_k = Y_i$
3. $X_i = X_k, Y_k = Y_j, X_j = X_l$ and $Y_l = Y_i$
4. $X_i = X_k, Y_k = Y_l, X_l = X_j$ and $Y_j = Y_i$
5. $X_i = X_l, Y_l = Y_j, X_j = X_k$ and $Y_k = Y_i$
6. $X_i = X_l, Y_l = Y_k, X_k = X_j$ and $Y_j = Y_i$

Moreover, since we have $L_i = L_j$, $L_k = L_l$, $R_i = R_k$, $R_j = R_l$, we always have $X_i \oplus X_j \oplus X_k \oplus X_l = 0$ and $Y_i \oplus Y_j \oplus Y_k \oplus Y_l = 0$ (because $X_i \oplus X_j \oplus X_k \oplus X_l = f_1(L_i) \oplus f_2(R_i) \oplus f_1(L_j) \oplus f_2(R_j) \oplus f_1(L_k) \oplus f_2(R_k) \oplus f_1(L_l) \oplus f_2(R_l)$ and similarly for Y).

So each of the 6 possible circles have a probability $\frac{1}{2^{2n}}$ to be true when f_1, f_2, f_3, f_4 are randomly chosen (since 2 of the 4 equalities are implied by the 2 other equalities). For example, we have $X_i = X_j$, $Y_j = Y_k$, $X_k = X_l$, $Y_l = Y_i$ if and only if $f_1(L_i) \oplus f_2(R_i) = f_1(L_j) \oplus f_2(R_j)$ and $f_3(L_j) \oplus f_4(R_j) = f_3(L_k) \oplus f_4(R_k)$ i.e. if and only if $f_2(R_i) = f_2(R_j)$ and $f_3(L_i) \oplus f_4(R_j) =$

$f_3(L_k) \oplus f_4(R_i)$. So for Benes functions, the average value of N is about $\frac{7A}{2^{2n}}$, where A is the number of "4-circles in L, R" (7=6+1 since we have 6 possible circles and a probability $\frac{1}{2^{2n}}$ to have (1) if we have no such circles).

Remark: Moreover unlike what appeared with $X_\alpha = X_\beta$ and $Y_\alpha = Y_\beta$ in appendix B, none of these 6 conditions is equivalent to another of these 6 conditions, so the probability that at least one of these conditions is satisfied is really near $\frac{6A}{2^{2n}}$ (for large values of m and when $\frac{6A}{2^{2n}} < 1$).

Conclusion. With $k = 4$, we have obtained here an attack with a probability to distinguish Benes functions from random functions of about $\frac{6m^2}{4 \cdot 2^{2n}}$ and an average number of about also $\frac{6m^2}{4 \cdot 2^{2n}}$ critical values $\{i, j, k, l\}$. This attack can also be combined with the attack with $k = 2$ given in appendix B. Then we obtain an attack with a probability of success of about $\frac{7m^2}{4 \cdot 2^{2n}}$ (with an average number of $\frac{8m^2}{4 \cdot 22^n}$ critical values).

E Why Fixing the Proof of [1] Was Not Easy When $m \gg 2^{2n/3}$

It may seem difficult to use the results of section 5 to get a security in $2^{n(1-\varepsilon)}$ for all $\varepsilon > 0$, since k can be (a priori) as large as m, and then $\frac{k^{2k} m^3}{2^{4n}}$ is not at all negligible when $m \ll 2^{n(1-\varepsilon)}$. Moreover, we can choose m as a square of an integer, and choose all the $[L_i, R_i]$, $1 \leq i \leq m$, such that $L_i \in L$ and $R_i \in R$ where L and R are sets of only $\sqrt{m}$ values. Then let A be the set of all the circles in L, R of length k that we can generate such that two different circles of A have at least one different index. If we consider one such circle C, the probability p that we will get a circle in X, Y between the indices of C can be evaluated as $p \geq$ about $\frac{(k-1)!}{2^{n(k-2)}}$, since we have potentially $(k-1)!$ possible circles in X, Y on the m indices, and since at least two equations (one in X and one in Y) are implied by the other equations in X and Y due to the circle in L, R. So the expectancy for the number of circles in X, Y of A that we will find can be evaluated as $\geq$ about $\frac{|A|(k-1)!}{2^{n(k-2)}}$. Moreover, with our very specific chosen values L_i and R_i, we can show that $|A|$ will be $\simeq \frac{m^{k/2}}{k}$ (for $k = 4$ the exact value is $|A| = \frac{m}{4} \cdot (\sqrt{m} - 1)^2 \simeq \frac{m^2}{4}$). Here, we can have $\frac{m^{k/2}}{k}(k-1)! \gg 2^{n(k-2)}$, with $m \ll 2^n$. For example, with $k = \frac{m}{2}$, it is possible to show that this may indeed happen when $m \gg 2^{2n/3}$. So the expectancy of the number N of circles in X, Y may be large. Nevertheless the probability to obtain at least one such circle will be always negligible when $m \ll 2^n$, as we have seen in section 6. One reason for this is that in a line of equations in X, Y (i.e. $X_{i_1} = X_{i_2}$, $Y_{i_2} = Y_{i_3}$, ..., $Y_{i_{k-1}} = Y_{i_k}$) the value k is not bounded by a fixed integer when $m \ll 2^n$, but the probability to have $k \geq \frac{2^n}{2}$ for example is negligible. We can also notice that all the circles of A are not independent, since we have about $\frac{m^{k/2}}{k}$ circles in A and they are all built from k points chosen in the same set of m points.

The expectancy of N is the sum of the expectancies on all the elements of A (the expectancy of a sum is the sum of the expectancies, even if the variables are not independent). However the probability for N to be $\neq 0$ is not the sum on all the elements of A of the probability to be $\neq 0$ (they are not independent). So we cannot hope to fix the proof of [1] just by computing the expectancy of the number of circles of length k and by summing them. We need to introduce the probability to obtain a line of length k in X, Y. This is what we have done in section 6.

F Improvements of Theorem 5

The value $|\mathcal{C}_k|$ Let $\mathcal{C}_k$ be the set of all possible non equivalent equalities in L and R between the indices $\{i_1, \ldots, i_k\}$. We have $|\mathcal{C}_k| \leq k^{2k}$. Proof: We can find all the equalities in L and R if we know all the $(l(i), r(i))$, $i \in \{i_1, \ldots, i_k\}$, where $l(i)$ is the smallest $j \leq i$ such that $L_j = L_i$ and $r(i)$ is the smallest $j \leq i$ such that $R_j = R_i$. Since we have $\leq k^2$ possibilities for $(l(i), r(i))$ we have immediately $|\mathcal{C}_k| \leq k^{2k}$. However more precise evaluations of $|\mathcal{C}_k|$ are possible. For example we can prove that $|\mathcal{C}_k| \leq \left(\sum_{\lambda=1}^{k/2} \frac{(\lambda+1)^k}{\lambda!}\right)^2$ and we can show that this value is near k^k instead of k^{2k} (the evaluation k^{2k} was enough for our theorems but the coefficient k^{2k} can be improved). Proof: we look first for the possibilities for the equations in L. Let $\mathcal{R}$ be the relation such that for $i, j \in \{i_1, \ldots, i_k\}$, $i\mathcal{R}j \Leftrightarrow L_i = L_j$. This is a relation of equivalence. Let λ be the number of equivalence classes with at least 2 different elements in the classes. We have $\lambda \leq k/2$. Now let B be a set of $\lambda + 1$ boxes, $B = \{B_0, \ldots, B_{\lambda+1}\}$. From an application f of $\{i_1, \ldots, i_k\} \to B$ we can associate equalities in L like this:

- if $f(i) \in B_0$ then there is no $j \neq i$ such that $L_i = L_j$.
- if $f(i) \in B_k$, $k \neq 0$ then the indices j such that $L_j = L_i$ will be exactly all the indices j such that $f(j) \in B_k$. We have $\leq (\lambda + 1)^k$ possibilities for f such that $\forall \alpha$, $1 \leq \alpha \leq \lambda + 1$, $\exists j \in \{i_1, \ldots, i_k\}$ / $f(j) \in B_\alpha$, and each set of possible equations in L can be associate with $\lambda!$ such functions f since we can permute $B_1, \ldots, B_\lambda$ (but not B_0). So the number of non equivalent possibilities for the equations in L is $\leq \sum_{\lambda=1}^{k/2} \frac{(\lambda+1)^k}{\lambda!}$ (the only case with $\lambda = 0$ can be included with $\lambda = 1$), and similarly for the equations in R. So for the equations in L and R we get $|\mathcal{C}_k| \leq \left(\sum_{\lambda=1}^{k/2} \frac{(\lambda+1)^k}{\lambda!}\right)^2$ as claimed.

Conclusion for p_k. Let $A \in \mathcal{C}_k$. Let $\alpha(A)$ be the number of dependent equations of (8) and (9) when the equalities in L and R are those of A. Let $\beta(A)$ be the number of indices of $\{i_1, \ldots, i_k\}$ that we can fix from the other indices of $\{i_1, \ldots, i_k\}$ due to the equalities in L and R of A. Our analysis done in section 5.4 of the equalities of (8) and (9) taken one by one shows that we always have $\alpha(A) \leq \beta(A) + 1$ (the $+1$ comes from the last equation). Moreover, if $\alpha(A) \neq 0$, then when we will consider the first equation of (8) or (9) that gives

a dependency, this dependency comes from one or more than one circle in L, R between the indices (these circles are not necessary disjoints). Now a circle in L, R has always a length ≥ 4. Moreover in a circle in L, R we can fix at least 50% of the indices from the other indices (we just need to know one over two indices and recover the other with the equalities in L and R of the circle). So the first time where we will get a dependency in X or Y we will have at least 2 indices fixed from the others to get this first dependency. So we always have $\alpha(A) \leq \beta(A)$.

We have $p_k \leq \frac{m^k}{k \cdot 2^{nk}} + \sum_{A \in \mathcal{C}_k} \frac{m^{k-\beta(A)}}{2^{n(k-\alpha(A))}}$ since we have $k - \alpha(A)$ independent equations and $\leq m^{k-\beta(A)}$ possibilities for the indices. Since $\alpha(A) \leq \beta(A)$, we get $p_k \leq \frac{m^k}{k \cdot 2^{nk}} + |\mathcal{C}_k| \frac{m^2}{2^{2n}}$. Moreover if $k \geq 6$ it is possible to show that we will have at least 4 independent equations in a circle of length k in X, Y. Then we get : $p_k \leq \frac{m^k}{k \cdot 2^{nk}} + |\mathcal{C}_k| \frac{m^4}{2^{4n}}$, with $|\mathcal{C}_k| \leq (\sum_{\lambda=1}^{k/2} \frac{(\lambda+1)^k}{\lambda!})^2$ for example, when $k \geq 6$.

G An Example for Theorem 5

In the proof of theorem 5, terms in $\mathcal{O}(\frac{m^\alpha}{2^{n\alpha}})$, for some values α, appear for independent equations of (8) and (9) (we then have a term in $\mathcal{O}(\frac{m^k}{2^{nk}})$) or with dependent equations. For dependent equations, all the terms are $\leq \mathcal{O}(\frac{m^2}{2^{2n}})$ when k is fixed, as proved in theorem 5. Is it really possible to have a term in $\mathcal{O}(\frac{m^\alpha}{2^{n\alpha}})$ or is it possible to prove that all the terms are in $\mathcal{O}(\frac{m^{\alpha-1}}{2^{n\alpha}})$ for some values α when we have at least one dependent equation? In fact, as we will see now, it is really possible to have a term in $\mathcal{O}(\frac{m^\alpha}{2^{n\alpha}})$ with some dependent equations. We give such an example in figure 5, with $k = 8$ indices, 4 equations in X and 4 equations in Y. In this example the term is in $\mathcal{O}(\frac{m^4}{2^{4n}})$.

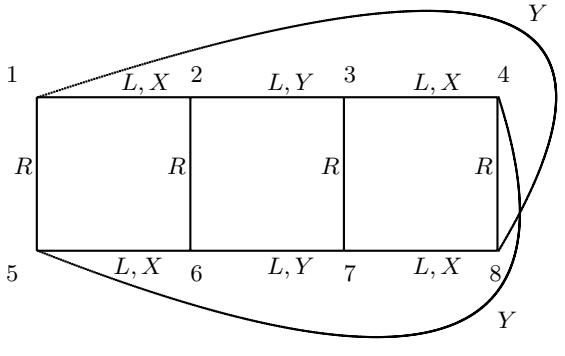


Fig. 8. A line shows an equality between two indices. Here 4 indices can be fixed from the other 4 indices and 4 equations are dependent (2 in X and 2 in Y since the $\oplus$ of all the Y is 0, since the L_i variables are identical two by two and the R_i variables are identical two by two).

H Modified Benes, i.e. Benes with $f_2 = f_3 = Id$

H.1 First Comments on Modified Benes

If we take $f_2 = f_3 = Id$ in the Benes schemes, we obtain a scheme called "Modified Benes" (see [1]). Then we have: $X_i = f_1(L_i) \oplus R_i$, $Y_i = L_i \oplus f_4(R_i)$ and the output $[S_i, T_i]$ is such that: $S_i = f_5(X_i) \oplus f_6(Y_i)$ and $T_i = f_7(X_i) \oplus f_8(Y_i)$. In [1] it is said that the probability p to distinguish this modified Benes from a random function of $2n$ bits $\rightarrow 2n$ bits satisfies $p \leq \frac{m^2}{2^{2n}}$ since we can proceed as for Benes. This evaluation is too optimistic. First, we have at least the same attack with $p \simeq \frac{7m^2}{4 \cdot 2^{2n}}$ as done for Benes in appendix D. Second, modified Benes require a specific analysis since it behaves not exactly as Benes. For example, let us evaluate p_4, the probability to have a circle in X, Y of length 4 for modified Benes. We have: $X_i = X_j$ and $X_k = X_l$ if and only if $f_1(L_i) \oplus R_i = f_1(L_j) \oplus R_j$ and $f_1(L_k) \oplus R_k = f_1(L_l) \oplus R_l$. This can occur for example for $L_i = L_k$, $L_j = L_l$ and $R_i \oplus R_j \oplus R_k \oplus R_l = 0$. Here only one index is fixed (for example l) unlike for Benes where we have seen that at least two indices were fixed for the first dependency. So in p_4 we will get a term in $\frac{m^3}{2^{3n}}$ that did not exist in the original Benes. Similarly, if we consider the probability q_3 to have a line $X_{i_1} = X_{i_2}$, $Y_{i_2} = Y_{i_3}$, $X_{i_3} = X_{i_4}$, we will get $q_3 \leq \frac{m^4}{2^{3n}} + \frac{2m^3}{2^{2n}}$ (unlike $q_3 \leq \frac{m^4}{2^{3n}} + \frac{6m^2}{2^{2n}}$ for the original Benes). So here we have a term in $\frac{m^3}{2^{2n}}$, and therefore we will have to consider longer lines in X, Y to get a security in $m \ll 2^{3n/4}$ for modified Benes compared with the original Benes. As we will see below, it is however possible to prove that for modified Benes, when ε is fixed and $n \rightarrow \infty$, there are no CPA-2 if $m \ll 2^{n(1-\varepsilon)}$. However the evaluation of the security parameter in k that we have obtained is larger for modified Benes compared with the original Benes schemes.

H.2 Ideas of the Proof of Security When $m \ll 2^{n(1-\varepsilon)}$ for Modified Benes

We give here only the main ideas.

Theorem 14. *Let us consider a line of $\lambda + \alpha$ equations in X, Y such that the λ first equations may be dependent or independent, and the other α equations are all dependent from the λ first equations. Then we always have: $\alpha \leq (\lambda + 1)^2$.*

Proof. We have: $X_{i_1} = X_{i_2}$, $Y_{i_2} = Y_{i_3}$, $X_{i_3} = X_{i_4}$, ..., $Y_{i_\lambda} = Y_{i_{\lambda+1}}$ (and these λ equations (A) are dependent or independent), and we have: $X_{i_{\lambda+1}} = X_{i_{\lambda+2}}$, $Y_{i_{\lambda+2}} = Y_{i_{\lambda+3}}$, ..., $Y_{i_{\lambda+\alpha}} = Y_{i_{\lambda+\alpha+1}}$ (and these α equations (B) are dependent from the λ equations of (A)). If an equation $X_i = X_j$ is dependent from previous equations, then L_i and L_j are values that have appeared before, since $X_i = X_j \Leftrightarrow f_1(L_i) \oplus R_i = f_1(L_j) \oplus R_j$, and here we see that $i \neq j$ implies $L_i \neq L_j$. Similarly, if an equation $Y_k = Y_l$ is dependent from previous equations, then R_k and R_l are values that have appeared before, since $Y_k = Y_l \Leftrightarrow L_k \oplus f_4(R_k) = L_l \oplus f_4(R_l)$, and here we see that $k \neq l$ implies $R_k \neq R_l$. Now from the λ equations of (A) we have at most $\lambda + 1$ values L_i and $\lambda + 1$ values R_j, so at most $(\lambda + 1)^2$ values (L_i, R_j) are possible in (B).

Circles in X, Y In a circle (C) in X, Y, we can analyze the equations in X and in Y as we did with theorem 14 in a line, if we can put at the end an equation in X or Y which is independent from the others. If not, then this means that all the equations in X and Y are dependent from the other equations in X or Y. However in this case all index i of the circle (C) is such that there is an index j in (C), $j \neq i$ such that $L_i = L_j$ and similarly there is an index k in (C) such that $k \neq i$ and $R_i = R_k$. In this case we will have at least one circle in L, R (with indices from the circle (C) in X, Y) so at least 2 indices can be fixed from the other indices (as we have seen for the original Benes). So for the modified Benes, we will get a security in $m \ll 2^{n(1-\varepsilon)}$ as for the original Benes (but with slightly different security coefficients).

Relative Doubling Attack Against Montgomery Ladder*

Sung-Ming Yen[1], Lee-Chun Ko[1], SangJae Moon[2], and JaeCheol Ha[3]

[1] Laboratory of Cryptography and Information Security (LCIS),
Dept of Computer Science and Information Engineering,
National Central University, Chung-Li, Taiwan 320, R.O.C
{yensm, cs222085}@csie.ncu.edu.tw
http://www.csie.ncu.edu.tw/~yensm/
[2] School of Electronic and Electrical Engineering,
Kyungpook National University, Taegu 702-701, Korea
sjmoon@ee.knu.ac.kr
[3] Dept of Computer and Information,
Korea Nazarene University, Choong Nam 330-718, Korea
jcha@kornu.ac.kr

Abstract. Highly regular execution and the cleverly included redundant computation make the square-multiply-always exponentiation algorithm well known as a good countermeasure against the conventional simple power analysis (SPA). However, the doubling attack threatens the square-multiply-always exponentiation by fully exploiting the existence of such redundant computation. The Montgomery ladder is also recognized as a good countermeasure against the conventional SPA due to its highly regular execution. Most importantly, no redundant computation is introduced into the Montgomery ladder. In this paper, immunity of the Montgomery ladder against the doubling attack is investigated. One straightforward result is that the Montgomery ladder can be free from the original doubling attack. However, a non-trivial result obtained in this research is that a relative doubling attack proposed in this paper threatens the Montgomery ladder. The proposed relative doubling attack uses a totally different approach to derive the private key in which the relationship between two adjacent private key bits can be obtained as either $d_i = d_{i-1}$ or $d_i \neq d_{i-1}$. Finally, a remark is given to the problem of whether the upward (right-to-left) regular exponentiation algorithm is necessary against the original doubling attack and the proposed relative doubling attack.

Keywords: Chosen-message attack, Cryptography, Doubling attack, Exponentiation, Scalar multiplication, Side-channel attack, Simple power analysis (SPA), Smart card.

1 Introduction

Cryptographic hardware devices like smart cards are widely used nowadays. During the past few years many research results have been published on considering

* This work was supported by University IT Research Center Project.

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 117–128, 2006.
© Springer-Verlag Berlin Heidelberg 2006

smart card side-channel attacks because of the popular usage of smart cards on implementing cryptosystems. This new branch of cryptanalysis is usually called the side-channel attack. The power analysis attack is an important category of side-channel attack originally pointed out by Kocher [1] in which both simple power analysis (SPA) and differential power analysis (DPA) were considered.

In a SPA, the attacker observes on one or a few collected power traces of the smart card executing an algorithm and tries to identify the occurrence of an instruction execution or a specific operand/data access which are driven by a part of the private key. Through the above observation, if precise enough, the private key can be derived. In a DPA, the attacker tries to verify his guess on a part of the private key by analyzing on only some specific bits of the result of a specific intermediate step of an algorithm which is a function of the private key. In order to largely enhance the signal to noise ratio to mount a successful DPA, it usually collects much more[1] power traces than in a SPA and partitions the power traces into some groups according to the guessed key bits and a underlying attack design. Difference of the above power traces of different groups is therefore used to verify the guess on key bits. Usually, a DPA is mounted by analyzing on many executions of the same algorithm with different random inputs, and theoretically those inputs will be better if statistically unrelated.

Exponentiation and its analogy, point scalar multiplication on elliptic curve, are of central importance in modern cryptosystems implementation as they are of the basic operation of almost all modern public-key cryptosystems, e.g., the RSA system [2], the ElGamal system [3], and the elliptic curve cryptography [4, 5]. Therefore, many side-channel attacks and also the related countermeasures on implementing exponentiation and point scalar multiplication have been reported in the literature.

The square-multiply-always exponentiation (or point scalar multiplication) algorithm [6] is a well-known SPA countermeasure which exploits a simple and useful trick to design a regularly executing algorithm by introducing redundant computation into each loop iteration when necessary. Unfortunately, Fouque and Valette proposed the doubling attack [7] to threaten the square-multiply-always algorithm (more precisely, the left-to-right version of the algorithm) by exploiting the existence of redundant computation in a novel approach.

Joye and Yen proposed an enhanced SPA countermeasure based on the Montgomery ladder [8] which was demonstrated to be also regularly executed but based on a totally different idea from the original square-multiply-always exponentiation. The most special thing about the Montgomery ladder is that no redundant computation exists in the algorithm which is also helpful to be immune from some hardware fault attacks [9, 10].

However, no research has been reported on whether the Montgomery ladder can be immune from the doubling attack or any doubling-like attack in the light of the fact of no redundant computation within the algorithm. The main contribution of this paper is that a totally different approach of doubling attack

[1] It usually collects a few thousands or more power traces in order to obtain a meaningful average power trace.

(called the relative doubling attack) is proposed which can successfully threaten the Montgomery ladder with the same attack assumption as the original doubling attack. The lesson learned is that redundant computation removing in a regular exponentiation algorithm may not be sufficient against doubling-like attack.

2 Exponentiation Algorithm and Simple Power Analysis

2.1 Exponentiation Algorithm

In this paper, we consider the problem of computing modular exponentiation. In the context of RSA private computation (e.g., decryption), we consider the computation of $S = M^d \bmod n$ where M, d, and n are the input datum, the private key, and the modulus integer, respectively.

Let $\sum_{i=0}^{m-1} d_i 2^i$ be the binary expansion of exponent d. The computation $S = M^d \bmod n$ needs efficient exponentiation algorithms to speedup its implementation. Although numerous exponentiation algorithms have been developed for computing $M^d \bmod n$ (see [11] for a survey), practical solutions for devices with constrained computation and storage capabilities (e.g., smart cards) are usually restricted to the basic square-multiply algorithms in Fig. 1 and some slightly modified ones. The left-to-right (MSB-to-LSB) version in Fig. 1 (a) is especially preferable to implementations in smart cards because this algorithm needs only one temporary memory R_0 if the input data M is also stored inside the smart cards.

	Input: $M, d = (d_{m-1} \cdots d_0)_2, n$ Output: $M^d \bmod n$
01	$R_0 \leftarrow 1$
02	**for** $i = m-1$ **downto** 0 **do**
03	$R_0 \leftarrow (R_0)^2 \bmod n$
04	**if** $(d_i = 1)$ **then**
	$R_0 \leftarrow R_0 \times M \bmod n$
05	**return** R_0

(a) Left-to-right binary algorithm.

	Input: $M, d = (d_{m-1} \cdots d_0)_2, n$ Output: $M^d \bmod n$
01	$R_0 \leftarrow 1$; $R_1 \leftarrow M$
02	**for** $i = 0$ **to** $m-1$ **do**
03	**if** $(d_i = 1)$ **then**
	$R_0 \leftarrow R_0 \times R_1 \bmod n$
04	$R_1 \leftarrow (R_1)^2 \bmod n$
05	**return** R_0

(b) Right-to-left binary algorithm.

Fig. 1. Classical binary exponentiation algorithms

2.2 Simple Power Analysis and Countermeasures

Side-channel attacks are developed based on the fact that in most real implementations some side-channel information (e.g., timing or power consumption) will depend on the private key related instructions being executed and/or the data being manipulated. Therefore, the side-channel information may be exploited to mount a successful attack to retrieve the embedded private key, e.g., the private exponent d in $M^d \bmod n$.

The classical binary exponentiation algorithm in Fig. 1 (a) includes a conditional branch (i.e., the Step (04)) that is driven by the secret data d_i. If the

two possible branches behave differently (or the branch decision operation itself behaves distinguishably), then some side-channel analysis (e.g., the simple power analysis–SPA) may be employed to retrieve the secret data d_i. So, further enhancement on the algorithms is necessary.

A novel idea of introducing redundant operations and eliminating secret data dependent statements was proposed previously to enhance the basic algorithms such that the improved versions behave more regularly. Some square-multiply-always (or its counterpart called the double-add-always for point scalar multiplication) based algorithms were already developed [6] by employing this observation. Two of these square-multiply-always algorithms are shown in Fig. 2.

```
   Input: M, d = (d_{m-1} ··· d_0)_2, n
   Output: M^d mod n
01   R_0 ← 1
02   for i = m − 1 downto 0 do
03     b ← ¬d_i
04     R_0 ← (R_0)^2 mod n
05     R_b ← R_0 × M mod n
06   return R_0
```

(a) SPA-protected left-to-right algorithm.

```
   Input: M, d = (d_{m-1} ··· d_0)_2, n
   Output: M^d mod n
01   R_0 ← 1; R_2 ← M
02   for i = 0 to m − 1 do
03     b ← ¬d_i
04     R_b ← R_0 × R_2 mod n
05     R_2 ← (R_2)^2 mod n
06   return R_0
```

(b) SPA-protected right-to-left algorithm.

Fig. 2. SPA-protected square-multiply-always countermeasures

2.3 Doubling Attack

The doubling attack[2] [7] is a special category of SPA with chosen message assumption and it has been shown to be useful to thwart the well-known SPA-protected left-to-right (downward) square-multiply-always countermeasure (see Fig. 2 (a)). The main idea is to choose two strongly related inputs M and $M^2 \bmod n$ (so being a chosen-message attack) and to observe the collision of two computations for $M^{2(2x+d_i)} \bmod n$ and $M^{4x} \bmod n$ if $d_i = 0$. In the doubling attack, even if the attacker cannot decide whether a computation being performed is squaring or multiplication, the attacker can still detect collision of two operations (basically the squaring operation) within two related computations. More precisely, for two computations $A^2 \bmod n$ and $B^2 \bmod n$, even if the attacker cannot tell the values of A and/or B, however the attacker can detect the collision if $A = B$.

The following example given in Table 1 provides the details of the doubling attack. Let the private exponent d be $75 = (1, 0, 0, 1, 0, 1, 1)_2$ and the two related input data be M and M^2, respectively. The computational process of raising M^d and $(M^2)^d$ using the left-to-right square-multiply-always algorithm reveals the fact that if $d_i = 0$, then both the first computations (both are squarings) of

[2] It can also be called the squaring attack for the scenario of exponentiation.

iteration[3] $i-1$ for M^d and iteration i for $(M^2)^d$ will be exactly the same. So, observing collisions (observation on the existence of same instruction with same operand) within computations of two collected power consumption traces enables the attacker to identify all private key bits of zero value except the LSB of d. In the scenario of RSA private computation, it is assumed that $d_0 = 1$.

The assumption made (was claimed in [7] to be correct by experiment) is very reasonable since the target computations usually take many machine clock cycles (thus more easy to measure and to observe) and depend greatly on the operands, so the collision is more easy to detect.

Table 1. Computations of M^d and $(M^2)^d$ in the square-multiply-always algorithm

i	d_i	Process of M^d	Process of $(M^2)^d$
6	1	1^2	1^2
		$1 \times M$	$1 \times M^2$
5	0	M^2	$(M^2)^2$
		$M^2 \times M$	$M^4 \times M^2$
4	0	$(M^2)^2$	$(M^4)^2$
		$M^4 \times M$	$M^8 \times M^2$
3	1	$(M^4)^2$	$(M^8)^2$
		$M^8 \times M$	$M^{16} \times M^2$
2	0	$(M^9)^2$	$(M^{18})^2$
		$M^{18} \times M$	$M^{36} \times M^2$
1	1	$(M^{18})^2$	$(M^{36})^2$
		$M^{36} \times M$	$M^{72} \times M^2$
0	1	$(M^{37})^2$	$(M^{74})^2$
		$M^{74} \times M$	$M^{148} \times M^2$
Return		M^{75}	M^{150}

2.4 Montgomery Ladder as Enhanced Countermeasure Against SPA

Montgomery ladder is originally due to Peter Montgomery [12] as a means to speed up scalar multiplication in the context of elliptic curves. It has been rediscovered in [13] in another context and applied to Lucas sequences.

In [8], an exponentiation algorithm based on Montgomery ladder was considered such that it can resist some side-channel attacks, e.g., SPA and timing attack, and also the safe-error attacks [9, 10] (a category of hardware fault attack). The algorithm is given in Fig. 3. This algorithm is only SPA resistant and is used to simplify the description of the proposed attack. However, an enhanced version in [8] meant to be immune from the safe-error attacks with Step `04` replaced by $(R_b \leftarrow R_b \times R_{d_i} \bmod n)$ is still vulnerable to the relative doubling attack proposed in this paper.

It is evident that the Montgomery ladder (and its enhanced version) behave regularly and most specially that there is no redundant computation within the algorithm.

[3] Here, the iteration number is denoted decreasingly from $m-1$ downward towards zero.

```
   Input: M, d = (d_{m-1} ··· d_0)_2, n
   Output: M^d mod n
01   R_0 ← 1; R_1 ← M
02   for i = m − 1 downto 0 do
03     b ← ¬d_i
04     R_b ← R_0 × R_1 mod n
05     R_{d_i} ← (R_{d_i})^2 mod n
06   return R_0
```

Fig. 3. SPA-protected Montgomery ladder

3 The Proposed Attack

3.1 Attack Assumption

The assumption made in this paper is basically the same as what considered in the doubling attack [7] and that in an attack reported in [14]. The assumption is that an adversary can distinguish collision of power trace segments (within a single or more power traces) when the smart card performs twice the same computation even if the adversary is not able to tell which exact computation is done. The collision instance to be distinguished in [7] and in our proposed attack is the modular squaring computation. An adversary is assumed to be able to detect the collision of $A^2 \bmod n$ and $B^2 \bmod n$ if $A = B$ even though A and B are unknown.

3.2 Relative Doubling Attack on Montgomery Ladder

Let $\sum_{j=0}^{m-1} d_j\, 2^j$ be the binary expansion of the exponent d. The Montgomery ladder (see Fig. 3) was designed based on the following observation [8]. Let $L_i = \sum_{j=i}^{m-1} d_j\, 2^{j-i}$ and $H_i = L_i + 1$, then we have

$$L_i = 2L_{i+1} + d_i = L_{i+1} + H_{i+1} - 1 + d_i,$$

$$H_i = L_{i+1} + H_{i+1} + d_i = 2H_{i+1} - 1 + d_i.$$

Based on the above observation, we obtain

$$(L_i, H_i) = \begin{cases} (2L_{i+1},\, L_{i+1} + H_{i+1}) & \text{if } d_i = 0\,, \\ (L_{i+1} + H_{i+1},\, 2H_{i+1}) & \text{if } d_i = 1\,. \end{cases} \tag{1}$$

In the algorithm (Fig. 3), the register R_0 is used to store the value of M^{L_i} and the register R_1 is used to store M^{H_i}. In order to develop an execution regular and SPA immune algorithm, the operations of Step 04 and Step 05 are designed to be as follows

$$(R_1 = M^{H_i}, R_0 = M^{L_i}) = \left(M^{L_{i+1}} \times M^{H_{i+1}}, (M^{L_{i+1}})^2\right) \quad \text{if } d_i = 0, \tag{2}$$

and

$$(R_0 = M^{L_i}, R_1 = M^{H_i}) = \left(M^{L_{i+1}} \times M^{H_{i+1}}, (M^{H_{i+1}})^2\right) \quad \text{if } d_i = 1. \tag{3}$$

The above statements clearly demonstrate that the Montgomery ladder executes highly regular and there is no redundant computation within the algorithm. Whatever the processed bit d_i, there is always a multiplication followed by a squaring. On the contrary, we want to emphasize that in the left-to-right square-multiply-always algorithm (see Fig. 2 (a)), redundant computation (i.e., Step 05: $R_b \leftarrow R_0 \times M \bmod n$) does exist when $d_i = 0$. The original doubling attack on the algorithm in Fig. 2 (a) exploits the existence of this redundant computation.

However, no research has been reported on whether the Montgomery ladder can be immune from the doubling attack or any doubling-like attack in the light of the fact of no redundant computation within the algorithm. A straightforward result can be obtained easily is that the original doubling attack does not apply to the Montgomery ladder. However, the following result will show that another doubling-like attack can still be applicable to the Montgomery ladder.

Fact 1. Given $d_i = 0$, then we have $L_i = 2L_{i+1}$.

Proof. This can be obtained directly from the definition of $L_i = \sum_{j=i}^{m-1} d_j \, 2^{j-i}$ since $d_i = 0$. □

Fact 2. Given $d_i = 1$, then we have $H_i = 2H_{i+1}$.

Proof. From the definitions of $L_i = \sum_{j=i}^{m-1} d_j \, 2^{j-i}$, $H_i = L_i + 1$, and also $d_i = 1$, we have $H_i = L_i + 1 = (2L_{i+1} + 1) + 1 = 2(L_{i+1} + 1) = 2H_{i+1}$. □

From Eq.(2), we understand that if $d_i = d_{i-1} = 0$ then both

$$\begin{cases} R_0 \leftarrow (M^{L_i})^2 : \text{ Step 05 of iteration } i-1 \text{ when evaluating } M^d \\ R_0 \leftarrow ((M^2)^{L_{i+1}})^2 : \text{ Step 05 of iteration } i \text{ when evaluating } (M^2)^d, \end{cases} \tag{4}$$

will perform the same computation because of $L_i = 2L_{i+1}$ (see Fact 1). Due to this observation of collision on computation, a new doubling-like attack can be mounted to derive the knowledge of $d_i = d_{i-1} = 0$.

On the other hand, from Eq.(3), we also observe that if $d_i = d_{i-1} = 1$ then both

$$\begin{cases} R_1 \leftarrow (M^{H_i})^2 : \text{ Step 05 of iteration } i-1 \text{ when evaluating } M^d \\ R_1 \leftarrow ((M^2)^{H_{i+1}})^2 : \text{ Step 05 of iteration } i \text{ when evaluating } (M^2)^d, \end{cases} \tag{5}$$

will perform the same computation because of $H_i = 2H_{i+1}$ (see Fact 2). This observation of collision on computation leads to the knowledge of $d_i = d_{i-1} = 1$.

All other cases, say $d_i \neq d_{i-1}$, will lead to either one of the following results

case (1): $d_i = 0$ **and** $d_{i-1} = 1$

$$\begin{cases} R_1 \leftarrow (M^{H_i})^2 : \text{ Step 05 of iteration } i-1 \text{ when evaluating } M^d \\ R_0 \leftarrow ((M^2)^{L_{i+1}})^2 : \text{ Step 05 of iteration } i \text{ when evaluating } (M^2)^d, \end{cases} \tag{6}$$

case (2): $d_i = 1$ **and** $d_{i-1} = 0$

$$\begin{cases} R_0 \leftarrow (M^{L_i})^2 : \text{ Step 05 of iteration } i-1 \text{ when evaluating } M^d \\ R_1 \leftarrow ((M^2)^{H_{i+1}})^2 : \text{ Step 05 of iteration } i \text{ when evaluating } (M^2)^d. \end{cases} \quad (7)$$

Based on the definition of Montgomery ladder, it is evident that in the case (1) we have $H_i \neq 2L_{i+1}$ and no collision of computation can be detected. Similarly, in the case (2) no collision of computation can be detected since $L_i \neq 2H_{i+1}$.

The Relative Doubling Attack. Recall that collision of two computations will not reveal the value of the operand. So, in the proposed attack, a collision of computations detected by the property of Eq.(4) and another collision of computations detected by the property of Eq.(5) cannot be distinguished. The only knowledge obtained is that $d_i = d_{i-1}$ if a collision is detected. On the other hand, the properties in Eq.(6) and Eq.(7) tell us that $d_i \neq d_{i-1}$ if no collision is detected. Due to its special property of the derived knowledge, the proposed attack is called the relative doubling attack to manifest the difference to the original doubling attack [7].

Based on the derived relationship between every two adjacent private key bits (either $d_i = d_{i-1}$ or $d_i \neq d_{i-1}$) and a given bit (e.g., d_0 or d_{m-1}), all other private key bits can be derived uniquely. For example, in RSA private computation it is assumed that $d_0 = 1$ under the same assumption made in the original doubling attack mentioned previously. Most importantly, we observed that it is sufficient to derive all the private key bits when given any d_i $(0 \leq i \leq m-1)$.

An Example of Attack. Following the same example of assuming the private exponent d to be $75 = (1,0,0,1,0,1,1)_2$ and the two related input data to be M and M^2 respectively, Table 2 provides the details of the proposed relative doubling attack on Montgomery ladder. The computational process of raising M^d and $(M^2)^d$ reveals the fact that given[4] $d_0 = 1$ and the observation of collision on Step 05 of the iteration 1 of $(M^2)^d$ and Step 05 of the iteration 0 of M^d will lead to the result of $d_1 = d_0 = 1$. Given $d_0 = 1$, if no collision is detected, then $d_1 = 0$ since in this case d_1 should be different from d_0.

3.3 Comparison of Doubling Attack and Relative Doubling Attack

The original doubling attack (against square-multiply-always algorithm) focuses on deriving the private key bit d_i by checking whether $d_i = 0$. So, the original doubling attack tries to obtain the knowledge of *absolute* value of each d_i. On the contrary, the proposed relative doubling attack (against Montgomery ladder) focuses on deriving the knowledge of whether $d_i = d_{i-1}$ (relationship between every two adjacent key bits), but not the knowledge of either d_i or d_{i-1} directly. Nonetheless, given the value of either d_i or d_{i-1} will provide the exact value of the other one indirectly.

[4] The RSA private exponent d is an odd integer. The original doubling attack also exploits this knowledge in order to obtain d_0.

Table 2. Computations of M^d and $(M^2)^d$ in the Montgomery ladder

i	d_i	Process of M^d	Process of $(M^2)^d$
6	1	$R_0 = 1 \times M$ $R_1 = M^2$	$R_0 = 1 \times M^2$ $R_1 = (M^2)^2$
5	0	$R_1 = M^2 \times M$ $R_0 = M^2$	$R_1 = M^4 \times M^2$ $R_0 = (M^2)^2$
4	0	$R_1 = M^3 \times M^2$ $R_0 = (M^2)^2$	$R_1 = M^6 \times M^4$ $R_0 = (M^4)^2$
3	1	$R_0 = M^4 \times M^5$ $R_1 = (M^5)^2$	$R_0 = M^8 \times M^{10}$ $R_1 = (M^{10})^2$
2	0	$R_1 = M^{10} \times M^9$ $R_0 = (M^9)^2$	$R_1 = M^{20} \times M^{18}$ $R_0 = (M^{18})^2$
1	1	$R_0 = M^{18} \times M^{19}$ $R_1 = (M^{19})^2$	$R_0 = M^{36} \times M^{38}$ $R_1 = (M^{38})^2$
0	1	$R_0 = M^{37} \times M^{38}$ $R_1 = (M^{38})^2$	$R_0 = M^{74} \times M^{76}$ $R_1 = (M^{76})^2$
Return		$R_0 = M^{75}$	$R_0 = M^{150}$

Furthermore, the original doubling attack fully exploits the existence of redundant computation. But the proposed relative doubling attack on the Montgomery ladder does not exploit the existence of any redundant computation. Evidently, in this paper, we showed a totally different approach of deriving the private key. The primary similarity of these two attacks is that both of them use (M, M^2) as the chosen input data.

3.4 Applicability of the Proposed Attack

Most published research results on side-channel attack considered the potential vulnerability on the computational algorithm (e.g., modular exponentiation) but not on a real cryptosystem and under a specific cryptographic standard (e.g., some padding or message format). This is basically reasonable since the computational algorithm itself is generic and can be employed as implementation to many different cryptosystems (or some cryptosystems to be designed in the future) each may have different padding or message format. So, to point out potential attacks to the computational algorithm is still important.

Nonetheless, we still wish to point out clearly that the proposed relative doubling attack is applicable at least to the following systems if they are implemented based on the Montgomery ladder or its enhanced version in [8].

(1) Traditional textbook RSA decryption and signature.
(2) The RSA-OAEP decryption [15, 16]. It should be noted that the proposed attack does work on RSA-OAEP decryption since the ciphertext validity checking is performed after the RSA private exponentiation computation. So, the attacker still can collect the necessary power traces.
(3) ElGamal decryption [3].

4 Possible Enhancement Against Relative Doubling Attack

4.1 Remarks on Random Blinding Technique

One may argue that the standard blinding technique can easily prevent the proposed relative doubling as well as the original doubling attacks. However, we have some remarks on this claim.

The first disagreement is that the standard blinding technique is well known as a countermeasure against DPA. The second disagreement is that in a standard blinding technique the input data should be protected by a random mask which will then be removed from the result. However, it has been pointed out clearly in [7] that a regular mask updating (meant to be efficient), e.g., the one mentioned in [6], will be vulnerable to the doubling attack. It can be verified easily that the regular mask updating in [6] is also vulnerable to the proposed relative doubling attack. It was suggested eventually that it had better use a real random mask to avoid the attack. Unfortunately, the computational overhead of employing a real random mask is usually very high.

4.2 Is Upward Exponentiation Necessary Against Doubling Attack

The work and especially the title of [7] imply that upward (right-to-left) exponentiation could be better than downward (left-to-right) exponentiation when considering vulnerability from the doubling attack. This is also the case for the proposed relative doubling attack. However, the above mentioned superiority of the upward exponentiation is not obtained without any additional cost. It is evident that the upward square-multiply-always exponentiation in Fig. 2 (b) needs one more temporary memory than the downward exponentiation does.

Purpose of the following discussion is to clarify that upward exponentiation is not a necessary requirement meant to be immune from the doubling attack and the proposed relative doubling attack. The following SPA-protected and safe-error-protected exponentiation algorithm [17] in Fig. 4 is a downward exponentiation algorithm. A limitation of this algorithm is that $d_{m-1} = 1$ is assumed. It can be verified easily that this algorithm is secure against the doubling and the relative doubling attacks.

Notice that the algorithm (Fig. 4) needs only two temporary memory (same as that in Fig. 2 (a)) and this leads to *one* less temporary memory requirement than the doubling attack immune upward algorithm in Fig. 2 (b). Recall that if we take into account the fact that the input datum M is also stored inside the smart card (as already described previously), then the algorithm in Fig. 4 needs only one temporary memory which leads to *two* less temporary memory requirement than the doubling attack immune upward algorithm in Fig. 2 (b). However, it is worth noting that protection against relative doubling attack does not necessarily ward off other potential attacks.

```
Input: M, d = (d_{m-1}, d_{m-2} ··· d_0)_2, d_{m-1}=1, n
Output: M^d mod n
01  R_0 ← 1; R_1 ← M; d_{-1} ← 1
02  for i = m − 1 downto 0 do
03    b ← ¬d_i; c ← d_{i-1}
04    R_0 ← R_0 × R_b mod n
05    R_0 ← R_0 × R_c mod n
06  return R_0
```

Fig. 4. SPA-protected and safe-error-protected downward exponentiation

5 Conclusions

The Montgomery ladder can be secure against both the ordinary SPA and the ordinary doubling attack. But, in this paper we showed that the Montgomery ladder is vulnerable to the proposed relative doubling attack. Both the ordinary doubling attack and the proposed relative doubling attack share the same reasonable attack assumption of observing collision on computations. One difference is that the original doubling attack (against square-multiply-always algorithm) fully exploits the existence of redundant computation, while the proposed relative doubling attack (against Montgomery ladder) does not exploit any redundant computation. Our relative doubling attack uses a different approach to derive the private key.

Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful suggestions and comments on both technical and editing issues. These suggestions improve extensively to the final version of this paper.

References

1. P. Kocher, J. Jaffe and B. Jun, "Differential power analysis," *Advances in Cryptology – CRYPTO '99*, LNCS 1666, pp. 388–397, Springer-Verlag, 1999.
2. R.L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystem," *Commun. of ACM*, vol. 21, no. 2, pp. 120–126, 1978.
3. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, 1985.
4. V. Miller, "Uses of elliptic curve in cryptography," *Advances in Cryptology – CRYPTO '85*, LNCS 218, pp. 417–426, Springer-Verlag, 1985.
5. N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, Jan. 1987.
6. J.-S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," *Proc. of Cryptographic Hardware and Embedded Systems – CHES '99*, LNCS 1717, pp. 292–302, Springer-Verlag, 1999.

7. P.-A. Fouque and F. Valette, "The doubling attack – why upwards is better than downwards," *Proc. of Cryptographic Hardware and Embedded Systems – CHES '03*, LNCS 2779, pp. 269–280, Springer-Verlag, 2003.
8. M. Joye and S.M. Yen., "The Montgomery powering ladder," *Proc. of Cryptographic Hardware and Embedded Systems – CHES '02*, LNCS 2523, pp. 291–302, Springer-Verlag, 2003.
9. S. M. Yen and M. Joye, "Checking Before Output May Not be Enough against Fault-Based Cryptanalysis," *IEEE Trans. on Computers*, 49(9):967-970, September 2000.
10. S.M. Yen, S.J. Kim, S.G. Lim and S.J. Moon, "A countermeasure against one physical cryptanalysis may benefit another attack," *Proc. of Information Security and Cryptology – ICISC '01*, LNCS 2288, pp. 414–427, Springer-Verlag, 2002.
11. D.M. Gordon, "A survey of fast exponentiation methods," *Journal of Algorithms*, vol. 27, pp. 129–146, 1998.
12. P.L. Montgomery, "Speeding the Pollard and elliptic curve methods of factorization," *Mathematics of Computation*, vol. 48, no. 177, pp. 243–264, Jan. 1987.
13. S.M. Yen and C.S. Laih, "Fast algorithms for LUC digital signature computation," *IEE Proc. Computers and Digital Techniques*, vol. 142, no. 2, pp. 165–169, March 1995.
14. K. Schramm, T. Wollinger, and C. Paar, "A new class of collision attacks and its application to DES," *Proc. of Fast Software Encryption – FSE '03*, LNCS 2887, pp. 206–222, Springer-Verlag, 2003.
15. PKCS #1 v2.1, "RSA Cryptography Standard", 5 January 2001. `http://www.` rsasecurity.com/rsalabs/pkcs/
16. M. Bellare and P. Rogaway, "Optimal asymmetric encryption padding – How to encrypt with RSA," *Advances in Cryptology – EUROCRYPT '94*, LNCS 950, pp. 92–111, Springer-Verlag, 1995.
17. S.M. Yen, C.C. Lu, and S.Y. Tseng, "Method for protecting public key schemes from timing, power and fault attacks," U.S. Patent Number US2004/0125950 A1, July 2004.

Improved Collision Attack on MD4 with Probability Almost 1

Yusuke Naito, Yu Sasaki, Noboru Kunihiro, and Kazuo Ohta

The University of Electro-Communications,
Chofugaoka 1-5-1, Chofo-shi, Tokyo 182-8585, Japan
{tolucky, yu339, kunihiro, ota}@ice.uec.ac.jp

Abstract. In EUROCRYPT2005, a collision attack on MD4 was proposed by Wang, Lai, Chen, and Yu. They claimed that collision messages were found with probability 2^{-6} to 2^{-2}, and the complexity was less than 2^8 MD4 hash operations. However, there were some tyops and oversights in their paper. In this paper, first, we reevaluate the exact success probability. Second, we point out the typos and oversights in the paper of Wang et al, and we show how to improve them. Third, we propose a new message modification method for the third round of MD4. From the first result, we reevaluate that the method of Wang et al. can find collision messages with success probability $2^{-5.61}$. From the second result, we can find collision messages with success probability 2^{-2}. Also by combining the second result and the third result, our improved method is able to find collision messages with probability almost 1. This complexity is less than 3 repetitions of MD4 hash operations. Our improved method is about 85 times as fast as the method of Wang et al.

1 Introduction

MD4 is a hash function which was proposed by Rivest in CRYPTO'90 [7]. After MD4 was proposed, various hash functions based on MD4 were proposed such as MD5, RIPEMD, SHA-family, and so on. MD4 consists of addition, XOR and bit-rotation, and thus can be calculated fast.

In EUROCRYPT 2005, Wang et al. proposed an efficient attack for MD4 [9]. This attack is a differential attack. While differential attacks before [9], such as [2] or [3] used XOR to calculate differential, the method of Wang et al. uses modular subtraction.

In this method, input differential is decided in advance. Then, the attack will succeed if output differential is 0. The method in [9] makes conditions on chaining variables in order to cancel the input differential. In this paper, we call these conditions on chaining variables introduced to cancel the input differential as "sufficient condition". If an inputted message into MD4 satisfies all the sufficient condition, the output differential becomes 0, and we can generate a collision message. However, the probability that a randomly chosen message satisfies the sufficient condition is very small, and we need more effort to raise this probability.

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 129–145, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Message modification methods were proposed in [9] in order to generate a message satisfying the sufficient condition. By applying the combination of these methods, the probability where a message satisfies the sufficient condition is greatly improved. Consequently, Wang et al. claimed that their attack will succeed with probability 2^{-6} to 2^{-2}.

In this paper, we verify the efficiency of [9]. As a result, we found that there were three types of typos and oversights related to success probability.

1. Some messages are appropriately modified if they are modified solely, however, they are modified incorrectly if other specific messages are modified (Discussed in section 4.1.1).
2. Our modification method defined in [9] always fails (Discussed in section 4.1.2).
3. The sufficient condition proposed by Wang et al. does not cover all necessary conditions (Discussed in section 4.2).

Next, we consider these three situations and more precisely evaluate the success probability of their attack. We show that the success probability of the attack by Wang et al. is about $2^{-5.61}$ (Discussed in section 4.3.1). Second, we correct these typos and oversights, and then, we show this attack succeeds with probability 2^{-2} (Discussed in section 4.3.2). Third, we propose new message modification methods to satisfy the sufficient condition in the third round (Discussed in section 5). By combining these improvement, the success probability can be almost 1. The complexity of our attack can be reduced to 3 repetitions of MD4 hash operations. Our improved method is 85 times as fast as the attack by Wang et al. As far as we know, our attack is the most efficient attack of all the attacks proposed so far.

In EUROCRYPT 2005, Wang et al. proposed the collision attack for MD5, which is the similar attack for MD4 [10]. Since MD5 and MD4 have the similar structure, we expect that our improved method can also be applied to MD5. We will present details of the attack for MD5 later.

2 Description of MD4

MD4 is a function which compresses an arbitrary length message into a 128-bit hash value. The hash value of MD4 is calculated by following procedure:

1. Message M is divided into plural words $M_1, M_2, ..., M_n$, where the length of each message block is 512.
2. Let h_i be an output value of i^{th} block. h_i is calculated by using a compression function with h_{i-1} and M_i. This process is repeated until all M_i are calculated. The initial value h_0 is defined as follows;

$$h_0 = (0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476).$$

3. An output value of the last message block h_n will be the hash value of MD4.

Compression Function. A 512-bit word M_i inputted into the compression function is divided into 32-bit words $m_0, ..., m_{15}$. The compression function consists of 48 steps. Step1 to step16 are called as the first round, similarly, step17 to step32 are the second round and step33 to step48 are the third round. In each step, one of chaining variables a, b, c, d is calculated in order of $a_1, d_1, c_1, b_1, a_2, ...$ Only one variable is newly calculated in each step. The output of the compression function is $h_j = (a_0 + a_{16}, b_0 + b_{16}, c_0 + c_{16}, d_0 + d_{16})$.

We define a function ϕ as follows:

$$\phi(x, y, z, w, m, s, t) = ((x + f(y, z, w) + m + t) \mod 2^{32}) \lll s,$$

where m is a input message in each step, and which m_i is used is defined in advance. s is a number of bit rotation defined in each step. t is a constant defined in each round. In the first round, $t = 0$, in the second round, $t = 0x5a827999$ and in the third round, $t = 0x10325476$. The function f is defined as $f(y, z, w) = F(y, z, w)$ in the first round, $f(y, z, w) = G(y, z, w)$ in the second round and $f(y, z, w) = H(y, z, w)$ in the third round. The function $F(y, z, w)$, $G(y, z, w)$ and $H(y, z, w)$ are defined by,

$$\begin{aligned} F(y, z, w) &= (y \wedge z) \vee (\neg y \wedge w), \\ G(y, z, w) &= (y \wedge z) \vee (y \wedge w) \vee (z \wedge w), \\ H(y, z, w) &= y \oplus z \oplus w. \end{aligned}$$

Chaining variables a_i, b_i, c_i, d_i are calculated as follows:

$$\begin{aligned} a_i &= \phi(a_{i-1}, b_{i-1}, c_{i-1}, d_{i-1}, m, s, t) \\ d_i &= \phi(d_{i-1}, a_i, b_{i-1}, c_{i-1}, m, s, t) \\ c_i &= \phi(c_{i-1}, d_i, a_i, b_{i-1}, m, s, t) \\ b_i &= \phi(b_{i-1}, c_i, d_i, a_i, m, s, t) \end{aligned}$$

3 Attack of Wang et al. on MD4

An attack of Wang et al. is a differential attack using modular subtraction to calculate differential. This method makes a collision message by canceling input differential ΔM and making output differential be 0. Usually, the probability that the output differential of two different messages is 0 is very small. In [9], conditions are introduced to chaining variables in order to cancel the input differential.

For example, we consider the case that $\Delta m_1 = 1_{32}$ is given as differential of m_1. Here, 1_i denotes a 32-bit binary number whose i-th bit is 1 and other bits are 0. Because of this differential, the differential $0 \rightarrow 1$ occurs on the 7th bit of d_1, which is a variable calculated in step 2. In this paper, we describe the resulting value of d_1 as $d_1[7]$. In case that the 7th bit would change from 1 to 0, we describe it as $d_1[-7]$.

d_1 is used for calculation of a_2. The calculation for a_2 is as follows.

$$a_2 = (a_1 + F(b_1, c_1, d_1) + m_4) \lll 3$$

Therefore, the differential may be transmitted to a_2. Since this differential should be canceled, we focus on a property of F function. Since the expression of F in this step is $F(b_1, c_1, d_1) = (b_1 \wedge c_1) \vee (\neg b_1 \wedge d_1)$, if $b_{1,7} = 1$, the expression $F(b_1, c_1, d_1) = F(b_1, c_1, d_1[7])$ is true, and thus the change of d_1 can be ignored. Therefore, the differential can be canceled by introducing $b_{1,7} = 1$ in the sufficient condition.

By repeating this process for every differential to introduce all sufficient conditions, the input differential can be canceled. However, the probability that a randomly chosen message satisfies all sufficient conditions is very small. Therefore, it is necessary to try to raise the probability by message modification. In the next section, we explain the message modification.

3.1 Technique of Attack of Wang et al.

The attack of Wang et al uses following values as input differential:

$$\begin{aligned}
&\Delta M = M^* - M' = (\Delta m_0, \Delta m_1, ..., \Delta m_{15}) \\
&\Delta m_1 = 1_{32}, \Delta m_2 = 1_{32} - 1_{28}, \Delta m_{12} = -1_{17} \\
&\Delta m_i = 0, 0 \leq i \leq 15, i \neq 1, 2, 12
\end{aligned}$$

The number of sufficient conditions is 122. Almost all sufficient conditions are satisfied by message modification. However, message modification is adapted only in both the first round and the second round. Therefore, the sufficient condition in the third round is not always satisfied. In this case, it is necessary to rechoose random messages, and start message modification again. Rechosen messages are only m_{14} and m_{15}, and thus operations from step 1 to step 14 are saved. A procedure to generate a collision message pair (M', M^*) is as follows.

1. Select a 512-bit random message M.
2. Modify message M by using message modification to satisfy the sufficient condition.
3. If all sufficient conditions are satisfied, message M' which is a modified message from M is outputted, otherwise m_{14} and m_{15} are rechosen and go back to step 2.
4. Calculate a message $M^* = M' + \Delta M$, and output (M', M^*).

Procedure of Message Modification
In this section, we explain the process of step 2. The purpose of message modification is satisfying the sufficient condition. There are two types of modification techniques. One is single-step modification and the other is multi-step modification. Single-step modification is message modification for the first round. Since this modification does not affect other chaining variables, the single-step modification can be easily done. Multi-step modification is message modification for the second round. Since message modification in the second round affects not only the second round but also the first round, it is necessary to cancel the effect to the first round.

Single-Step Modification
Single-step modification modifies messages in order to satisfy the sufficient condition in the first round. We show an example of the single-step modification. Now, we consider the way to satisfy sufficient conditions in the third step ($c_{1,7} = 1$, $c_{1,8} = 1$, $c_{1,11} = 0$ and $c_{1,26} = d_{1,26}$). The calculation for c_1 is $c_1 = (c_0 + F(d_1, a_1, b_0) + m_2) \lll 11$. In this calculation, m_2 is the only variable which can be changed directly. Therefore, we satisfy sufficient conditions for the third step by modifying m_2 as follows;

$$c_1^{new} \leftarrow c_1 \oplus (c_{1,7} \oplus 1_7) \oplus (c_{1,8} \oplus 1_8) \oplus c_{1,11} \oplus (c_{1,26} \oplus d_{1,26})$$
$$m_2 \leftarrow (c_1^{new} \ggg 11) - c_0 - F(d_1, a_1, b_0).$$

Multi-Step Modification
Multi-step modification modifies messages in the second round in order to satisfy the sufficient condition in the second round. We show an example of the multi-step modification. Now, we consider the way to satisfy $a_{5,19} = c_{4,19}$ which is a sufficient condition in the 17th step. The calculation for a_5 is $a_5 = (a_4 + G(b_4, c_4, d_4) + m_0 + 0x5a827999) \lll 3$. In this calculation, m_0 is the only variable which can be changed directly. Therefore, if $a_{5,19} = c_{4,19}$ is not satisfied, we modify m_0 in order to satisfy this condition in the following way;

$$m_0 \leftarrow m_0 \pm 1_{16}.$$

The condition $a_{5,19} = c_{4,19}$ is satisfied by this modification. However, the change of m_0 causes changes of chaining variables in the first round because m_0 is used not only in the second round but also in the first round. The influence to the first round by modifying m_0 is shown as follows.

$$m_0 \leftarrow m_0 \pm 1_{16} \Rightarrow a_1^{new} = a_1[\pm 19]$$

This influence causes changes of variables which have already satisfied the sufficient condition. Therefore, it is necessary to add following modifications in order to prevent the influence of the change from transmitting to variables in the first round.

$$m_0 \leftarrow m_0 \pm 1_{16} \Rightarrow a_1^{new} = a_1[\pm 19]$$
$$m_1 \leftarrow (d_1 \ggg 7) - d_0 - F(a_1^{new}, b_0, c_0)$$
$$m_2 \leftarrow (c_1 \ggg 11) - c_0 - F(d_1, a_1^{new}, b_0)$$
$$m_3 \leftarrow (b_1 \ggg 19) - b_0 - F(c_1, d_1, a_1^{new})$$
$$m_4 \leftarrow (a_2 \ggg 3) - a_1^{new} - F(b_1, c_1, d_1)$$

By this modification, the influence to the first round is canceled. In this paper, we call this modification as multi-step modification(1).

However, there exists some sufficient conditions in the second round which are failed to be satisfied by the multi-step modification(1). We show an example of this situation. Now, we consider $c_{5,26} = d_{5,26}$ which is a sufficient condition

Table 1. Correction method of $C_{5,27}$

step	shift	Modify m_i	Chaining value after message modification	Extra Conditions
6	7	$m_5 \leftarrow m_5 + 1_{11}$	$d_2[18], a_2, b_1, c_1$	$d_{2,18} = 0$
7	11		$c_2, d_2[18], a_2, b_1$	$a_{2,18} = b_{1,18}$
8	19		$b_2, c_2, d_2[18], a_2$	$c_{2,18} = 0$
9	3	$m_8 \leftarrow m_8 - 1_{18}$	$a_3, b_2, c_2, d_2[18]$	$b_{2,18} = 0$
10	7	$m_9 \leftarrow m_9 - 1_{18}$	d_3, a_3, b_2, c_2	

in step 19. If we try to apply the multi-step modification(1), the change of the message affects the first round as follows;

$$m_8 \leftarrow m_8 \pm 1_{18} \Rightarrow a_3^{new} = a_3[\pm 21].$$

Since there exists a sufficient condition about $a_{3,21}$, this modification method breaks the sufficient condition on $a_{3,21}$, and thus we failed to find collision messages. Therefore, we try to apply another message modification method shown in Table 1 in order to keep $a_{3,21}$ unchanged. In Table 1, "extra condition" means conditions of chaining variables introduced to stop the influence in the second round from transmitting to the first round. The extra condition is set to be satisfied when the first round is calculated.

If this modification is applied, the value of the 18th bit of function F in step 9 is changed as follows;

$$F(b_{2,18}, c_{2,18}, d_{2,18}) = 0 \rightarrow 1.$$

The influence to a_3 caused by modifying message m_8 can be canceled by the change of $F(b_{2,18}, c_{2,18}, d_{2,18})$. Therefore, the condition on $c_{5,26}$ is satisfied and the value of $a_{3,20}$ is unchanged by this modification. In this paper, we call this type of modification as multi-step modification(2).

4 Exact Evaluation of the Method of Wang et al.

In [9], Wang et al. claimed that success probability of their collision attack on MD4 is 2^{-6} to 2^{-2}. However, this success probability is not enough precise. First, we examined details of the method of Wang et al, and evaluate precise probability.

4.1 Oversights in the Method of Wang et al.

Wang et al. proposed various message modification methods. However, as a result of our analysis, we found some of their modifications could not satisfy the sufficient condition. We describe the reason why they could not satisfy the sufficient condition and how to improve the method of Wang et al.

Exclusive Modifications

If some messages are simultaneously modified in order to satisfy specific two sufficient conditions, one modification can conflict the other modification. These cases are a pair of corrections for $d_{5,19} = a_{5,19}$ and $c_{5,26} = d_{5,26}$ and a pair of corrections for $c_{5,32} = d_{5,32}$ and $c_{6,32} = d_{6,32}$. We describe the reason why these corrections are failed, and propose an improved method. .

(1) Problems

First, we explain correction of $d_{5,19} = a_{5,19}$ and $c_{5,26} = d_{5,26}$. Wang et al. used the multi-step modification(1) to correct the condition on $d_{5,19}$. In this case, m_4 is modified such as $m_4 \leftarrow m_4 + 1_{14}$. This modification causes change of the value of $a_{2,17}$. Whereas, Wang et al. used the multi-step modification(2) to correct the condition on $c_{5,26}$. In this case, it is necessary to add an extra condition $a_{2,17} = b_{1,17}$. However, if both modification is done, the value of $a_{2,17}$ is changed because of the correction of $d_{5,19}$, and then the extra condition $a_{2,17} = b_{1,17}$ is broken. This extra condition is added in order to cancel the influence of change in the second round. Therefore, if the extra condition is broken, the value of c_2 may change into the different value from desired one.

(2) Success Probability of the Correction

The above problem occurs when both of $d_{5,19}$ and $c_{5,26}$ are corrected. This probability is $1/4$. However, if $d_{6,26}$ is corrected additionally, the correction of $d_{5,19}$ and $c_{5,26}$ works appropriately. We explain this mechanism.

When $d_{5,19}$ and $c_{5,26}$ are corrected, and additionally, $d_{6,26}$ is corrected, $c_{5,26}$ is appropriately corrected. The problem is that extra condition $a_{2,17} = b_{1,17}$ is broken, and c_2 will be unexpectedly changed from this influence. However, while $d_{6,26}$ is corrected, m_6 is modified as follows;

$$m_6 \leftarrow (c_2 \ggg 11) - c_1 - f(d'_2, a_2, b_1).$$

The purpose of this modification is keeping the value of c_2 unchanged. Therefore, the value of c_2 is kept unchanged even if the extra condition $a_{2,17} = b_{1,17}$ is broken. Therefore, if $d_{5,19}$ and $c_{5,26}$ are corrected and furthermore $d_{6,26}$ is corrected, $c_{5,26}$ is appropriately corrected. Therefore, the success probability of the correction of both $d_{5,19}$ and $c_{5,26}$ is given by

Table 2. Correction method of $d_{5,19}$

step	shift	Modify m_i	Chaining value after message modification	Extra Conditions
2	7	$m_1 \leftarrow m_1 + 1_7$	$d_1[14], a_1, b_0, c_0$	$d_{1,14} = 0$
3	11		$c_1, d_1[14], a_1, b_0$	$a_{1,14} = b_{0,14}$
4	19		$b_1, c_1, d_1[14], a_1$	$c_{1,14} = 0$
5	3	$m_4 \leftarrow m_4 - 1_{14}$	$a_2, b_1, c_1, d_1[14]$	$b_{1,14} = 0$
6	7	$m_5 \leftarrow m_5 - 1_{14}$	d_2, a_2, b_1, c_1	

Table 3. Correction method of $c_{5,32}$

step	Modify m_i	Chaining value after message modification	Extra Conditions
15	$m_{14} \leftarrow m_{14} + 1_{11}$	$c_4[22], d_4, a_4, b_3$	$c_{4,22} = 0$
16		$b_4, c_4[22], d_4, a_4$	$d_{4,22} = a_{4,22}$
17		$a_5, b_4, c_4[22], d_4$	$b_{4,22} = d_{4,22}$
18		$d_5, a_5, b_4, c_4[22]$	$a_{5,22} = b_{4,22}$
19		$c_5^{new} = c_5 + 1_{31}, d_5, a_5, b_4$	$c_{5,31} = 1$

$$1 - (\frac{1}{4} \times \frac{1}{2}) = \frac{7}{8}.$$

Similar situation occurs when both of $c_{5,32}$ and $c_{6,32}$ are corrected. In this situation, the success probability is 1/4.

(3) Improve Method

Improved Correction Methods of $d_{5,19}$ and $c_{5,26}$

Since the method of Wang et al. contains problems explained above, we propose the method modifying $d_{5,19}$ by the multi-step modification(2) in order to avoid the problems. Details of this modification method is shown in Table 2. As a result, the correction of $d_{5,19}$ does not break the extra condition $a_{2,17} = b_{1,17}$, and thus, the success probability of corrections of $d_{5,19}$ and $c_{5,26}$ becomes 1.

Improved Correction Methods of $c_{5,32}$ and $c_{6,32}$

Since the method of Wang et al. contains problems explained above, we change the correction method of $c_{5,32}$ as shown in Table 3 so that this correction method does not cause problems when both of $c_{5,32}$ and $c_{6,32}$ are corrected.

In the case we change the correction method of $c_{5,32}$, other changes except for Table 3 seem to be possible, for example, modifying message used in 19th step. However, all of other changes which we tried caused influence to other conditions. Therefore, we decided to use carry in the 31st bit in order to change the value of $c_{5,32}$. This modification does not break other conditions. Therefore, the success probability of the corrections of $c_{5,32}$ and $c_{6,32}$ becomes 1.

A Modification Breaking the Sufficient Condition

(1) Problems

In [10], $c_{5,29}$ is modified by using the multi-step modification(2). This modification changes the value of $d_{2,20}$ from 0 to 1. However, there exists a sufficient condition $d_{2,20} = a_{2,20}$. Therefore, the modification of $c_{5,29}$ breaks this condition and we fail to get collision messages.

(2) Improved method

c_5 is calculated as follows.

$$c_5 = (c_4 + G(d_5, a_5, b_4) + m_8 + 0x5a827999) \lll 9$$

Table 4. Correction of $c_{5,29}$

step	Modify m_i	Chaining value after message modification	Extra Conditions
15	$m_{14} \leftarrow m_{14} + 1_9$	$c_4[20], d_4, a_4, b_3$	$c_{4,20} = 0$
16		$b_4, c_4[20], d_4, a_4$	$d_{4,20} = a_{4,20}$
17		$a_5, b_4, c_4[20], d_4$	$b_{4,20} = d_{4,20}$
18		$d_5, a_5, b_4, c_4[20]$	$a_{5,20} = b_{4,20}$
19		$c_5^{new} = c_5 + 1_{29}, d_5, a_5, b_4$	

If we modify m_8 in order to satisfy a sufficient condition $c_{3,29} = 1$, it causes influence to other variables. Therefore, we change the value of $c_{4,20}$ instead. c_4 is calculated by following expression,

$$c_4 = (c_3 + F(d_4, a_4, b_3) + m_{14}) \lll 11.$$

Therefore, we can change $c_{4,20}$ by changing $m_{14,9}$. This correction does not cause influence to other variables, and thus the success probability of the correction of $c_{5,29}$ becomes 1. Table 4 shows details of the correction of $c_{5,29}$

4.2 Lack of the Sufficient Condition

We carefully checked the sufficient condition from input differential, and found that conditions $a_{6,30} = 0$ and $b_{4,32} = c_{4,32}$ are lacked in the table 5 written in [9]. In this section, we show how we found these conditions.

Table 5. Introduction of "$a_{6,30} = 0$"

Step	Shift	Δm_i	The i-th output for M'
21	3	2^{31}	$a_6[-29, 30, -32]$

Introduction of a Sufficient Condition "$a_{6,30} = 0$"
At the first,we explain how to find "$a_{6,30} = 0$". Table 5 is a part of table 5 written in [9]. $a_6[30]$ in the Table 5 means that the value of $a_{6,30}$ changes from 0 to 1 because of differential. Therefore, we need to set the sufficient condition $a_{6,30} = 0$ in advance. The correction method of $a_{6,30}$ is the same with other correction methods of a_6.

Introduction of a Sufficient Condition "$b_{4,32} = c_{4,32}$"
Table 6 shows that the value of $a_{5,32}$ is changed because of differential, but d_5 is not affected by differential. Therefore, when a_5 is calculated, the differential $a_5[-32]$ has to be canceled. We explain how to cancel this differential. d_5 is calculated by

$$d_5 = ((d_4 + G(a_5, b_4, c_4) + m_4 + 0x5a827999) \bmod 2^{32}) \lll 5,$$

Table 6. Introduction of "$b_{4,32} = c_{4,32}$"

Step	Shift	Δm_i	The i-th output for M'
14	7		$d_4[-27, -29, 30]$
15	11		c_4
16	19		$b_4[19]$
17	3		$a_5[-26, 27, -29, -32]$
18	5		d_5

where the function G is defined as $G(y, z, w) = (y \wedge z) \vee (y \wedge w) \vee (z \wedge w)$. Since the function G uses a_5, we can cancel differential by arranging other variables used in G. Since G is a majority function, the differential $a_{5,32}$ is canceled by constructing condition $b_{4,32} = c_{4,32}$.

4.3 Reevaluation of Success Probability of the Method of Wang et al. and Our Improved Method

Precise Probability Evaluation of the Method of Wang et al.
Considered all the problems explained so far, precise probability of [9] is as follows,

- Success probability considered lack of sufficient conditions $b_{4,32} = c_{4,32}$ and $a_{6,30} = 0$ (section4.2): $\left(\frac{1}{2}\right)^2$
- Success probability considered wrong correction of $c_{5,29}$(section4.2):$\frac{1}{2}$
- Success probability considered problems of corrections of $d_{5,19}$ and $c_{5,26}$ and corrections of $c_{5,32}$ and $c_{6,32}$(section4.1.2):$\frac{7}{8} \times \frac{3}{4}$
- Probability satisfying the sufficient condition in the third round (section4.1.1):$\left(\frac{1}{2}\right)^2$

By combining above analyses, the precise success probability of [9] is given as follows:

$$\begin{aligned} Pr[success] &= \frac{3}{4} \times \frac{7}{8} \times \frac{1}{2} \times \left(\frac{1}{2}\right)^2 \times \left(\frac{1}{2}\right)^2 \\ &= 2^{-5.61} \end{aligned}$$

Furthermore, we experimentally verified this evaluation. In our experiment, we generated 5,000 collision messages by the method of Wang et al, and the average number where we needed to rechoose random messages to find a collision was 48.76 times($\fallingdotseq 2^{5.60}$ times). Therefore, the theoretical result is consistent with the experimental result.

Our Improved Method
We improved oversights and typos of [9] as follows,

- Add two sufficient conditions $b_{4,32} = c_{4,32}$ and $a_{6,30} = 0$
- Change the modification method of $d_{5,19}$ so that both of $d_{5,19} = a_{5,19}$ and $c_{5,26} = d_{5,26}$ can be corrected.

– Change the modification method of $c_{5,32}$ so that both of $c_{5,32} = d_{5,32}$ and $c_{6,32} = d_{6,32} + 1$ can be corrected.

By our improvement, all sufficient conditions in both the first round and the second round can be corrected with probability 1. Therefore, the success probability becomes exactly 2^{-2}.

In [9], details of the multi-step modification is not written. In this paper, we attach the list of all multi-step modification methods in appendix.

Table 7. Shortcut Modification

step	Modify m_j	Chaining value after message modification	Extra Conditions in 1^{st} round
12	$m_{11} \leftarrow m_{11} \pm 1_i$	$b_3[\pm(i+19)], c_3, d_3, a_3$	
13		$a_4, b_3[\pm(i+19)], c_3, d_3$	$c_{3,i+19} = d_{3,i+19}$
14		$d_4, a_4, b_3[\pm(i+19)], c_3$	$a_{4,i+19} = 0$
15		$c_4, d_4, a_4, b_3[\pm(i+19)]$	$d_{4,19} = 0$
16	$m_{15} \leftarrow (b_4 \ggg 19) - b_3[\pm(i+19)] - F(c_4, d_4, a_4)$	b_4, c_4, d_4, a_4	

5 Shortcut Modification

5.1 Our Idea

In the case of multi-step modification, only the first round is affected by message modifications. Therefore, to make the multi-step modification succeed, only the first round has to be considered. If we try to apply same approach to modify messages in the third round, we have to consider the influence of the modification about both the first round and the second round. This is very difficult. Actually, the message modification for the third round is not mentioned in [9]. To overcome this difficulty, we randomly modify messages in the second round many times in order to satisfy the sufficient condition in the third round. In the method of Wang et al, if collision message cannot be generated, algorithm returns to the calculation for the first round, rechooses m_{14} and m_{15} and apply the multi-step modification in the second round. In the proposed method, we change few bits of a message in the second round instead of rechoosing random messages. Therefore, the complexity of each repetition is reduced by the complexity for the multi-step modification.

5.2 Consideration

Here, we consider to modify m_{11}. To begin with, we experimentally found probability that the sufficient condition in the third round is satisfied by modifying m_{11} with 1 bit. Hence, we can assume that conditions are satisfied with probability about $1/4$ for all single bit change. When we modify m_{11} in the second round, we have to cancel the influence to the first round caused by modification of m_{11}.

We apply message modification shown in Table 7 to cancel the influence from m_{11}. The number of i, where the extra conditions does not break the sufficient condition is 19. If we simultaneously change several bits of m_{11}, we can similarly assume that conditions are satisfied with probability 1/4. Therefore, the number of repetition by modifying m_{11} is equal to the number of all combination of 19 bits, that is, 2^{19}. If we modify m_{11} many times, probability to satisfy the sufficient conditions in the third round becomes very high. This probability is given by

$$1 - \left(1 - \frac{1}{4}\right)^{2^{19}} \fallingdotseq 1.$$

Therefore, the probability that the sufficient condition in the third round becomes almost 1. Since this probability 1/4 for each m_{11}, the average number that we change m_{11} to satisfy conditions in the third round is three. (First time, we calculate m_{11} in the standard way. Then, if we repeat the process three times, four of different m_{11} would be tried.)

Remark 1
The reason we chose m_{11} is that since m_{11} is the second last message in the second round, the number of steps of each repetition is small compared to other messages. The reason we did not choose m_{15} is that we could not cancel the influence to the first round caused by m_{15}.

6 Our Improvement

In section 4 and 5, we proposed the message modification which satisfies the sufficient condition with probability almost 1. The followings are our improvement,

- We add the sufficient condition $b_{4,32} = c_{4,32}$ and $a_{6,30} = 0$
- We change the message modification method of $d_{5,19}$ in order to enable both $d_{5,19} = a_{5,19}$ and $c_{5,26} = d_{5,26}$ to be modified.
- We change the message modification method of $c_{5,32}$ in order to enable both $c_{5,32} = d_{5,32}$ and $c_{6,32} = d_{6,32} + 1$ to be modified.
- We propose the shortcut modification which raise the probability satisfying the sufficient condition in the third round.

In our improved method, collision message can be generated with probability almost 1. The complexity of our method is given by,

- 1 MD4 operation to get hash value of the message in standard calculation
- Few steps for the single-step modification
- 26 steps for the multi-step modification and 13 steps for recalculation of chaining variables after the multi-step modification is applied

- 24 steps for modification of m_{11} to satisfy the sufficient condition in the third round (the average number of repetition of changing m_{11} is 3 (section5.2) and each repetition needs 8 steps)
- Few steps to cancel the influence of m_{11}

Therefore, total complexity is 1 MD4 + 26 steps + 13 steps + 24 steps + few steps = 1 MD4 + (63 + few) steps. Since one MD4 operation consists of 48 steps, This complexity is less than three repetitions of MD4 operations. Consequently, collision messages can be generated with less than three repetitions of MD4 operations.

The method of Wang et al. finds collision with complexity less than 2^8 MD4 operations. Whereas, the complexity of our method is 3 MD4 operations. Therefore, our method is $2^8/3 \fallingdotseq 85$ times as fast as the method of Wang et al.

Remark 2
It is possible to reduce the complexity of the first round by randomly generating chaining variables in the first round instead of messages. First, set each chaining variables in the first round to satisfy the sufficient condition. Second, randomly generate other bits. Finally, calculate the message from generated chaining variables. This process reduces the complexity of the first round.

Table 8. An example of generated collision pair

M	0x24ce9d37de4dfca0a3b88fc39c9f9e5c92ee86ada2c9e8b088f3a020c5368a69 0e503cc80c2368f978ff57bf21a1762ad018afb8daa431e9308bf382806a18a1
M'	0x368b9d377e2dfc60b5b88fcb0c8fbe5601a6662d9ecc3929aa35aabf887f929f 2740a2c8c8c12039bbb401bdc1983331e45e1f61c150d565ee27d04af1dfec4c
M^*	0x368b9d37fe2dfc6025b88fcb0c8fbe5601a6662d9ecc3929aa35aabf887f929f 2740a2c8c8c12039bbb401bdc1983331e45d1f61c150d565ee27d04af1dfec4c
$H(M')$	0x26a280327c3068532de33b679d022e59
$H(M^*)$	0x26a280327c3068532de33b679d022e59

7 Conclusion

Wang et al. claimed that collisions of MD4 can be generated with complexity less than 2^8 MD4 hash operations. However, this evaluation was not enough precise. In this paper, we reevaluated the success probability of the method of Wang et al. Then, we correct typos and oversights of [9]. As a result, the attack succeeded with probability 2^{-2}. At the last, we proposed the method to satisfy the sufficient condition in the third round. As a result, the complexity to generate collision messages was reduced to 3 repetitions of MD4 operations.

We show a message, where the method of Wang et al. cannot generate a collision pair. In Table 8, M is the message before modified, M' is the message after modified and M^* is calculated by $M^* = M' + \Delta M$. $H(M')$ is the hash value of M' and $H(M^*)$ is the hash value of M^*.

References

1. B. den Boer, A. Bosselaers: Collisions for the compression function of MD5, EUROCRYPT'93, 1993
2. E. Biham, R. Chen: Near collision of SHA-0, CRYPTO2004, LNCS 3152, pp290–305, 2004.
3. E. Biham, R. Chen, A. Joux, P. Carribault: Collisions of SHA-0 and Reduced SHA-1, EUROCRYPT2005, LNCS3494, pp.36–57, 2005.
4. F. Charband, A. Joux: Differential Collisions in SHA-0, CRYPTO'98, 1998
5. H. Dobbertin: Cryptanalysis of MD4, First Software Encryption 1996, LNCS 1039, 1996
6. H. Dobbertin: The First Two Rounds of MD4 are Not One-Way, Fast Software Encryption 1998, 1998.
7. R. Rivest: The MD4 Message Digest Algorithm, CRYPTO'90 Proceedings, 1991, http://theory.lcs.mit.edu/~rivest/Rivest-MD4.txt
8. X. Wang, D. Feng, X. Lai, H. Yu: Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, rump session, CRYPTO 2004, e-Print, 2003.
9. X. Wang, X. Lai, D. Feng, H. Chen, X. Yu: Cryptanalysis of the Hash Functions MD4 and RIPEMD, Advances in EUROCRYPT2005, LNCS 3494, pp. 1–18, 2005.
10. X. Wang, H. Yu: How to break MD5 and Other Hash Functions, Advances in EUROCRYPT2005, LNCS 3494, pp. 19–35, 2005.

Appendix(*List of Multi-step Modification Methods*)

Table 9. Modification of "$a_{5,i}$" $(i = 19, 26, 27, 29, 32)$

step	shift	Modify m_i	Chaining value after after message modification
1	3	$m_0 \leftarrow m_0 \pm 1_{i-3}$	$a_1^{new} = a_1[\pm i], b_0, c_0, d_0$
2	7	$m_1 \leftarrow (d_1 \ggg 7) - d_0 - F(a_1^{new}, b_0, c_0)$	d_1, a_1^{new}, b_0, c_0
3	11	$m_2 \leftarrow (c_1 \ggg 11) - c_0 - F(d_1, a_1^{new}, b_0)$	c_1, d_1, a_1^{new}, b_0
4	19	$m_3 \leftarrow (b_1 \ggg 19) - b_0 - F(c_1, d_1, a_1^{new})$	b_1, c_1, d_1, a_1^{new}
5	3	$m_4 \leftarrow (a_2 \ggg 3) - a_1^{new} - F(b_1, c_1, d_1)$	a_2, b_1, c_1, d_1

Table 10. Modification of "$d_{5,19}$"

step	shift	Modify m_i	Chaining value after message modification	Extra Conditions
2	7	$m_1 \leftarrow m_1 + 1_7$	$d_1[14], a_1, b_0, c_0$	$d_{1,14} = 0$
3	11		$c_1, d_1[14], a_1, b_0$	$a_{1,14} = b_{0,14}$
4	19		$b_1, c_1, d_1[14], a_1$	$c_{1,14} = 0$
5	3	$m_4 \leftarrow m_4 - 1_{14}$	$a_2, b_1, c_1, d_1[14]$	$b_{1,14} = 0$
6	7	$m_5 \leftarrow m_5 - 1_{14}$	d_2, a_2, b_1, c_1	

Table 11. Modification of "$d_{5,i}$" $i = 16, 17, 29, 32$

step	shift	Modify m_i	Chaining value after message modification
5	3	$m_4 \leftarrow m_4 \pm 1_{i-5}$	$a_2^{new} = a_2[\pm(i-2)], b_1, c_1, d_1$
6	7	$m_5 \leftarrow (d_2 \ggg 7) - d_1 - F(a_2^{new}, b_1, c_1)$	d_2, a_2^{new}, b_1, c_1
7	11	$m_6 \leftarrow (c_2 \ggg 11) - c_1 - F(d_2, a_2^{new}, b_1)$	c_2, d_2, a_2^{new}, b_1
8	19	$m_7 \leftarrow (b_2 \ggg 19) - b_1 - F(c_2, d_2, a_2^{new})$	b_2, c_2, d_2, a_2^{new}
9	3	$m_8 \leftarrow (a_3 \ggg 3) - a_2^{new} - F(b_2, c_2, d_2)$	a_3, b_2, c_2, d_2

Table 12. Modification of "$c_{5,i}$" $i = 26, 27$

step	shift	Modify m_i	Chaining value after message modification	Extra Conditions
6	7	$m_5 \leftarrow m_5 + 1_{i-16}$	$d_2[i-9], a_2, b_1, c_1$	$d_{2,i-9} = 0$
7	11		$c_2, d_2[i-9], a_2, b_1$	$a_{2,i-9} = b_{1,i-9}$
8	19		$b_2, c_2, d_2[i-9], a_2$	$c_{2,i-9} = 0$
9	3	$m_8 \leftarrow m_8 - 1_{i-9}$	$a_3, b_2, c_2, d_2[i-9]$	$b_{2,i-9} = 0$
10	7	$m_9 \leftarrow m_9 - 1_{i-9}$	d_3, a_3, b_2, c_2	

Table 13. Modification of "$c_{5,29}$"

step	shift	Modify m_i	Chaining value after message modification	Extra Conditions
15	11	$m_{14} \leftarrow m_{14} + 1_9$	$c_4[20], d_4, a_4, b_3$	$c_{4,20} = 0$
16	19		$b_4, c_4[20], d_4, a_4$	$d_{4,20} = a_{4,20}$
17	3		$a_5, b_4, c_4[20], d_4$	$b_{4,20} = d_{4,20}$
18	5		$d_5, a_5, b_4, c_4[20]$	$a_{5,20} = b_{4,20}$
19	9		$c_5^{new} = c_5 + 1_{29}, d_5, a_5, b_4$	

Table 14. Modification of "$c_{5,30}$"

step	shift	Modify m_i	Chaining value after after message modification
9	3	$m_8 \leftarrow m_8 \pm 1_{21}$	$a_3^{new} = a_3[\pm 24], b_2, c_2, d_2$
10	7	$m_9 \leftarrow (d_3 \ggg 7) - d_2 - F(a_3^{new}, b_2, c_2)$	d_3, a_3^{new}, b_2, c_2
11	11	$m_{10} \leftarrow (c_3 \ggg 11) - c_2 - F(d_3, a_3^{new}, b_2)$	c_3, d_3, a_3^{new}, b_2
12	19	$m_{11} \leftarrow (b_3 \ggg 19) - b_2 - F(c_3, d_3, a_3^{new})$	b_3, c_3, d_3, a_3^{new}
13	3	$m_{12} \leftarrow (a_4 \ggg 3) - a_3^{new} - F(b_3, c_3, d_3)$	a_3, b_3, c_3, d_3

Table 15. Modification of "$c_{5,31}$"

step	shift	Modify m_i	Chaining value after message modification	Extra Conditions
15	11	$m_{14} \leftarrow m_{14} + 1_{11}$	$c_4[22], d_4, a_4, b_3$	$c_{4,22} = 0$
16	19		$b_4, c_4[22], d_4, a_4$	$d_{4,22} = a_{4,22}$
17	3		$a_5, b_4, c_4[22], d_4$	$b_{4,22} = d_{4,22}$
18	5		$d_5, a_5, b_4, c_4[22]$	$a_{5,22} = b_{4,22}$
19	9		$c_5^{new} = c_5 + 1_{31}, d_5, a_5, b_4$	$c_{5,31} = 1$

Table 16. Modification of "$b_{5,i}$" $i = 29, 32$

step	shift	Modify m_i	Chaining value after message modification	Extra Conditions
11	11	$m_{10} \leftarrow m_{10} + 1_{i-24}$	$c_3[-(i-13)], d_3, a_3, b_2$	$c_{3,i-13} = 0$
12	19		$b_3, c_3[-(i-13)], d_3, a_3$	$d_{3,i-13} = a_{3,i-13}$
13	3	$m_{12} \leftarrow m_{12} - 1_{i-13}$	$a_4, b_3, c_3[-(i-13)], d_3$	$b_{3,i-13} = 1$
14	7		$d_4, a_4, b_3, c_3[-(i-13)]$	$a_{4,i-13} = 1$
15	11	$m_{14} \leftarrow m_{14} - 1_{i-13}$	c_4, d_4, a_4, b_3	

Table 17. Modification of "$b_{5,30}$"

step	shift	Modify m_i	Chaining value after message modification	Extra Conditions
12	19	$m_{11} \leftarrow m_{11} + 1_{30}$	$b_3[17], c_3, d_3, a_3$	$b_{3,17} = 0$
13	3	$m_{12} \leftarrow m_{12} - 1_{17}$	$a_4, b_3[17], c_3, d_3$	
14	7		$d_4, a_4, b_3[17], c_3$	$a_{4,17} = 0$
15	11		$c_4, d_4, a_4, b_3[17]$	$d_{4,17} = 1$
16	19	$m_{15} \leftarrow m_{15} - 1_{17}$	b_4, c_4, d_4, a_4	

Table 18. Modification of "$a_{6,i}$" $i = 29, 30, 32$

step	shift	Modify m_i	Chaining value after after message modification	Extra Conditions
2	7	$m_1 \leftarrow m_1 \pm 1_{i-3}$	$d_1^{new} = d_1[\pm(i+4)], a_1, b_0, c_0$	
3	11	$m_2 \leftarrow (c_1 \ggg 11) - c_0 - F(d_1^{new}, a_1, b_0)$	c_1, d_1^{new}, a_1, b_0	
4	19	$m_3 \leftarrow (b_1 \ggg 19) - b_0 - F(c_1, d_1^{new}, a_1)$	b_1, c_1, d_1^{new}, a_1	
5	3		a_2, b_1, c_1, d_1^{new}	$b_{1,i+4} = 1$
6	7	$m_5 \leftarrow (d_2 \ggg 7) - d_1^{new} - F(a_2, b_1, c_1)$	d_2, a_2, b_1, c_1	

Table 19. Modification of "$d_{6,29}$"

step	shift	Modify m_i	Chaining value after after message modification	Extra Conditions
6	7	$m_5 \leftarrow m_5 \pm 1_{24}$	$d_2^{new} = d_2[\pm 31], a_2, b_1, c_1$	
7	11	$m_6 \leftarrow (c_2 \ggg 11) - c_1 - F(d_2^{new}, a_2, b_1)$	c_2, d_2^{new}, a_2, b_1	
8	19	$m_7 \leftarrow (b_2 \ggg 19) - b_1 - F(c_2, d_2^{new}, a_2)$	b_2, c_2, d_2^{new}, a_2	
9	3		a_3, b_2, c_2, d_2^{new}	$b_{2,31} = 1$
10	7	$m_9 \leftarrow (d_3 \ggg 7) - d_2^{new} - F(a_3, b_2, c_2)$	d_3, a_3, b_2, c_2	

Table 20. Modification of "$c_{6,i}$" $i = 29, 30$

step	shift	Modify m_i	Chaining value after after message modification	Extra Conditions
10	7	$m_9 \leftarrow m_9 \pm 1_{i-9}$	$d_3^{new} = d_3[\pm(i-2)], a_3, b_2, c_2$	
11	11	$m_{10} \leftarrow (c_3 \ggg 11) - c_2 - F(d_3^{new}, a_3, b_2)$	c_3, d_3^{new}, a_3, b_2	
12	19	$m_{11} \leftarrow (b_3 \ggg 19) - b_2 - F(c_3, d_3^{new}, a_3)$	b_3, c_3, d_3^{new}, a_3	
13	3		a_4, b_3, c_3, d_3^{new}	$b_{3,i-2} = 1$
14	7	$m_{13} \leftarrow (d_4 \ggg 7) - d_3^{new} - F(a_4, b_3, c_3)$	d_4, a_4, b_3, c_3	

Table 21. Modification of "$c_{6,32}$"

step	shift	Modify m_i	Chaining value after after message modification	Extra Conditions
7	11	$m_6 \leftarrow m_6 \pm 1_{12}$	$c_2[23], d_2, a_2, b_1$	$c_{2,23} = 0$
8	19		$b_2, c_2[23], d_2, a_2$	$d_{2,23} = a_{2,23}$
9	3		$a_3, b_2, c_2[23], d_2$	$b_{2,23} = 0$
10	7	$m_9 \leftarrow m_9 - 1_{23}$	$d_3, a_3, b_2, c_2[23]$	
11	11	$m_{10} \leftarrow m_{10} - 1_{23}$	c_3, d_3, a_3, b_2	

Finding Collision on 45-Step HAS-160

Aaram Yun[1], Soo Hak Sung[2], Sangwoo Park[1], Donghoon Chang[3], Seokhie Hong[3], and Hong-Su Cho[4]

[1] National Security Research Institute,
161 Gajeong-dong, Yuseong-gu, Daejeon 305-350, Korea
{aaram, psw}@etri.re.kr
[2] Department of Computing information & mathematics, Paichai University,
426-6 Doma-dong, Seo-gu, Daejeon 302-735, Korea
sungsh@mail.pcu.ac.kr
[3] Center for Information and Security Technologies, Korea University, Seoul, Korea
{dhchang, hsh}@cist.korea.ac.kr
[4] Graduate School of Information Security, Korea University
1, 5-Ka, Anam-dong, Sungbuk-ku, Seoul 136-701, Korea
karma3432@korea.ac.kr

Abstract. HAS-160 is a cryptographic hash function designed and used widely in Korea. While similar in structure to SHA-1, up to now there was no published attack or security analysis of the algorithm. Applying techniques introduced by Wang et al.[1], we have found collision in the first 45 steps of HAS-160, with complexity 2^{12}.

1 Introduction

HAS-160 is a cryptographic hash function standardized by Korean government (TTAS.KO-12.0011/R1)[3]. It is a Merkle-Damgård hash function, having a compression function whose input length is 512 and output length 160. The overall structure is similar to SHA-1, with some modifications, like simpler message scheduling and variable amounts of bit rotation.

Despite its wide use in Korea, so far there was no published evaluation of its cryptographic strength. In the light of the recent advances of analysis of dedicated hash functions by Wang et al.[1, 2, 5, 6], we feel that it is important to analyze the strength of HAS-160, especially since HAS-160 shares many design elements of MD5 and SHA-1.

In this paper we apply the techniques introduced in [1] to HAS-160, and produce collision pairs for the first 45 steps (out of the total 80) of HAS-160. We believe that this could serve as a starting point for further cryptographic analysis of the hash function.

2 Structure of HAS-160

HAS-160 is a dedicated hash algorithm following the Merkle-Damgård construction. Therefore, for our purposes here we are going to omit other non-relevantl definitions and describe only the compression function

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 146–155, 2006.
© Springer-Verlag Berlin Heidelberg 2006

$$(a', b', c', d', e') = \mathcal{H}\left(a, b, c, d, e, (x_i)_{i=0}^{15}\right).$$

- Input/output: the compression function $\mathcal{H}$ takes five 32-bit values a, b, c, d, e, and sixteen 32-bit words x_0, x_1, ..., x_{15} as input, and produces five 32-bit values a', b', c', d', e' as output.
- Initial values: the compression function $\mathcal{H}$ takes as initial values the following:
  - a=0x67452301
  - b=0xefcdab89
  - c=0x98badcfe
  - d=0x10325476
  - e=0xc3d2e1f0

  If the 512-bit message block $X = (x_i)_{i=0}^{15}$ is not the first block to process, then the previous output of $\mathcal{H}$ is used as input.
- Rounds and steps: starting with the original input a, b, c, d, e, and $(x_i)_{i=0}^{15}$, $\mathcal{H}$ takes 80 steps of transformations to the variables a, ..., e to produce the output. The steps are organized into four rounds, where one round consists of 20 steps. Steps are indexed from 0, and we denote by a_i, b_i, c_i, d_i, e_i the values of the variables a, ..., e at the beginning of the step i. a_{80}, ..., e_{80} are the values at the end of the step 79.
- Output: after undergoing the 80 steps, the final output of $\mathcal{H}$, (a', b', c', d', e'), is computed by $(a_0 + a_{80}, b_0 + b_{80}, \ldots, e_0 + e_{80})$, where $+$ is the addition modulo 2^{32}.
- Message scheduling: each step of transformation involves a message word. 20 words x_0, ..., x_{19} are used in a round which consists of 20 steps. Since there are only sixteen words x_0, ..., x_{15} in the original input message, in each round the four words x_{16}, x_{17}, x_{18}, x_{19} are defined as an XOR of some four words in x_i ($i = 0, \ldots, 15$). Table 1 defines the definition used in each round.
  And the Table 2 summarizes the message scheduling.
- Boolean functions: for each round, a boolean function f is assigned. The boolean function is used in the steps belonging to the round. The boolean functions are as follows:
  - Round 1: $f(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$
  - Round 2: $f(x, y, z) = x \oplus y \oplus z$
  - Round 3: $f(x, y, z) = (x \vee \neg z) \oplus y$
  - Round 4: $f(x, y, z) = x \oplus y \oplus z$
- Constants: similarly, for each round, a constant k is assigned. They are as follows:

Table 1. Message Expansion of HAS-160

	Round 1	Round 2	Round 3	Round 4
x_{16}	$x_0 \oplus x_1 \oplus x_2 \oplus x_3$	$x_3 \oplus x_6 \oplus x_9 \oplus x_{12}$	$x_{12} \oplus x_5 \oplus x_{14} \oplus x_7$	$x_7 \oplus x_2 \oplus x_{13} \oplus x_8$
x_{17}	$x_4 \oplus x_5 \oplus x_6 \oplus x_7$	$x_{15} \oplus x_2 \oplus x_5 \oplus x_8$	$x_0 \oplus x_9 \oplus x_2 \oplus x_{11}$	$x_3 \oplus x_{14} \oplus x_9 \oplus x_4$
x_{18}	$x_8 \oplus x_9 \oplus x_{10} \oplus x_{11}$	$x_{11} \oplus x_{14} \oplus x_1 \oplus x_4$	$x_4 \oplus x_{13} \oplus x_6 \oplus x_{15}$	$x_{15} \oplus x_{10} \oplus x_5 \oplus x_0$
x_{19}	$x_{12} \oplus x_{13} \oplus x_{14} \oplus x_{15}$	$x_7 \oplus x_{10} \oplus x_{13} \oplus x_0$	$x_8 \oplus x_1 \oplus x_{10} \oplus x_3$	$x_{11} \oplus x_6 \oplus x_1 \oplus x_{12}$

Table 2. Message Scheduling of HAS-160

Step	Round 1	Round 2	Round 3	Round 4
0	x_{18}	x_{18}	x_{18}	x_{18}
1	x_0	x_3	x_{12}	x_7
2	x_1	x_6	x_5	x_2
3	x_2	x_9	x_{14}	x_{13}
4	x_3	x_{12}	x_7	x_8
5	x_{19}	x_{19}	x_{19}	x_{19}
6	x_4	x_{15}	x_0	x_3
7	x_5	x_2	x_9	x_{14}
8	x_6	x_5	x_2	x_9
9	x_7	x_8	x_{11}	x_4
10	x_{16}	x_{16}	x_{16}	x_{16}
11	x_8	x_{11}	x_4	x_{15}
12	x_9	x_{14}	x_{13}	x_{10}
13	x_{10}	x_1	x_6	x_5
14	x_{11}	x_4	x_{15}	x_0
15	x_{17}	x_{17}	x_{17}	x_{17}
16	x_{12}	x_7	x_8	x_{11}
17	x_{13}	x_{10}	x_1	x_6
18	x_{14}	x_{13}	x_{10}	x_1
19	x_{15}	x_0	x_3	x_{12}

- Round 1: `0x00000000`
- Round 2: `0x5a827999`
- Round 3: `0x6ed9eba1`
- Round 4: `0x8f1bbcdc`

– Rotation: each step involves two bit rotations. The amount of the rotations, s_1 and s_2 are dependent on the step. These are as follows: first, s_1 is given as follows.

step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
s_1	5	11	7	15	6	13	8	14	7	12	9	11	8	15	6	12	9	14	5	13

The sequence for s_1 is the same for other rounds. Next, s_2 is dependent on the round:

- Round 1: $s_2 = 10$
- Round 2: $s_2 = 17$
- Round 3: $s_2 = 25$
- Round 4: $s_2 = 30$

– Transformation: at each step, from the given values of the variables a_i, b_i, ..., e_i, the updated values a_{i+1}, b_{i+1}, ..., e_{i+1} are calculated as follows:

- $a_{i+1} \leftarrow (a_i \lll s_1) + f(b_i, c_i, d_i) + e_i + x + k$
- $b_{i+1} \leftarrow a_i$

- $c_{i+1} \leftarrow (b_i \lll s_2)$
- $d_{i+1} \leftarrow c_i$
- $e_{i+1} \leftarrow d_i$

where f, x, k are the boolean function, the message, and the constant for the step as described before, and s_1, s_2 are amounts of left bit rotations. $+$ is the addition modulo 2^{32}.

3 Collision Pair for 45-Step HAS-160

3.1 Notation

In this article we follow notations of Wang et al.[1] for keeping track of the modular differences.

Definition 1. *For any v representing a bit sequence of length 32, and any index j (j=1, ..., 32), $v[j]$ (or sometimes $v[+j]$) denotes the value obtained by changing the j-th bit of v from 0 to 1. This notation implicitly states that j-th position of v is 0.*

Similarly $v[-j]$ denotes the value obtained by changing the j-th bit of v from 1 to 0. Also this implicitly means that the j-th position of v is 1.

Finally, for l distinct indices j_1, ..., j_l, the notation $v[\pm j_1, \pm j_2, \ldots, \pm j_l]$ is a shorthand for

$$v[\pm j_1][\pm j_2]\cdots[\pm j_l] = (\cdots((v[\pm j_1])[\pm j_2])\cdots)[\pm j_l].$$

We are going to find a collision pair of messages $X = (x_i)_{i=0}^{15}$ and $X' = (x'_i)_{i=0}^{15}$. Then a_i, b_i, c_i, d_i, e_i will denote the values of a, b, c, d, e at the beginning of the step i when the message is X, and a'_i, b'_i, c'_i, d'_i, e'_i will denote the values of a, b, c, d, e corresponding to the message X'. At each step, the difference of the variables a, ..., e will be represented by writing a', ..., e' in terms of a, ..., e using the notation of the Definition 1.

3.2 The Differential Path

The Table 4 at the Appendix A represents the differential path we have used in order to find the 45-step collision of HAS-160.

The message differences are at x_3 and x_9:

$$x'_3 - x_3 = 2^{31}, x'_9 - x_9 = 2^{31}, \quad \text{and} \quad x'_i = x_i \quad \text{for } i \neq 3, 9.$$

The differences of the variables a_i, b_i, c_i, d_i, e_i vanishes at the step 24 and it continues to do so up until the step 46, by the influence of the message word difference at the step 45.

We have chosen the message differences due to the following reasons; The boolean function $x \oplus y \oplus z$, for rounds 2 and 4, is not very nice, in the sense that if we have a difference at one of the input variables, then it is not possible to

cancel the difference at the output, so the difference propagates. And the same is true for the boolean function $(x \vee \neg z) \oplus y$ for the input y. Due to the step-dependent bit rotations and the message scheduling which is basically just word re-ordering, it is difficult to find message word differences which give the full collision with high probability. Therefore, by giving differences only to message words x_3 and x_9 at the same bit position 32, we can produce an inner collision between steps 24 and 45, essentially avoiding most of the round 2, where the boolean function is $x \oplus y \oplus z$.

The next table in the Appendix A, Table 5, shows the sufficient conditions for the differential path in the Table 4. The differential table was found essentially by trial-and-error search, the guideline of the search being minimizing the Hamming weight of the modular difference of the words. Since actual bit position where the difference occurs can be modified without affecting the modular difference, we can modify the path by expanding the differing bit positions of variables appropriately. For example, at the step 13, instead of using $a_{13}[-12, -26]$ to represent a modular difference of $-2^{11} - 2^{25}$, we have used $a_{13}[12, 13, -14, 26, 27, -28]$ so that the differential path above the step 13 becomes simpler.

3.3 The Collision Search Algorithm

Since in the Table 5 there are 233 equations in total, we cannot efficiently search the correct collision pair simply by choosing messages randomly. Therefore we use the message modification technique introduced in [1].

For example, suppose that all the equations for a_i are already satisfied for some step i. if x is the word involved in the transformation at step i, we many modify the word x in order to 'correct' the next variable a_{i+1} as follows: since we have the defining relation

$$a_{i+1} = (a_i \lll s_1) + f(b_i, c_i, d_i) + e_i + x + k,$$

instead of a_{i+1}, we substitute the 'correct' value α which satisfies all the required equations for a_{i+1}, then we redefine the message word x as $\alpha - (a_i \lll s_1) - f(b_i, c_i, d_i) - e_i - k$.

Contrary to MD5 and SHA-1, HAS-160 doesn't use the original message words directly in the first round; It is interspersed with 'synthesized' words $x_{16}, \ldots, x_{19}$ so that the words in the first round are no longer independent. Therefore, we need to be a little more careful in applying the message modification technique.

One simple observation is that we can reverse the roles of the defined words and the independently chosen words: for example, in the round 1 the relation $x_{18} = x_8 \oplus x_9 \oplus x_{10} \oplus x_{11}$ holds, and this can be rewritten as $x_{11} = x_{18} \oplus x_8 \oplus x_9 \oplus x_{10}$. Now we may regard x_{18}, x_8, x_9, and x_{10} are the independently given input and x_{11} is defined in terms of the other four.

Therefore, in the first round we may consider the 16 words x_{18}, x_0, x_1, x_2, x_3, x_{19}, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{12}, x_{13}, x_{14} are independent words and $x_{16} = x_0 \oplus x_1 \oplus x_2 \oplus x_3$, $x_{11} = x_{18} \oplus x_8 \oplus x_9 \oplus x_{10}$, $x_{17} = x_4 \oplus x_5 \oplus x_6 \oplus x_7$, and $x_{15} = x_{19} \oplus x_{12} \oplus x_{13} \oplus x_{14}$ defined in terms of other words, i.e., among the

linearly dependent five words, the word which appears at the last is considered as dependent.

Then the search algorithm works as follows:

1. Choose the message words $(x_i)_{i=0}^{15}$ randomly.
2. In the round 1 compute $x_{16}, \ldots, x_{19}$ following the definition for them.
3. Prepare the other message words $(x'_i)_{i=0}^{15}$ so that the two messages have the prescribed difference: $x'_3 - x_3 = 2^{31}$, $x'_9 - x_9 = 2^{31}$.
4. In the round 1, at each step if the involved message word is considered independent, then rewrite the message to correct the next a_i. Since the first (now redefined) dependent word, x_{16}, occurs at step 10, all the equations occuring at step 0 to step 10 may be forced to be satisfied by the message modification technique.
5. Since x_{16} is a dependent word, we cannot apply the message modification technique to this word. Since at the step 11, there are 14 equations in total for the variable a_{11} to be satisfied, heuristically the probability that all the equations for a_{11} are satisfied is 2^{-14}. If any of the equations for a_{11} is not satisfied, go to 1 again to choose random message words. Proceed to next if all the equations for a_{11} are satisfied.
6. Correct independent words x_8, x_9, and x_{10} to satisfy all the equations for steps 12, 13, and 14.
7. Now the next two words, x_{11} and x_{17} are dependent words. There are 8 equations for a_{15} and 4 equations for a_{16}. Therefore the probability that all the 12 equations are satisfied is 2^{-12}. If any of the equations is not satisfied, then randomly choose x_8, x_9, and x_{10} again, go to 6, define dependent words, and continue the message modification process for the newly chosen words, until all the 12 equations are satisfied. Proceed to next.
8. Now all the equations for steps up to 16 are satisfied. After this all the remaining words, except the last word x_{15}, are independent. For these we can use the message modification technique to correct all the remaining equations. This will produce a inner collision between step 24 and 45.

Therefore, the complexity of the above algorithm is dominated by the number of expected trial-and-errors involved. In terms of the number of 45-step hash operations, the complexity in finding messages for step 0 to 10 is bounded by

$$2^{14} \cdot \frac{11}{45} \leq 2^{12},$$

and the complexity for step 11 to 15 is bounded by

$$2^{12} \cdot \frac{5}{45} \leq 2^{9}.$$

So the total complixity can be estimated to be 2^{12}.

3.4 The Actual Collision Found

Table 3 is a collision for 45-step HAS-160 we have found. For both messages X and X', the 45-th output (that is, the output of the step 44) is equal to `b4b01f8e`. It took less than a second on a PC with a 2.66 GHz Pentium 4 CPU.

Table 3. A collision pair for 45-step HAS-160

X	value	X'	value
x_0	cf31e167	x'_0	cf31e167
x_1	0568c383	x'_1	0568c383
x_2	5b04edde	x'_2	5b04edde
x_3	9583016e	x'_3	1583016e
x_4	2e351a4b	x'_4	2e351a4b
x_5	74049b4f	x'_5	74049b4f
x_6	41839df9	x'_6	41839df9
x_7	68e50c1a	x'_7	68e50c1a
x_8	745148dc	x'_8	745148dc
x_9	611f2697	x'_9	e11f2697
x_{10}	0ee6ca41	x'_{10}	0ee6ca41
x_{11}	b9b3f627	x'_{11}	b9b3f627
x_{12}	237aec88	x'_{12}	237aec88
x_{13}	bc1f3ac9	x'_{13}	bc1f3ac9
x_{14}	397e6fd8	x'_{14}	397e6fd8
x_{15}	506a788c	x'_{15}	506a788c

4 Conclusion

In this paper, we have described an 2^{12} algorithm to find collisions of 45-step HAS-160, and actually presented a collision pair. It seems difficult to extend this method directly to the full HAS-160, or even beyond the round 3. Further study is needed to refine the attack to evaluate the strength of the algorithm more fully.

Acknowledgement. Many thanks to the anonymous reviewers for careful reading and valuable comments.

References

1. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, Xiuyuan Yu, *Cryptanalysis of the Hash Functions MD4 and RIPEMD*, In Lecture Notes in Computer Science, Volume 3494, May 2005, Pages 1–18.
2. Xiaoyun Wang, Hongbo Yu, *How to Break MD5 and Other Hash Functions*, Lecture Notes in Computer Science, Volume 3494, May 2005, Pages 19–35.
3. Telecommunications Technology Association, *Hash Function Standard Part 2: Hash Function Algorithm Standard (HAS-160)*, TTAS.KO-12.0011/R1, Dec. 2000.
4. FIPS 180-2, *Secure Hash Standard*, `http://csrc.nist.gov/publications/`, 2002.
5. Xiaoyun Wang, Hongbo Yu, Yiqun Lisa Yin, *Efficient Collision Search Attacks on SHA-0*, Crypto '05
6. Xiaoyun Wang, Yiqun Lisa Yin, Hongbo Yu, *Finding Collisions in the Full SHA-1*, Crypto '05.

A The Differential Path and the Equations

In this section we present the differential path we used to find the collision and the sufficient conditions for the path.

The Table 4 represents the differential path. The column x displays the message words used at each step, and the column Δx means the index of the bit where the difference is introduced. Note that we have introduced the differences only at the bit position 32.

In the table, $\cdot$ means that the difference is zero. The empty boxes in steps 4–20 are intentionally left blank because they contain values that can be mechanically determined based on the column a'; for example the blanks on the column b' are simply translates of the corresponding spaces on the column a', and the blanks on the column c' can be determined by simple rotation from the corresponding values on the column b'.

The Table 5 represents the sufficient conditions for the path. Similarly, we used the simple relations between the variables to standardize most equations in Table 5 so that they are written in terms of the variables a_i.

Table 4. The differential path used in the paper

Step	x	Δx	s_1	a'	b'	c'	d'	e'
0	x_{18}	32	5	·	·	·	·	·
1	x_0		11	$a_1[-32]$	·	·	·	·
2	x_1		7	$a_2[-11]$	$b_2[-32]$	·	·	·
3	x_2		15	$a_3[18,-19]$	$b_3[-11]$	$c_3[-10]$	·	·
4	x_3	32	6	$a_4[1,\ldots,12,-13]$		$c_4[-21]$	$d_4[-10]$	·
5	x_{19}		13	$a_5[-7,-32]$			$d_5[-21]$	$e_5[-10]$
6	x_4		8	$a_6[-20]$				$e_6[-21]$
7	x_5		14	$a_7[14,22]$				
8	x_6		7	$a_8[4,10,15,\ldots,21,-22]$				
9	x_7		12	·				
10	x_{16}	32	9	$a_{10}[-4,-16,\ldots,-21,22]$				
11	x_8		11	$a_{11}[-13,14,20]$				
12	x_9	32	8	$a_{12}[4,\ldots,8,-9,-17]$				
13	x_{10}		15	$a_{13}[12,13,-14,26,27,-28]$				
14	x_{11}		6	$a_{14}[17]$				
15	x_{17}		12	$a_{15}[-18]$				
16	x_{12}		9	$a_{16}[5]$				
17	x_{13}		14	$a_{17}[22]$				
18	x_{14}		5	$a_{18}[-22]$	$b_{18}[22]$			
19	x_{15}		13	$a_{19}[15]$	$b_{19}[-22]$	$c_{19}[32]$		
20			5	·	$b_{20}[15]$	$c_{20}[-32]$	$d_{20}[32]$	
21		32	11	·	·	$c_{21}[32]$	$d_{21}[-32]$	$e_{21}[32]$
22			7	·	·	·	$d_{22}[32]$	$e_{22}[-32]$
23		32	15	·	·	·	·	$e_{23}[32]$
24			6	·	·	·	·	·
25			13	·	·	·	·	·
26			8	·	·	·	·	·
27			14	·	·	·	·	·
28			7	·	·	·	·	·
29			12	·	·	·	·	·
30			9	·	·	·	·	·
31			11	·	·	·	·	·
32			8	·	·	·	·	·
33			15	·	·	·	·	·
34			6	·	·	·	·	·
35			12	·	·	·	·	·
36			9	·	·	·	·	·
37			14	·	·	·	·	·
38			5	·	·	·	·	·
39			13	·	·	·	·	·
40			5	·	·	·	·	·
41			11	·	·	·	·	·
42			7	·	·	·	·	·
43			15	·	·	·	·	·
44			6	·	·	·	·	·
45		32	13	·	·	·	·	·
46			8	$a_{46}[\pm 32]$	·	·	·	·

Table 5. The equations for the differential path

	sufficient conditions
a_0	$a_{0,22} = b_{0,22}$
a_1	$a_{1,1} = a_{0,1}$, $a_{1,32} = 1$
a_2	$a_{2,3} = 1$, $a_{2,8} = a_{1,8}$, $a_{2,9} = a_{1,9}$, $a_{2,10} = 0$, $a_{2,11} = 1$, $a_{2,29} = 0$, $a_{2,32} = 0$
a_3	$a_{3,1} = a_{2,1}$, $a_{3,2} = a_{2,2}$, $a_{3,3} = 0$, $a_{3,10} = 1$, $a_{3,18} = 0$, $a_{3,19} = 1$, $a_{3,21} = 0$, $a_{3,23} = a_{2,23}$, $\ldots$, $a_{3,31} = a_{2,31}$, $a_{3,32} = 1$
a_4	$a_{4,1} = 0$, $\ldots$, $a_{4,12} = 0$, $a_{4,13} = 1$, $a_{4,21} = 1$, $a_{4,22} = a_{3,22}$, $a_{4,28} = 0$, $a_{4,29} = 0$
a_5	$a_{5,7} = 1$, $a_{5,10} = 1$, $a_{5,11} = 0$, $a_{5,12} = 0$, $a_{5,13} = 0$, $a_{5,14} = 1$, $a_{5,15} = 0$, $\ldots$, $a_{5,20} = 0$, $a_{5,21} = 1$, $a_{5,22} = 1$, $a_{5,23} = 0$, $a_{5,28} = 0$, $a_{5,29} = 1$, $a_{5,32} = 1$
a_6	$a_{6,4} = a_{5,4}$, $a_{6,6} = 0$, $a_{6,7} = 0$, $a_{6,10} = 0$, $a_{6,11} = 0$, $a_{6,12} = 1$, $a_{6,13} = 1$, $a_{6,14} = 1$, $a_{6,15} = 0$, $a_{6,16} = 0$, $a_{6,17} = 1$, $\ldots$, $a_{6,23} = 1$, $a_{6,26} = 1$, $a_{6,32} = 0$
a_7	$a_{7,5} = a_{6,5}$, $a_{7,6} = 1$, $a_{7,7} = 1$, $a_{7,8} = a_{6,8}$, $a_{7,9} = a_{6,9}$, $a_{7,10} = 1$, $a_{7,11} = a_{6,11}$, $a_{7,12} = a_{6,12}$, $a_{7,14} = 0$, $a_{7,17} = 0$, $a_{7,22} = 0$, $a_{7,26} = 0$, $a_{7,30} = 0$, $a_{7,32} = 1$
a_8	$a_{8,4} = 0$, $a_{8,7} = 0$, $a_{8,10} = 0$, $a_{8,15} = 0$, $\ldots$, $a_{8,21} = 0$, $a_{8,22} = 1$, $a_{8,24} = 0$, $a_{8,26} = 0$, $a_{8,30} = 1$, $a_{8,32} = 0$
a_9	$a_{9,4} = 1$, $a_{9,6} = a_{8,6}$, $a_{9,7} = 1$, $a_{9,8} = a_{8,8}$, $a_{9,9} = a_{8,9}$, $a_{9,10} = 1$, $a_{9,11} = a_{8,11}$, $a_{9,12} = a_{8,12}$, $a_{9,14} = 1$, $a_{9,20} = 1$, $a_{9,24} = 1$, $\ldots$, $a_{9,29} = 1$, $a_{9,30} = 0$, $a_{9,31} = 0$
a_{10}	$a_{10,3} = a_{9,3}$, $a_{10,4} = 1$, $a_{10,7} = 1$, $a_{10,10} = 0$, $a_{10,14} = 1$, $a_{10,16} = 1$, $\ldots$, $a_{10,21} = 1$, $a_{10,22} = 0$, $a_{10,25} = 0$, $\ldots$, $a_{10,30} = 0$, $a_{10,31} = 1$, $a_{10,32} = 1$
a_{11}	$a_{11,4} = 1$, $a_{11,7} = 0$, $a_{11,13} = 1$, $a_{11,14} = 0$, $a_{11,16} = 1$, $a_{11,17} = 0$, $a_{11,20} = 0$, $a_{11,26} = 0$, $\ldots$, $a_{11,31} = 0$, $a_{11,32} = 1$
a_{12}	$a_{12,2} = a_{11,2}$, $a_{12,3} = a_{11,3}$, $a_{12,4} = 0$, $\ldots$, $a_{12,8} = 0$, $a_{12,9} = 1$, $a_{12,14} = 1$, $a_{12,16} = 0$, $a_{12,17} = 1$, $a_{12,18} = a_{11,18}$, $a_{12,23} = 0$, $a_{12,24} = 0$, $a_{12,26} = 1$, $a_{12,27} = 0$, $a_{12,28} = 0$, $a_{12,29} = 0$, $a_{12,30} = 1$, $a_{12,31} = 1$, $a_{12,32} = 1$
a_{13}	$a_{13,7} = 0$, $a_{13,12} = 0$, $a_{13,13} = 0$, $a_{13,14} = 1$, $a_{13,15} = 0$, $a_{13,16} = 0$, $a_{13,17} = 0$, $a_{13,18} = 1$, $a_{13,19} = 1$, $a_{13,23} = 0$, $a_{13,24} = 1$, $a_{13,26} = 0$, $a_{13,27} = 0$, $a_{13,28} = 1$, $a_{13,30} = 1$
a_{14}	$a_{14,4} = 0$, $a_{14,5} = 1$, $a_{14,6} = 0$, $a_{14,8} = a_{13,8}$, $a_{14,14} = 1$, $a_{14,15} = 1$, $a_{14,16} = 1$, $a_{14,17} = 0$, $a_{14,18} = 1$, $a_{14,19} = 1$, $a_{14,22} = 0$, $a_{14,23} = 1$, $a_{14,24} = 1$, $a_{14,27} = 1$
a_{15}	$a_{15,4} = 1$, $a_{15,5} = 1$, $a_{15,6} = 1$, $a_{15,18} = 1$, $a_{15,22} = 0$, $a_{15,23} = 1$, $a_{15,24} = 1$, $a_{15,27} = 1$
a_{16}	$a_{16,5} = 0$, $a_{16,12} = a_{15,12}$, $a_{16,27} = 1$, $a_{16,28} = 0$
a_{17}	$a_{17,12} = a_{16,12}$, $a_{17,15} = 1$, $a_{17,22} = 0$, $a_{17,28} = 1$
a_{18}	$a_{18,5} = a_{17,5} \oplus 1$, $a_{18,15} = 1$, $a_{18,22} = 1$, $a_{18,32} = 0$
a_{19}	$a_{19,15} = 0$

The Program Counter Security Model: Automatic Detection and Removal of Control-Flow Side Channel Attacks

David Molnar[1], Matt Piotrowski[1], David Schultz[2], and David Wagner[1]

[1] UC-Berkeley
{dmolnar, pio, daw}@eecs.berkeley.edu
[2] MIT
das@csail.mit.edu

Abstract. We introduce new methods for detecting control-flow side channel attacks, transforming C source code to eliminate such attacks, and checking that the transformed code is free of control-flow side channels. We model control-flow side channels with a *program counter transcript*, in which the value of the program counter at each step is leaked to an adversary. The program counter transcript model captures a class of side channel attacks that includes timing attacks and error disclosure attacks.

Further, we propose a generic source-to-source transformation that produces programs provably secure against control-flow side channel attacks. We implemented this transform for C together with a static checker that conservatively checks x86 assembly for violations of program counter security; our checker allows us to compile with optimizations while retaining assurance the resulting code is secure. We then measured our technique's effect on the performance of binary modular exponentiation and real-world implementations in C of RC5 and IDEA: we found it has a performance overhead of at most 5× and a stack space overhead of at most 2×. Our approach to side channel security is practical, generally applicable, and provably secure against an interesting class of side channel attacks.

1 Introduction

The last decade has seen a growing realization that side channel attacks pose a significant threat to the security of both embedded and networked cryptographic systems. The issue of information leakage via covert channels was first described by Lampson [19] in the context of timesharing systems, but the implications for cryptosystem implementations were not recognized at the time. In his seminal paper, Kocher showed that careful timing measurements of RSA operations could be used to discover the RSA private key through "timing analysis" [17]. Kocher, Jaffe, and Jun showed how careful power measurements could reveal private keys through "power analysis" [18]. Since then, side channel attacks have been used to break numerous smart card implementations of both symmetric and public-key

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 156–168, 2006.
© Springer-Verlag Berlin Heidelberg 2006

cryptography [10, 22, 21, 23]. Later, Boneh and Brumley showed that a timing attack could be mounted against a remote web server [9]. Other recent attacks on SSL/TLS web servers make use of bad version oracles or padding check oracles; remote timing attacks can reveal error conditions enabling such attacks even if no explicit error messages are returned [20, 6, 32, 7, 15, 16, 25].

Defending against side channels requires a combination of software and hardware techniques. We believe a principled solution should extend the hardware/software interface by disclosing the side-channel properties of the hardware. Just as an instruction set architecture specifies the behavior of the hardware to sufficient fidelity that a compiler can rely on this specification, for side-channel security we advocate that this architecture should specify precisely what information the hardware might leak when executing by each instruction. This boundary forms a "contract" between hardware and software countermeaures as to who will protect what; the role of the hardware is to ensure that what is leaked is nothing more than what is permitted by the instruction set specification, and the role of the software countermeasures is to ensure that the software can tolerate leakage of this information without loss of security.

Our main technical contribution is an exploration of one simple but useful contract, the *program counter transcript model*, where the only information the hardware leaks is the value of the program counter at each step of the computation. The intuition behind our model is that it captures an adversary that can see the entire control flow behavior of the program, so it captures a non-trivial class of side-channel attacks. This paper develops methods for detecting such attacks on C programs and shows how to secure software against these attacks using a C-to-C code transformation.

We introduce a source-to-source program transformation that takes a program P and produces a transformed program P' with the same input-output behavior and with a guarantee that P' will be program counter secure. We built a prototype implementation of this transformation that works on C source code. We applied our implementation to implementations of RC5 and IDEA written in C, as well as an implementation of binary modular exponentiation. The resulting code is within a factor of at most 5 in performance and a factor of 2 in stack usage of untransformed code (§ 5.1).

Because our transform works at the C source level, we must be careful that the compiler does not break our security property. We build a static analysis tool that conservatively checks x86 assembly code for violations of program counter security. For example, we were able to show that our transformed code, when compiled with the Intel optimizing C compiler, retains the security properties.

The program counter model does not cover all side channel attacks. In particular, data dependent side channels (such as DPA or cache timing attacks [5]) are *not* eliminated by our transform. Nevertheless, we still believe the model is of value. We do not expect software countermeasures alone to solve the problem of side channels.

In short, we show how to discover and defend against a class of attacks, while leaving defenses against some important attacks as an open question. Our work

opens the way to exploring a wide range of options for the the interface between hardware and software side channel countermeasures, as formalized by different transcript models. As such, our work is a first step towards efficient, principled countermeasures against side channel attacks.

2 A Transcript Model for Side Channel Attacks

We formalize the notion of side information by a *transcript*. We view program execution as a sequence of n steps. A transcript is then a sequence $T = (T_1, \ldots, T_n)$, where T_i represents the adversary's observation of the side channel at the i^{th} step of the computation. We will then write $P_k(x)$ to mean the program P running on secret input k and non-secret input x. Informally, a program is secure if the adversary "learns nothing" about k even given access to the side-channel transcript produced during execution of $P_k(x)$ for x values of its choice. Our model can be thought of as a "single-program" case of the Micali-Reyzin model, in which their "leakage function" corresponds to our notion of a transcript [24].

The transcript is the way we formalize the contract between hardware and software. It is the job of hardware designers to ensure that the hardware leaks nothing more than what is specified in the transcript, and the rest of this paper assumes that this has been done.

We write $D \sim D'$ if D and D' have the same distribution (perfect indistinguishability). Programs will take two inputs, a key k and an input x; we write $P_k(x)$ for the result of evaluating P on key k and input x. Define $\#P_k(x)\# \stackrel{\text{def}}{=} (y, T)$, where $y = P_k(x)$ is the result of executing P on (k, x) and T denotes the transcript produced during that execution. If P is randomized, $P_k(x)$ and $\#P_k(x)\#$ are random variables. We abuse notation and write $\#P_k\#$ for the map $\#P_k\#(x) = \#P_k(x)\#$. We can then define *transcript security* as follows:

Definition 1 *(transcript security). A program P is said to be transcript-secure (for a particular choice of transcript) if for all probabilistic polynomial time adversaries A, there exists a probabilistic polynomial time simulator S, which for all keys k satisfies $S^{P_k} \sim A^{\#P_k\#}$.*

3 Program Counter Security: Security Against Certain Side-Channel Attacks

In the PC model, the transcript T conveys the sequence of values taken on by the processor's program counter during the computation. To be specific, our concrete notion of security is as follows:

Definition 2 *(PC-security). A program P is* PC-secure *if P is transcript-secure when the transcript $T = (T_1, \ldots, T_n)$ is defined so that T_i represents the value of the program counter at the i^{th} step of the execution.*

Consequently, in the PC model, the attacker learns everything about the program's control-flow behavior but nothing about other intermediate values computed during execution of the program. In the remainder of this work, we make two assumptions about the hardware used to execute the program: first, that the the program text is known to the attacker. This implies that the program counter at time i reveals the opcode that was executed at time i. Second, the side-channel signal observed by the attacker depends only on the sequence of program counter values, e.g., on the opcode executed. For example, we assume that the execution time of each machine instruction can be predicted without knowledge of the values of its operands, so that its timing behavior is not data-dependent in any way. Warning: This is not true on some architectures, due to, among other things, cache effects [5], data-dependent instruction timing, and speculation.

We stress that our transcript model is intended as an idealization of the information leaked by the hardware; it may be that no existing system meets the transcript precisely. Nonetheless, we believe these assumptions are reasonable for some embedded devices, namely those which do not have caches or sophisticated arithmetic units. With these assumptions, PC-security subsumes the attacks mentioned above. Given a transcript of PC values, the attacker can infer the total number of machine cycles used during the execution of the program, and thus the total time taken to run this program; consequently, any program that is PC-secure will also be secure against timing attacks.

4 Example: Error Disclosure Side Channels

Some implementation attacks exploit information leaks from the disclosure of decryption failures. Consider a decryption routine that can return several different types of error messages, depending upon which stage of decryption failed (e.g., improper PKCS formatting, invalid padding, MAC verification failure). It turns out that, in many cases, revealing which kind of failure occurred leaks information about the key [6, 32, 7, 15, 16, 25].

Naïvely, one might expect that attacks can be avoided if the implementation always returns the same error message, no matter what kind of decryption failure occurred. Unfortunately, this simple countermeasure does not go far enough. Surprisingly, in many cases timing attacks can be used to learn which kind of failure occurred, even if the implementation does not disclose this information explicitly [6, 32, 7, 15, 16, 25, 20]. See, for instance, Fig. 1(a). The existence of such attacks can be viewed as a consequence of the lack of PC-security. Thus, a better way to defend against error disclosure attacks is to ensure that all failures result in the same error message and that the implementation is PC-secure. We show several similar applications of PC-security in the full paper [26].

Suppose we had a subroutine $\textsc{Cond}(e, t, f)$ that returns t or f according to whether e is true or false. Using this subroutine, we propose in Fig. 1(b) one possible implementation strategy for securing the code in Fig. 1(a) against error disclosure attacks. If there is an error in Line 1, we generate a dummy value

OAEP-Insecure(d, x):
1. $(e, y) \leftarrow$ IntToOctet$(x^d \bmod n)$
2. **if** e **then return** Error
3. $(e', z) \leftarrow$ OAEPDecode(y)
4. **if** e' **then return** Error
5. **return** z

(a) Naïve code (insecure)

OAEP-Secure(d, x):
1. $(e, y) \leftarrow$ IntToOctet$(x^d \bmod n)$
2. $y \leftarrow$ Cond$(e,$ dummy value$, y)$
3. $(e', z) \leftarrow$ OAEPDecode(y)
4. **return** Cond$(e \vee e',$ Error$, z)$

(b) A transformed version (PC-secure)

Fig. 1. Two implementations of OAEP decryption. We assume that each subroutine returns a pair (e, y), where e is a boolean flag indicating whether any error occurred, and y is the value returned by the subroutine if no error occurred. The code on the left is insecure against Manger's attack, because timing analysis allows to distinguish an error on Line 2 from an error on Line 3. The code in the right is PC-secure and hence not vulnerable to timing attacks, assuming that the subroutines are themselves implemented in a PC-secure form.

(which can be selected arbitrarily from the domain of OAEPDecode) to replace the output of Line 1. If all subroutines are implemented in a PC-secure way and we can generate a dummy value in a PC-secure way, then our transformed code will be PC-secure and thus secure against error disclosure attacks.

There is one challenge: we need a PC-secure implementation of Cond. We propose one way to meet this requirement through logical masking:

Cond(e, t, f):
1. $m \leftarrow$ Mask(e)
2. **return** $(m \wedge t) \vee (\neg m \wedge f)$

Here $\neg, \vee, \wedge$ represent the bitwise logical negation, OR, AND (respectively). This approach requires a PC-secure subroutine Mask satisfying Mask(false) $= 0$ and Mask(true) $= 2^\ell - 1 = 11 \cdots 1_2$, assuming t and f are ℓ-bit values. The Mask subroutine can be implemented in many ways. For instance, assuming true and false are encoded as values 1 and 0, we could use Mask$(e) = (2^\ell - 1) \times e$; Mask$(e) = -e$ (on two's-complement machines); Mask$(e) = (e \ll (\ell - 1)) \ggg (\ell - 1)$ (using a sign-extending arithmetic right shift); or several other methods. With the natural translation to machine code, these instantiations of Mask and Cond will be PC-secure.

4.1 Straight-Line Code is PC-Secure

The key property we have used to show PC-security of code in the previous section is that the code is *straight-line*, by which we mean that the flow of control through the code does not depend on the data in any way. We now encapsulate this in a theorem.

Theorem 1. *(PC-security of straight-line code). Suppose the program P has no branches, i.e., it has no instructions that modify the PC. Then P is PC-secure.*

Proof. Since P is branch-free, for all inputs x and all keys k the program counter transcript T of $P_k(x)$ will be the same. For any adversary A, consider the simulator S that runs A, outputting whatever A does and answering each query x that A makes to its oracle with the value $(P_k(x), T)$. Then $S^{P_k} \sim A^{\#P_k\#}$ for all k.

In fact, it suffices for P to be free of key-dependent branches. We can use this to show that some looping programs are PC-secure.

Theorem 2. *(PC-security of some looping programs). Suppose the program P consists of straight-line code and loops that always run the same code body for a fixed constant number of iterations in the same order (i.e., the loop body contains no break statements and no assignments to the loop counter; there are no* `if` *statements). Then P is PC-secure.*

Proof. As before, for all inputs x and all keys k, the program counter transcript T of $P_k(x)$ will be the same, so we can use the same simulation strategy.

5 Code Transformation for PC-Security

The examples we have seen so far illustrate the relevance of PC-security, but enforcing PC-security by hand is highly application-dependent and labor-intensive. We describe a generic method for transforming code to be PC-secure. Given a program P, the transformed program P' is PC-secure and has the same input-output behavior as P on all inputs (i.e., $P_k(x) \sim P'_k(x)$ for all k, x). It may be surprising that almost all code can be transformed in this way, but we show in the full paper that this can be done for any fragment of code where all loops executed for a bounded number of iterations [26].

Transforming conditional statements. An `if` statement containing only assignment expressions can be handled in a fairly simple way. To provide PC-security, we execute both branches of the `if`, but only retain the results from the branch that would have been executed in the original program. The mechanism we use to nullify the side-effects of either the consequent or the alternative is *conditional assignment*. We have already seen one way to implement PC-secure conditional assignment using logical masking and the COND subroutine. For example, the C statement `if (p) { a = b; }` can be transformed to `a = COND(m, b, a)`, where `m = MASK(p)`. If `p` represents a 0-or-1-valued boolean variable[1], this might expand to the C code `m = -p; a = (m & b) | (~m & a)`.

Loops. Loops present difficulties because the number of iterations they will perform may not be known statically, and in particular, the number of iterations may depend on sensitive data. Fortunately, in many cases a constant upper bound can be established on the number of iterations. We transform the loop so that it always executes for the full number of iterations, but the results of any iterations that would not have occurred in the original program are discarded. A specification of our entire transform may be found in Appendix A

[1] If `p` is not guaranteed to be 0-or-1-valued, this definition of `m` does not work. We use `m = !p - 1` instead.

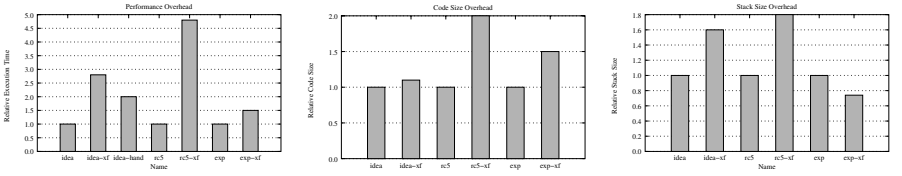


Fig. 2. The speed, code size, and stack size overhead of our transform, as applied to modular exponentiation, IDEA, and RC5. The -xf suffix indicates the automatic application of our transform, while the -hand suffix indicates a hand-optimized application of our transform. Values are normalized: the untransformed version of a program takes unit time by definition, while the transformed version of the same program is shown with the relative multiplicative overhead compared to the untransformed version.

5.1 Transform Implementation

We applied our transform to implementations of the IDEA and RC5 block ciphers, which are known to be susceptible to timing attacks unless implemented carefully [14, 12]. We also applied our transform to a simple binary modular exponentiation implementation built on top of the GNU Multiprecision Library. We chose x86 as our reference platform due to its widespread popularity, and we used the Intel C Compiler, Version 8.1 for our performance results. Our tests were run on a 2.8 GHz Pentium 4 running FreeBSD 6-CURRENT (April 17, 2005).

We first optimized our transform by hand on IDEA's multiplication routine to determine how fast our transform can be in principle. Our hand-optimized transformation achieves a factor of 2× slowdown compared to untransformed code, when both are compiled using the Intel C compiler with `-O3`.

We then implemented an automatic C source-to-source transformation using the C Intermediate Language package [27]. Our implementation was intended as an early prototype to demonstrate the feasibility of applying our transformation automatically. With more careful attention, better performance from an automatic transform may be possible.

Performance results. Our performance results for modular exponentiation, IDEA, and RC5 are presented in Fig. 2. For IDEA, we transformed only the `mul` routine, which we identified as the main candidate for timing and power attacks. For RC5, we performed the transformation on the rotate routine, for similar reasons. For modexp, we transformed only the main loop, but did not transform GnuMP library functions.

The performance of untransformed, optimized code is set to 1, and the performance of transformed code is reported relative to this value; for example, the bar with height "2" for `idea-hand` indicates that our hand-transformed IDEA code took 2 times as long as untransformed code. We also found that code size increased by at most a factor of 2. Finally, we considered the stack usage of transformed code; this is the most relevant metric of memory usage for systems without dynamically allocated memory.

We can see that both our automatic transform is within a factor of 5 in performance of the optimized untransformed code in all cases. Again, our implementation is a prototype intended to test feasibility; with more work, more efficient code may be possible. Further, our stack size and code size increase by at most a factor of 2, showing that the transformed code may still be reasonable even in constrained environments. Our results suggest that a fully automatic transformation with slowdown acceptable for many applications is possible with care.

A static analysis for PC-security. We cannot guarantee that the compiler will preserve our transform's straight-line + restricted-loop guarantee when it generates assembly language. We addressed this problem by building a simple static checker for x86 assembly code that detects violations of PC-security. If the compiler does introduce assembly constructs that violate PC-security, these constructs will be flagged by the checker. We can then revise the code or improve our transform. Our checker is sound, but not complete: it will catch all violations of PC-security, but may also report false positives.

In fact, our checker caught unsafe constructs in the gcc 3.3.2 compilation of our transformed C code to x86 assembly. In certain expression contexts, gcc compiles the logical negation (`!`) operator into an assembly sequence involving a conditional branch. Further experimentation reveals that this idiom is not limited to the x86; the Sun C compiler on an UltraSPARC-60 machine exhibits similar behavior. We discovered, however, that the Intel C compiler does not compile `!` using conditional jumps, so we used the Intel compiler for our performance experiments. One alternative would be to change the transform to avoid the `!` operator, but we did not find a portable and efficient replacement. Another alternative would be to modify gcc to add an extra mode that respects the PC-security of compiled code; we found it easier, however, to simply use the Intel compiler for our tests. Our experience shows the merely turning off optimizations does not guarantee that transformed C code will be PC-secure after compilation. Details of our checker's construction and operation may be found in the full version of the paper [26].

6 Related Work

Many previous side channel defenses are application-specific. For example, blinding can be used to prevent timing attacks against RSA [17, 9]. The major advantage of an application-specific defense is that it can be efficient. Experimental measurements show that blinding only adds a 2–10% overhead; contrast this with the overhead we measured in § 5.1.

Unfortunately, no proof of security for blinding against side channel attacks is known. In the absence of proof, it is difficult to assess whether the defense works. For example, defenses were designed for the five AES finalists [21]. These defenses had no formal model of information leaked to the adversary and hence no way to verify security. In fact, Coron and Goubin later pointed out a subtle flaw in one of the techniques [11]. Blömer, et al., give several more examples of techniques that were thought to be secure but failed, and

of cases where innocent-looking "simplifications" failed. These examples motivate their study of provably secure defenses against side channels [8]. We note that Hevia and Kiwi showed that conditional branches in some implementations of DES leak information about the secret key; this is another motivation for PC-security.

Chevallier-Mames, Ciet, and Joye show a clever construction for performing modular exponentiation without incurring undue overhead. They show that, under an appropriate physical assumption, only the Hamming weight of the exponent is leaked by their algorithm. Blömer, et al., also define a model for provable security against side channel attacks and show how to implement AES securely in this model [8]. While these methods are a step forward, they still require a great deal of new effort for each new application.

The programming languages community has studied the problem of secure information flow extensively, but most work regarding C code has focused on detecting covert channels and side channels [29], not on eliminating them via code transformation. One exception is Agat's work, which transforms out timing leaks by inserting dummy instructions to balance out the branches in a program's control-flow graph [1, 2]. His work is focuses primarily on timing attacks, while our approach is more general. There are also languages such as Jif and Flowcaml that include information flow support as part of the language [33, 31].

Micali and Reyzin examine "physically observable cryptography" through a framework that is closely related to ours. Their model specifies a "leakage function" (analogous to our notion of transcript) and allows the adversary to measure the outputs of this leakage function on a "physical machine" which may be executing some number of "virtual Turing Machines." Our model, in contrast, is simpler since we consider only a single program executing at a time. Also, Micali and Reyzin focus more on how side channel attacks affect basic theorems of cryptography, while we are interested in automatic transforms that improve security against such attacks [24].

The above defenses focus on software; there are also promising solutions that focus on hardware [30, 3, 4, 28]. To coordinate these defenses, we need a contract between hardware researchers and software researchers as to who will protect what. Our transcript is exactly this: a contract specifying what information the software can expect the hardware to leak to the adversary.

7 Conclusion and Open Problems

We presented a program counter model for reasoning about side channel attacks, a system that transforms code to increase resistance against attacks, and a static verifier that checks the code output by our compiler is PC-secure. This framework allows us to prove transformed code is secure against an interesting class of side channels. With enough work, an even more efficient automatic transformation for PC-security may be possible.

Looking forward, it is an interesting open problem to extend these methods to handle a larger class of side-channel attacks. We have argued that specifying

a transcript model as part of the hardware/software interface simplifies development of both hardware and software countermeasures. We leave it as an open problem to find the "right" contract between these two worlds.

Acknowledgments

We thank Nikita Borisov, Eric Brewer, Karl Chen, Evan Chang, Adam Chlipala, Rob Johnson, Chris Karlof, Naveen Sastry, Rusty Sears, Umesh Shankar, and Fran Woodland for discussions and advice. David Molnar was supported by an Intel OCR Fellowship and a National Science Foundation Graduate Fellowship. This work supported by NSF ANI-0113941 and NSF CCR-0325311.

References

1. Johan Agat. Transforming Out Timing Leaks. In *Proceedings on the 27th ACM Symposium on the Principles of Programming Languages*, 2000.
2. Johan Agat. *Type Based Techniques for Covert Channel Elimination and Register Allocation.* PhD thesis, Chalmers University of Technology, 2001.
3. Luca Benini, Alberto Macii, Enrico Macii, Elvira Omerbegovic, Massimo Poncino, and Fabrizio Pro. A Novel Architecture for Power Maskable Arithmetic Units. In *Proceedings of the 13th ACM Great Lakes symposium on VLSI*, 2003.
4. Luca Benini, Alberto Macii, Enrico Macii, Elvira Omerbegovic, Massimo Poncino, and Fabrizio Pro. Energy-aware Design Techniques for Differential Power Analysis Protection. In *Proceedings of the 40th conference on Design automation*, 2003.
5. D.J. Bernstein. Cache-timing attacks on AES, 2005. `http://cr.yp.to/papers.html#cachetiming`.
6. John Black and Hector Urtubia. Side-Channel Attacks on Symmetric Encryption Schemes: The Case for Authenticated Encryption. In *Proceedings of the 11th USENIX Security Symposium*, 2002.
7. D. Bleichenbacher. Chosen ciphertext attacks against protocols based on RSA encryption standard PKCS #1. In *CRYPTO*, 1998.
8. Johannes Blömer, Jorge Guajardo Merchan, and Volker Krummel. Provably secure masking of AES. In *SAC*, 2004.
9. Dan Boneh and David Brumley. Remote Timing Attacks Are Practical. In *Proceedings of the 12th USENIX Security Symposium*, 2003.
10. Suresh Chari, Charanjit Jutla, Josyula R. Rao, and Pankaj Rohatgi. A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards. In *Proceedings of the Second AES Candidate Conference*, 1999.
11. Jean-Sébastien Coron and Louis Goubin. On Boolean and Arithmetic Masking Against Differential Power Analysis. In *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems*, 2000.
12. H. Handschuh and H. Heys. A timing attack on RC5. In *Lecture Notes in Computer Science: Selected Areas in Cryptography*, pages 306–318. Springer-Verlag, 1999.
13. Matthew Hennessy. *The Semantics of Programming Languages: an Elementary Introduction using Structural Operational Semantics.* John Wiley and Sons, New York, N.Y., 1990.
14. John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Side Channel Cryptanalysis of Product Ciphers. *Journal of Computer Security*, 8:141–158, 2000.

15. Vlastimil Klima, Ondrej Pokorny, and Tomas Rosa. Attacking RSA-based sessions in SSL/TLS. In *CHES*, 2003.
16. Vlastimil Klima and Tomas Rosa. Side channel attacks on CBC encrypted messages in the PKCS #7 format. Cryptology ePrint Archive, Report 2003/098, 2003. `http://eprint.iacr.org/`.
17. Paul Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Proceedings of the 16th Annual International Cryptology Conference*, 1996.
18. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Proceedings of the 19th Annual International Cryptology Conference*, 1999.
19. Butler W. Lampson. A Note on the Confinement Problem. *Communications of the ACM*, 16(10):613–615, 1973.
20. J. Manger. A chosen ciphertext attack on RSA optimal asymmetric encryption padding (OAEP) as standardized in PKCS #1 v2.0. In *CRYPTO*, 2001.
21. Thomas S. Messerges. Securing the AES Finalists Against Power Analysis Attacks. In *Proceedings of the Fast Software Encryption Workshop*, 2000.
22. Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. Investigations of Power Analysis Attacks on Smartcards. In *Proceedings of the USENIX Workshop on Smartcard Technology*, 1999.
23. Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. Power Analysis Attacks of Modular Exponentiation in Smartcards. In *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems*, 1999.
24. Silvio Micali and Leo Reyzin. Physically observable cryptography. In *Theory of Cryptography*, 2004.
25. Bodo Möller. Security of CBC ciphersuites in SSL/TLS: Problems and countermeasures, May 2004. `http://www.openssl.org/~bodo/tls-cbc.txt`.
26. David Molnar, Matt Piotrowski, David Schultz, and David Wagner. The program counter security model: Automatic detection and removal of control-flow side channel attacks (Full Version), 2005. IACR eprint archive report 2005/368.
27. George Necula, Scott McPeak, S.P. Rahul, and Westley Weimer. CIL: Intermediate Language and Tools for Analysis and Transformation of C Programs. In *Proceedings of the Conference on Compilier Construction*, 2002.
28. Patrick Rakers, Larry Connell, Tim Collins, and Dan Russell. Secure Contactless Smartcard ASIC with DPA Protection. In *Proceedings of the Custom Integrated Circuits Conference*, 2000.
29. Andrei Sabelfeld and Andrew C. Myers. Language-Based Information-Flow Security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003.
30. Adi Shamir. Protecting Smart Cards from Passive Power Analysis with Detached Power Supplies. In *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems*, 2000.
31. Vincent Simonet. Flowcaml, 2005. `http://cristal.inria.fr/~simonet/soft/flowcaml/`.
32. S. Vaudenay. Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS... In *EUROCRYPT*, 2002.
33. Lantian Zheng and Andrew Myers. End-to-end availability policies and noninterference, 2005. Computer Security Foundations Workshop.

A Specifying the Transform

We are now ready to specify the transform more precisely. As we discussed, our implementation handles all of the C language. However, the lack of a formal semantics for C makes it difficult to prove anything about C, so we focus on a subset of C that contains most of the language features that are relevant to our analysis. For this subset, we can prove that the transform is semantically preserving and that it produces PC-secure code.

To precisely capture this subset of C, we introduce IncredibL, a simple imperative language with restricted control flow. IncredibL is our own invention, but it is derived from Hennessy's WhileL [13]. The grammar for IncredibL can be found in Fig. 3.

Roughly, IncredibL captures a memory-safe subset of C with only bounded loops, `if` statements, and straight-line assignments. Note that we do not allow any forms of recursion or unstructured control flow, as these may introduce unbounded iteration. We also disallow calls to untransformed subroutines, including I/O primitives. Note that because loop bounds are known statically in IncredibL, we can in principle unroll all loops in any IncredibL program to obtain code with no branches.

Our transformation $\mathcal{T}_{\text{Program}}$ is specified in Fig. 4. We state the main theorems here. Proofs can be found in full version [26].

$$
\begin{aligned}
C &\in \text{Com} = \text{Program} \\
E &\in \text{Exp} \\
B &\in \text{BoolExp} \subset \text{Exp} \\
I &\in \text{Identifier} \\
\text{arithop} &\in \text{AOp} = \{\texttt{+}, \texttt{-}, \texttt{*}, \texttt{\&}, \texttt{|}\} \\
\text{relop} &\in \text{RelOp} = \{\texttt{>}, \texttt{<}, \texttt{=}\} \\
\text{boolop} &\in \text{BoolOp} = \{\texttt{and}, \texttt{or}\} \\
n &\in \text{Num}
\end{aligned}
$$

(a) Syntactic domains.

$$
\begin{aligned}
C ::= {} & \\
& I \texttt{ := } E \mid C'\texttt{; } C'' \mid \texttt{if } B \texttt{ then } C' \texttt{ else } C'' \\
& \mid \texttt{for } I \texttt{ := } n \texttt{ to } n' \texttt{ do } C' \mid \texttt{break} \\
E ::= {} & \; I \mid n \mid B \mid E' \text{ arithop } E'' \mid \texttt{\~{}}E' \\
B ::= {} & \\
& 0 \mid 1 \mid B' \text{ boolop } B'' \mid E' \text{ relop } E'' \mid \texttt{!}B'
\end{aligned}
$$

(b) Grammar.

Fig. 3. The abstract syntax of IncredibL

Theorem 3. *$\mathcal{T}_{Program}$ is semantics-preserving. for every IncredibL program P, $\mathcal{T}_{Program}[\![P]\!]$ consists only of straight-line code and loops with straight-line code bodies that run for a fixed constant number of iterations with no assignments to induction variables.*

Corollary 1. *$\mathcal{T}_{Program}$ enforces PC-security. for every IncredibL program P, $\mathcal{T}_{Program}[\![P]\!]$ is PC-secure.*

$$
\begin{aligned}
\mathcal{T}_{\text{Program}}[\![C]\!] &= I_0 \texttt{ := -1; } \mathcal{T}_{\text{Com}}[\![C]\!](I_0, I_0) \quad \text{where } I_0 \text{ is a fresh identifier} \\
\mathcal{T}_{\text{Com}}[\![I \texttt{ := } E]\!](I_{\text{if}}, I_{\text{brk}}) &= \textit{conditional-assign}(I, \texttt{(}I_{\text{if}} \texttt{ \& } I_{\text{brk}}\texttt{)}, E, I) \\
\mathcal{T}_{\text{Com}}[\![C\texttt{; } C']\!](I_{\text{if}}, I_{\text{brk}}) &= \mathcal{T}_{\text{Com}}[\![C]\!](I_{\text{if}}, I_{\text{brk}})\texttt{; } \mathcal{T}_{\text{Com}}[\![C']\!](I_{\text{if}}, I_{\text{brk}}) \\
\mathcal{T}_{\text{Com}}[\![\texttt{if } B \texttt{ then } C \texttt{ else } C']\!](I_{\text{if}}, I_{\text{brk}}) &= \textit{conditional-assign}(I_0, \texttt{(}I_{\text{if}} \texttt{ \& } I_{\text{brk}}\texttt{)}, \texttt{(0-}B\texttt{)}, \texttt{0})\texttt{; } \mathcal{T}_{\text{Com}}[\![C]\!](I_0, I_{\text{brk}})\texttt{;} \\
&\quad \textit{conditional-assign}(I_0, \texttt{(}I_{\text{if}} \texttt{ \& } I_{\text{brk}}\texttt{)}, \texttt{\~{}}I_0, \texttt{0})\texttt{; } \mathcal{T}_{\text{Com}}[\![C']\!](I_0, I_{\text{brk}}) \\
&\qquad \text{where } I_0 \text{ is a fresh identifier} \\
\mathcal{T}_{\text{Com}}[\![\texttt{for } I \texttt{ := } n \texttt{ to } n' \texttt{ do } C]\!](I_{\text{if}}, I_{\text{brk}}) &= \textit{conditional-assign}(I_0, \texttt{(}I_{\text{if}} \texttt{ \& } I_{\text{brk}}\texttt{)}, \texttt{-1}, \texttt{0})\texttt{;} \\
&\quad \texttt{for } I \texttt{ := } n \texttt{ to } n' \texttt{ do } \mathcal{T}_{\text{Com}}[\![C]\!](I_{\text{if}}, I_0) \\
&\qquad \text{where } I_0 \text{ is a fresh identifier} \\
\mathcal{T}_{\text{Com}}[\![\texttt{break}]\!](I_{\text{if}}, I_{\text{brk}}) &= \textit{conditional-assign}(I_{\text{brk}}, \texttt{(}I_{\text{if}} \texttt{ \& } I_{\text{brk}}\texttt{)}, \texttt{0}, I_{\text{brk}}) \\
\textit{conditional-assign}(I, E_m, E_t, E_f) &= I \texttt{ := (}E_t \texttt{ \& } E_m\texttt{) | (}E_f \texttt{ \& \~{}}E_m\texttt{)}
\end{aligned}
$$

Fig. 4. A formal specification of our transform

The Dilemma of Covert Channels Searching*

Changda Wang[1,2] and Shiguang Ju[1]

[1] School of Computer Science and Telecommunications Engineering,
Jiangsu University, Zhenjiang, Jiangsu 212013, China
[2] School of Computer Science, Carleton University,
Ottawa, Ontario K1S 5B6, Canada
wangchangda@hotmail.com

Abstract. Covert channel is a famous drawback exists in most of multilevel security systems. Both TESEC and CC standards need covert channel analysis when secure software tries to get the certification of some security levels, i.e. B2 and EAL5 or above in TCSEC and CC, respectively. Search method is one of the most important works with ad hoc characters in covert channels analysis. Though some semi auto tools have been built, peoples who work in this area are eager to develop an auto search tool to find all of covert channels since it was first known in 1973. This paper proves that willingness is a kind of undecidable problems, by which illustrates it's impossible to build a program which can identify all of covert channels in a security computer system automatically.

1 Introduction

Security computer system use both mandatory and discretionary access controls to restrict the flow of information through legitimate communication channels such as files, shared memory, and process signals. Unfortunately, in practice one finds that computer systems are built such that users are not limited to communicating only through the intented communication channels. These illegitimate channels are known as covert channels [1]. From this point of view, covert channels can be defined as those that use entities not normally viewed as data objects to transfer information from one subject to another [2].

Covert channel analysis is a necessary work required by both TCSEC and CC when secure software tries to get the certification of some security levels, i.e. B2 and EAL5 or above in TCSEC and CC, respectively. Covert channel analysis can be categorized as three parts, viz. search, audit and elimination, in which search is the most booming research area. Although some search methods provide semi automatic tools to help analyzers to identify covert channels, all of these methods need human intervention [3]. Can an automatic tool that can find all of the covert channels within a security computer system be built? This paper discusses the relationship between covert channels searching and undecidable problem, by which to illustrate theoretically that's impossible.

* This work was supported in part by National Natural Science Foundation of China (No.60573046).

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 169 – 174, 2006.
© Springer-Verlag Berlin Heidelberg 2006

In the following pages, what characters should a covert channels' searching program has is discussed. Section 3 proves that searching covert channels is a kind of undecidable problem. In section 4, the relationship between covert channels and ambiguous of context-free grammar is discussed. It provides another point of view to understand the conclusion in section 3. Finally, section 5 concludes the paper and gives some advices on covert channels searching.

2 The Characters of Searching Program

Before going further we should understand why covert channels use entities not normally viewed as data objects to transfer information from one subject to another? Why don't they communicate directly?

Generally, in a multilevel security system, classified information is only permitted flows from low security level to higher. By this weird method, actually, even under the supervision of security model, covert channels can be used to leak sensitive information from high security level to lower [4, 5]. Most of multilevel security models have the problem of covert channels, e.g. BLP (Bell-La Padula) model.

Provided an amazing program that can identify covert channels automatically in multilevel security system has been built, what characters should it has?

Lemma 1: If a program can identify covert channels automatically in multilevel security system, it has the ability to tell the difference among security levels.

Proof: If a program can identify covert channels automatically, that is to say, this program knows there is a channel by which information flows from high security level to lower, i.e. this program can distinguish the security levels difference.

Lemma 2: If a program can tell the difference among security levels, this program is a covert channel.

Proof: Assumes that subject $\boldsymbol{A}$ has higher security level than that of $\boldsymbol{B}$. Under BLP model, sensitive information only can flow from low security level to higher, i.e. from $\boldsymbol{B}$ to $\boldsymbol{A}$.

If $\boldsymbol{B}$ has a program can tell difference among security levels, then $\boldsymbol{B}$ and $\boldsymbol{A}$ can establish a covert channel by following method,

- $\boldsymbol{A}$ creates an object with security level $\boldsymbol{L}$, by which send "1" to $\boldsymbol{B}$
- $\boldsymbol{A}$ creates an object with security level $\boldsymbol{L'}$($\boldsymbol{L} \neq \boldsymbol{L'}$), by which send "0" to $\boldsymbol{B}$

Because $\boldsymbol{B}$ has the program which knows security levels difference, so $\boldsymbol{B}$ can decode the content sent by $\boldsymbol{A}$.

Theorem 1: If a program can identify covert channels automatically in multilevel security system, this program is a covert channel.

Proof: By Lemma 1 and Lemma 2.

Corollary 1: None of programs can illustrate itself doesn't contain covert channels.

Now we knew that a program can identify covert channels automatically in a multilevel security system if and only if this program is a covert channel. From this

point of view, embed a program into operating system by which to search covert channels [4] will introduce a new covert channel into the whole system.

3 Halting Problem vs. Covert Channels Searching

At first glance, it's so disappointed that an auto searching program itself is a covert channel. It seems embed an auto searching program will do nothing but introduce a new covert channel into the whole system. Is that true? Yes, but don't forget it's a program can be used to search covert channels automatically. Just like a double-edged sword, the usage depends on user and so does auto searching program.

If only trusted subjects have rights to access the searching program, i.e. the malicious subjects can't use it to construct covert channels, which will be very useful for security administrator to identify covert channels in a multilevel security system. Can an auto covert channels searching program be built?

Theorem 2: None of programs can identify covert channels total automatically in a multilevel security computer system.

Proof: Assumes *CC_Detector ( )* is an auto covert channels searching program. *CC_Detector ( )* will stop if and only if it finds a covert channel.

```
Bool CC_Detector (Char *program, Char *input) {
     If (Covert channel detected)
          Return TRUE; //Stop
     Else
          Return FALSE;
}
```

Detect_in_Self ( ) is a program that can detect covert channels in itself.

```
Bool Detect_in_Self (Char * program) {
     Return CC_Detector (program, program);
}
```

Thinks the flowing program,

```
Prog_Is_CC (Char *program) {
   If (Detect_in_Self (program)) {
        While (TRUE) {};
        Return FALSE;
  } Else {
         Return TRUE;  //Stop
         }
}
```

What happens when we try *Prog_Is_CC ( )* on itself? Well, clearly one of two things can happen: either it runs forever, or it stops and returns true, depending on whether the call to *Detect_in_Self ( )* returns true or false.

If *Prog_Is_CC (Prog_is_cc)* goes into an infinite loop, it is because *Detect_in_Self (Prog_is_cc)* returned true, which means that *CC_Detector (Prog_is_cc, Prog_is_cc)* returned true. But this means that *Prog_Is_CC ( )* would stop when fed itself as input. This contradicts the assumption that is goes into an infinite loop.

If *Prog_Is_CC (Prog_is_cc)* stops and returns true, it's because *Detect_in_Self (Prog_is_cc)* returned false, which means that *CC_Detector (Prog_is_cc, Prog_is_cc)* returned false. But this means that *Prog_Is_CC ( )* would run forever when fed itself as input. This contradicts the assumption that is terminates.

We have therefore led ourselves to a contradiction. *Prog_Is_CC ( ) s*tops if and only if it runs forever. Since this impossible, one of our assumptions must be invalid. By tracing our reasoning backwards, we find that it is therefore impossible to have written *CC_Detector ( )* in the first place, i.e. *CC_Detector ( )* doesn't exist. That is to say, auto covert channels searching program can't be built.

Actually, A. Turing used similar method to prove that halting problem is undecidable. The same idea is used in here, by which you can understand the difficulty of building such a program is equal to solve halting problem, viz. it's impossible.

4 Covert Channels vs. Ambiguous Grammar

Although an auto searching program can't be built has proved in previous section, the relationship between covert channels and ambiguous context-free grammar will be discussed as follows, by which gives another way to understand the same question.

4.1 The Strategy of Covert Channels

Before going further we should understand how covert channels can leak sensitive information from high security level to lower under the supervision of security models. Each security model builds on the foundation of a smaller trusted code, i.e. Trust Compute Base (TCB). All information flows are inspected by security model according to security policy except some carried by TCB [5]. That is to say, in a multilevel security system, e.g. a system applying BLP model, some information can flow from high security level to lower even that prohibited by security policy. It's worth mentioning that most of this degrade information is *Ack* signals or something like that. Without these signals the system will be unstable and fragile.

From this point of view, it's not easy to send classified information from high security level to lower. Covert channels make duplicity use of *Ack* signals to leak sensitive information, i.e. via dominates some shared resource to frustrate or not frustrate the operations from a subject *Receiver*, which has low security level, subject *Sender* with higher security level can send any digital information to *Receiver*. For instance, if *Sender* frustrates the operation from *Receiver* send "1", and whereas send "0", any digital information can be sent from high security level to lower. Furthermore, if there is a coding book between *Sender* and *Receiver*, the communications content would be scrambled, which will frustrate most of works that try to identify covert channels by semantic analysis.

4.2 The Similarity Between Covert Channels and Ambiguous Grammar

A context-free grammar ***G*** is ambiguous if there is a word in L(***G***) possessing two leftmost derivations from the initial letter [6]. Otherwise, ***G*** is unambiguous.

As mentioned above, via covert channel, a subject sends classified information to another subject with lower security level, some *Ack signals* are carriers. So the *sender* and *receiver* of the same covert channel look these *Ack signals* as a serials of "1" and "0", whereas the security model believes there are nothing but *Ack signals*. This provides the foundation of ambiguous.

Assumes grammar $G : S \to A \mid B \mid C$

$$A \to m \mid n \quad \& \quad B \to b \quad \& C \to c$$

Where, the security model and covert channel look A as m and n ($m \neq n$), respectively. For instance, A can be looked as an *Ack signal* in multilevel security system.

So for any sentence generates by grammar G include A, e.g. ABC, it has two leftmost derivations, i.e. mBC and nBC.

Actually, none methodology can prove a grammar isn't an ambiguous one [6]. That is to say, the difficulty of auto covert channels searching is equal to an unsolvable problem. Although auto covert channels searching are not proved equal to the ambiguous problem in context-free grammar, if ambiguous grammar can be solved, it will provide a new way to conquer auto covert channels searching.

5 Conclusions

The dilemma of covert channels searching is discussed with different point of views. Build an auto searching program for covert channels is impossible. Furthermore, even this program can be built, it will introduce a new covert channel into the whole system.

Although many researches have been done in this area and some semi-auto tools were built for covert channels searching [1, 2, 7, 8], none of them is a total automatic searching program. Human intervention is a permanent part of every covert channel searching programs; the difference among them is how much does it depend on human being. In short, we should pay more attention on how to decrease the bald work for covert channel analyst instead of dreaming a total automatic searching tool.

References

1. Richard A. Kemmerer, Phillip A. Porras. *Covert Flow Trees: A Visual Approach to Analyzing Covert Storage Channels,* IEEE Transactions on Software Engineering, VOL 17. No. 11, November 1991, pages: 1166~1184
2. Richard A. Kemmerer. *Shared resource matrix methodology: A practical approach to identifying covert channels.* ACM Transactions on Computer Systems.1(3), August 1983, pages: 256-277.
3. Changda Wang, Shiguang Ju, Dianchun Guo, Zhen Yang and Wenyi Zheng, *Research on the methods of search and elimination in covert channel*, Grid and Cooperative Computing, LNCS, PT 1 3032, 2004, pages: 988-991

4. Changda Wang and Shiguang Ju, *Searching Covert Channels by Identifying Malicious Subjects in the Time Domain*, 5th IEEE Information Assurance Workshop, 9-11, June 2004, NY, U.S.A, pages: 68-73
5. Changda Wang and Shiguang Ju, *The minimum criteria of covert channels' existence and its application*, Journal of Computer Science, 2005, Vol.32, No. 1, pages: 77-79
6. Arto salomaa, *Formal language*, ACM Monography Series, Academic Press, New York and London, 1973
7. C. R. Tsai, Virgil D. Gligor, and C. S. Chandersekaran, *On the identification of covert storage channels in secure systems*, IEEE Transactions on Software Engineering, Vol. 16, No.6 JUNE 1990, pages: 569-580
8. C. R. Tsai, Virgil D. Gligor, and C. S. Chandersekaran. *A formal method for the identification of covert storage channels in source code.* In 1987 IEEE Symposium on Security and Privacy, Oakland, CA, April 1987. IEEE Computer Society, Computer Society Press, pages: 74-86
9. http://www.cgl.uwaterloo.ca/~csk/halt/

A Probabilistic Approach to Estimate the Damage Propagation of Cyber Attacks

Young-Gab Kim[1], Taek Lee[1], Hoh Peter In[1,*], Yoon-Jung Chung[2], InJung Kim[2], and Doo-Kwon Baik[1,*]

[1] Department of Computer Science and Engineering Korea University, 1, 5-ga, Anam-dong, SungBuk-gu, 136-701, Seoul, Korea
{always, comtaek, hoh_in, baikdk}@korea.ac.kr
[2] Electronics and Telecommunications and Research Institute, 463 Jeonmin-dong Yuseong-gu Daejeon, Korea
{yjjung, cipher}@etri.re.kr

Abstract. With rapid development in the Internet technology, business management in an organization becomes dependent on network dependency and cohesiveness in a critical information and communications infrastructure. However, the occurrence of cyber attacks has increased, targeted against vulnerable resources in information systems. Hence, in order to protect private information and computer resources, risk analysis and damage propagation need to be studied. However, the existing models present mechanisms for risk management, and these models can only be applied to specified threats such as a virus or a worm. Therefore, a probabilistic model for damage propagation based on Markov process is proposed, which can be applied to diverse threats in information systems. The proposed model enables us to predict the occurrence probability and occurrence frequency of each threat in the information systems.

1 Introduction

With rapid development of the Internet technology, business management in an organization or an enterprise depends the Internet-based technology for critical operations. Furthermore, as the occurrence of cyber attacks against vulnerable resources in information systems has also increased, damage is increasing. Hence, in order to protect critical information and computer resources, damage propagation need to be researched. Damage propagation models are used to select appropriate security policies through risk analysis [1] and estimate future damages of the attacks to protect critical assets of an organization effectively.

A few damage propagation models have been proposed, whereas quite a few risk analysis models have been studied in [1][2][3]. However, the existing damage propagation models are limited and inadequate in analyzing cyber attacks caused by various threats, because it has focus on only a specific threat such as a virus or a

* Corresponding authors.

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 175 – 185, 2006.
© Springer-Verlag Berlin Heidelberg 2006

worm. Therefore, it is difficult to holistically analyze the scope and velocity of damage propagation caused by various threats. Therefore, in this paper, a probabilistic model for damage propagation based on the Markov process [4][5] is proposed for the holistic analysis based on past data. Using our proposed model, the occurrence probability and occurrence frequency of each threat in information systems can be estimated holistically, and applied to establish countermeasures against those threats.

The sebsequent sections of this paper are organized as follows: In Section 2, the background relating to the Markov process is presented. Section 3 shows the approach method to design the probabilistic model of damage propagation. Section 4 presents the probabilistic model of damage propagation based on the Markov process, and explains the detailed procedures of the model. In addition, a case study to show the creation of the model is presented. Section 5 shows the related work, including the worm propagation model. Section 6 concludes this paper.

2 Background

In this section, a Markov process is explained, and applied to design the proposed damage propagation model. A Markov process is a stochastic process that the next states are determined by only the current state itself, not by a set of the states (history) reaching to the current state [4]. It is called a memoryless process because it does not memorize the past states. If X(t) is a Markov process at time step $t_1 < t_2 < \ldots < t_k < t_{k+1}$, the Markov property can be succinctly stated as following:

$$P[a<X(t_{k+1}) = x_{k+1} | X(t_k) = x_k , \ldots , X(t_1) = x_1]$$
$$= P[X(t_{k+1}) = x_{k+1} | X(t_k) = x_k]$$

If a value of the Markov process is a discrete-value, it is called a Markov chain. Depending on whether the time *t* is discrete or continuous in the Markov chain, the Markov chain can be divided into a discrete-time Markov process and a continuous-time Markov chain.

The Markov process is a system, can be explained as consisting of the following elements:

- A set of states: the set of all possible values from the process
- π vector (initial probability): the initial probability of each state
- State transition matrix: the probabilities of moving from an state to another in a dynamic system

In this paper, the damage propagation model is designed by defining the elements of the Markov process. That is, a set of states, the initial probability and the state transition matrix are defined clearly.

3 The Overview of the Proposed Damage Propagation Model

Types of damage propagation are formed based on the a kind of threats (that is, T_1, T_2, … ,T_8), the relationship among each other, and the organization countermeasures against threat behavior as shown in Fig.1.

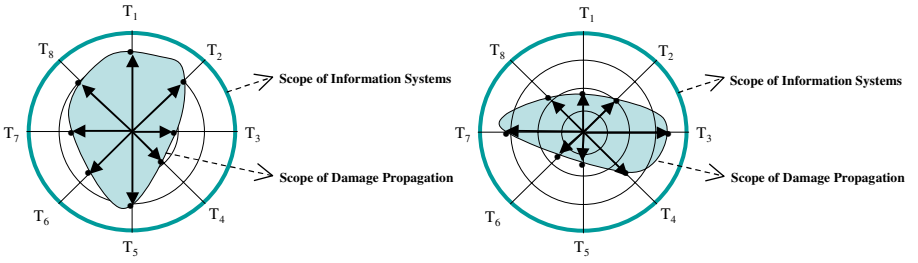


Fig. 1. Types of damage propagation

The probability model proposed in this paper can apply to many kinds of threats. Therefore, it is possible to analyze the relationship among the threats. To design the probabilistic model of damage propagation based on the Markov process, several assumptions are required: First, the past data are sufficient and credible. The collection of the past data is very important to design the Markov process-based model. Second, the damage propagation of an information asset is spread by the discrete time event. That is, unlike biological infection, the spread of damage in information systems is propagated by specific events or discrete time. Third, the information system has more than one threat. Therefore the damage propagation model should be able to calculate the damage of each threat. Finally, the organization or people can establish countermeasures against threats and deal it with them dynamically.

A probabilistic (Markov-based) model of damage propagation estimates a degree of the spreading damage from a single system to its dependent systems. The damage is determined by the speed and scope of its propagation, which are changed over the time or by a kind of threat. However, the existing damage propagation models especially has limited to only the speed of damage propagation over the time. To overcome this limitation, the Markov process is applied to estimate not only the speed but also the scope of its propagation. In our proposed damage propagation model, the frequency and probability of threat-occurrence are used to estimate the speed and scope. In this paper, the following formula (1) is used to express the damage propagation quantitatively.

$$\mathrm{RISK}(s, t) = \mathrm{Loss}(s, t) \times \mathrm{Probability}(s, t) \tag{1}$$

RISK means a damage amount when assets are damaged by threats in a vulnerable system. Furthermore consideration is given scope *s*, and time *t* in this formula. The scope means the size of damage propagation, and is decided by the propagation speed or frequency of each threat. The time *s* is an element for considering over time, this also affects the scope of damage propagation. As a result, damage propagation is calculated as multiplication of loss, which means the degree of shrinkage in the asset-value caused by threats, and the probability, which is the probability of threat occurrence. In this paper, the focus is on calculating the probability of threat occurrence.

4 The Proposed Damage Propagation Model Based on the Markov Process

In this section, the procedure to construct the probabilistic model for damage propagation is explained in detail. The proposed model in this paper is created as presented Fig. 2, and is composed of 4 steps: State Definition, Threat-state Transition Matrix, Initial Vector, and Threat Prediction.

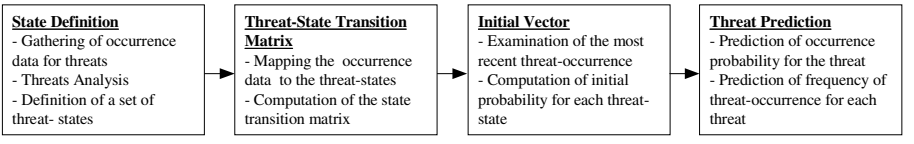


Fig. 2. Procedure for creation of probabilistic model for damage propagation

In the 'State Definition' step, the threat-states, which can occur in the critical information and communication infrastructure, are defined. A set of the states is defined as scopes of a threat-state or combination of many threat-states. In the 'Threat-State Transition Matrix' step, a transition matrix is calculated among the threat-states using the threat-occurrence data and the defined threat-states in the previous step. In the 'Initial Vector' step, the initial probability is calculated against the threat-state's occurrence. Finally, in the 'Threat Prediction' step, the probability and frequency of threat-occurrence using the threat-state transition matrix and the initial vector calculated in previous steps are estimated. A more detailed description will be presented in the following subsections.

4.1 State Definition (Step 1)

In the 'States Definition' step, three tasks are performed to define the threat-states: The gathering of occurrence data of threats, threat analysis, and definition of a set of threat-state. That is, in this step, all kinds of threats are looked for, the threat-occurrence data are collected and analyzed in information systems, and finally the possible threat-states can be defined. If S is a set of threat-sate, S can be defined as formula (2).

$$S= \{S_1, S_2, \dots , S_n\} \tag{2}$$

Especially it is very important the collection of the reliable and huge past data related with the threats because the past data are more important than other elements in the probability model based on the Markov process. Therefore, in this paper, a case study in the subsection 4.5, the statistics of hacking and virus published by the Korea Information Security Agency (KISA) were used for 54 months, from January 2001 to June 2005, for trust in the past data [6].

The threat analysis task is an important process to analyze potential threats, which can damage an information system. Through this task, the threats are classified into several kinds of threats, and a threat-occurrence is investigated. Furthermore the priority for each threat is given, considering the criticalness of threat. This helps prepare the effective countermeasures for critical threats, which must take the priority.

The definition of a threat-states task decides the threat-states by analyzing threat-occurrence, and establishing a threshold that means the range of the frequency of threat-occurrence. How to define the set of threat-state can be divided into two methods, according to the dependency among threats. That is, when a threat occurs independently of others, the set of threat-state is composed of as a number of several thresholds. Conversely, when a threat occurs that relates with other threats, the set of threat-state is created with the combination of thresholds for each threat. Therefore, in the case of the latter, the number of threat-state and the complexity of transition matrix, which describes the probabilities of moving from one state to another, will increase in proportion with the number of threat-states.

4.2 Transition Matrix of Threat-State (Step 2)

In Step 2, the threat-state transition matrix is calculated, which is a square matrix describing the probabilities of moving from one threat-state to another. To obtain the transition matrix, two tasks are performed: First, threat-states are listed by mapping the threat-occurrence data of each threat into the threat-state defined in the previous step. Second, the number from one threat-state to another is counted, and finally the matrix is constructed.

Like Step 1, the method of creating a transition matrix is divided into two, according to the dependency among threats. In the case of a threat occurring independently, the transition matrix can be created simply with the two tasks mentioned previously. However, in the case a threat occurs that relates to others, the size and complexity of the threat-transition matrix are increased, depending on the number of related threats and the threat-state defined in Step1. Therefore, to reduce complexity and the size of the transition matrix, it is very important to decide the proper number of threat-states in Step 1.

If P is the transition probability matrix created in this step, it is compactly specified as the form of (3). Furthermore the entries of the matrix P satisfy the property (4).

$$P = \begin{bmatrix} P_{11} & P_{12} & \ldots & P_{1n} \\ P_{21} & P_{22} & \ldots & P_{2n} \\ \ldots & \ldots & P_{ij} & \ldots \\ P_{n1} & P_{n2} & \ldots & P_{nn} \end{bmatrix} \tag{3}$$

$$\sum_{j=1}^{n} P_{1j} = 1, \sum_{j=1}^{n} P_{2j} = 1, \ldots, \sum_{j=1}^{n} P_{nj} = 1. \text{ That is, } \sum_{j=1}^{n} P_{ij} = 1, i = 1,2,\ldots,n \tag{4}$$

In each row, the probabilities of moving from the state represented by that row, to the other states, are shown. Thus the rows of a Markov transition matrix each add up to one.

4.3 Initial Probability (π Vector) (Step 3)

Step 3 is a process to obtain the initial probability vector, which represents the occurrence possibility of each threat-state in the initial state. To obtain the initial probability, the most recent threat-occurrence data are used, which can be divided by the unit

of month such as three months, six months, nine months and one year. Through analyzing the most recent data, the initial probability vector is calculated using the formula (5) satisfied by the condition (6).

$$P(S_1 \quad S_2 \quad \ldots \quad S_k \quad \ldots \quad S_n) = P(\frac{\alpha}{F} \quad \frac{\beta}{F} \quad \ldots \quad \frac{\gamma}{F} \quad \ldots \quad \frac{\delta}{F}) \tag{5}$$

$$F = \sum_{i=1}^{n} f_i = \alpha + \beta + \ldots + \gamma + \ldots + \delta \tag{6}$$

where α, β, γ and δ represent the number of threat-occurrence for each state, S_1, S_2, S_k and S_n. Furthermore the initial probability $P(S_i)$ for each state S_i satisfy the formula (7) because the sum of initial probability must add up to one.

$$\sum_{i=1}^{n} P(S_i) = 1 \tag{7}$$

4.4 Prediction of Threats (Step 4)

In Step 4, the probability and frequency of threat-occurrence is estimated, which will occur in the future, using the transition matrix created in Step 2 and the initial probability vector created in Step 3. Formula (8) depicts the computation of probability of threat-occurrence.

$$P(S_1 \quad S_2 \quad \ldots \quad S_k \quad \ldots \quad S_n) \begin{bmatrix} P_{11} & P_{12} & \ldots & P_{1n} \\ P_{21} & P_{22} & \ldots & P_{2n} \\ \ldots & \ldots & P_{ij} & \ldots \\ P_{n1} & P_{n2} & \ldots & P_{nn} \end{bmatrix} = P'(S_1 \quad S_2 \quad \ldots \quad S_k \quad \ldots \quad S_n) \tag{8}$$

where n is the number of threat-state, $P(S_i)$ is the initial probability for each threat-state, and $P'(S_i)$ is the next probability of threat-occurrence.

Furthermore, the occurrence probability for specific threat-state can be calculated using the formula (9).

$$P'(S_k) = \sum_{i=1}^{n} P(S_i) P_{ik} \tag{9}$$

where k is the specific threat-state.

Finally, the Expected Frequency (EF) of threat-occurrence is estimated using the probability of threat-occurrence and the median for each threat-state as formula (10).

$$EF = \sum_{i=1}^{n} P(S_i) M(S_i) \tag{10}$$

where n is the number of threat-states, $P(S_i)$ is the probability of threat-occurrence for each threat-state, and $M(S_i)$ is the median of each threat-state.

4.5 A Case Study

A case study presents how to make the damage propagation model. Especially, one case that a threat occurs independently from others is presented. In addition, the result from the model will be compared with the frequency of threat-occurrence. As mentioned previously, in this case study, the statistics for hacking and virus published by the KISA, are used for 54 months, from January 2001 to June 2005, for trust in the past data.

First, threat-occurrence data are gathered and analyzed, and the priority is given to threats. After this step, the frequency and statistics of threat for each month is obtained, as presented in Table 1.

Table 1. Frequency and statistics of threat T_1 for each month

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Total
2001	85	125	70	89	85	64	65	495	268	77	51	97	1571
2002	401	119	82	59	286	417	313	298	210	465	472	990	4112
2003	1148	557	1132	934	306	450	185	544	119	137	129	96	5837
2004	154	148	118	1066	493	181	72	22	16	24	125	90	2509
2005	29	20	15	3	15	36	-	-	-	-	-	-	118

T_1 is one of hacking threats in information system that is an 'Illegal Intrusion using malicious applications such as Netbus and Subseven' as one of the hacking threats in information systems. This threat leaks information and interrupts the normal process in information systems.

First of all, in order to define the threat-state, several thresholds are created, considering the frequency of threat-occurrence of T_1 as follows:

- S_1: 0~300, S_2: 301~600, S_3: 601~900, S_4: 901~1200

As mentioned in Subsection 4.1, the threat-state is defined as the threshold when threats occur independently. Therefore, the threat-sate can be defined as formula (11).

$$S = \{S_1, S_2, S_3, S_4\} \tag{11}$$

Next, in order to calculate the transition matrix of threat-state, the threat-states are listed by mapping the threat-occurrence data into threat-state defined in (11).

$S_1, S_1, S_1, S_1, S_1, S_1, S_1, S_2, S_1, S_1, S_1, S_1, S_2, S_1, S_1, S_1, S_1, S_2, S_2, S_1, S_1, S_2,$
$S_2, S_4, S_4, S_2, S_4, S_4, S_2, S_2, S_1, S_2, S_1, S_1, S_1, S_1, S_1, S_1, S_1, S_4, S_2, S_1, S_1, S_1,$
$S_1, S_1, S_1, S_1, S_1, S_1, S_1, S_1, S_1, S_1$

From the above listing of threat-states, the transition number from a threat-state ($S_1 \sim S_4$) is counted to another, and the transition matrix is made in the following formula (12):

$$\begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \\ S_4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} S_1 & S_2 & S_3 & S_4 \end{array} \\ \begin{bmatrix} 31 & 5 & 0 & 1 \\ 6 & 3 & 0 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 2 \end{bmatrix} \end{array} \qquad \begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \\ S_4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} S_1 & S_2 & S_3 & S_4 \end{array} \\ \begin{bmatrix} 0.84 & 0.13 & 0 & 0.03 \\ 0.55 & 0.27 & 0 & 0.18 \\ 0 & 0 & 0 & 0 \\ 0 & 0.60 & 0 & 0.40 \end{bmatrix} \end{array} \tag{12}$$

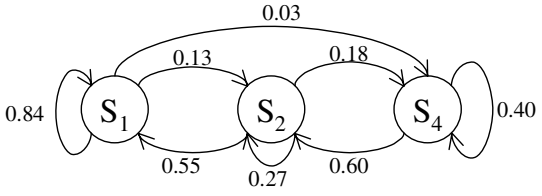


Fig. 3. Threat-State diagram for T_1

As shown in formula (12), the entries of transition matrix satisfy formula (4) that the rows of transition matrix add to one. Furthermore, the transition matrix can be translated as a threat-state diagram shown in Fig.3.

To calculate the initial probability for T_1, the most recent threat-occurrence data, in this case study, are used for six months, from January 2005 to June 2005. The initial probability can be calculated using formulas (5), (6) and (7) as follows:

- Frequency: 29, 20, 15, 3, 15, 36 = $S_1, S_1, S_1, S_1, S_1, S_1$
- Initial Probability: $P(S_1 \quad S_2 \quad S_3 \quad S_4) = P(1 \quad 0 \quad 0 \quad 0)$ (13)

Finally, the probability and frequency of threat-occurrence can be estimated using the transition matrix and the initial probability, that is, formulas (12) and (13). First of all, the probability of threat-occurrence can be calculated by formula (14).

$$(1 \quad 0 \quad 0 \quad 0)\begin{bmatrix} 0.84 & 0.13 & 0 & 0.03 \\ 0.55 & 0.27 & 0 & 0.18 \\ 0 & 0 & 0 & 0 \\ 0 & 0.60 & 0 & 0.40 \end{bmatrix} = (0.84 \quad 0.13 \quad 0 \quad 0.03) \tag{14}$$

From the above result, the probability of threat-occurrence issue to be 0.84 in S_1, 0.13 in S_2 and 0.03 in S_4 in next month. That is, the frequency of threat-occurrence will be from 0 to 300. In order to obtain the detail frequency, the formula (10) is used. First of all, the median $M(S_i)$ for each threat-state defined in (11) is calculated as follows:

- $M(S_1)$: 36, $M(S_2)$: 0, $M(S_3)$:0, $M(S_4)$:0

In this case study, in order to calculate the median, the frequency of threat-occurrence of the previous month is used.

The frequency of threat-occurrence can be calculated using the formula (10), where n is 4 as follows:

$$EF = \sum_{i=1}^{4} P(S_i)M(S_i) = 0.84 \times 36 \cong 30$$

From the result, the frequency of threat-occurrence can be predicted at approximately 30 in the next month. However, there is a result gap between the expected frequency and the real frequency reported by KISA because the real frequency of the next month is 76. In order to obtain a more precise estimation in the proposed model, some requirements must be considered. First, the estimation, which is close to the real

occurrence of a threat, is decided by the subdivision of threshold. That is, the larger the number of thresholds, the more precise data can be obtained. However the complexity of the threat-state and transition matrix increases, depending on the number of thresholds. Therefore, it is necessary to define appropriate thresholds of each threat to define the set of threat-state. Second, the scope of the most recent data to define the Initial Probability should be considered. The scope of the most recent occurrence data also affects the probability and frequency of threat-occurrence. Third, in the proposed model, it is very important to analyze the past data, which are reliable and holds large amounts of data relating to the threats. It is impossible to predict the probability and frequency of threat-occurrence without the reliable data of threats. Therefore, making process of statistics is required. In this paper, although the past data of each month are used, a more precise result can be obtained than if past data are used relative to the date or week.

The proposed model in this paper has different advantages from existing models. First, the proposed model estimates the probability or frequency of threat-occurrence unlike the worm or virus propagation model, which obtains the number of damaged systems, in particular, the number of infected computers in the system. This probabilistic approach can be applied to diverse kinds of information systems, making useful countermeasures. Second, the proposed model can be applied for the diverse threats. Therefore the threats can be analyzed synthetically with an analysis of the relationship among the threats.

5 Related Work

Quite a few damage propagation models, especially about virus and worm, have proposed. Two classical epidemic models were introduced: A simple epidemic model and the Kermack-Mckendrick epidemic model. A simple epidemic model is an epidemic model of an infectious disease in a population without the removing state [7][8][9]. It is assumed that the population consists of two types of individuals: the susceptible individuals (denoted "S") and the infective individuals (denoted "I"). The susceptible ones do not have the disease but could be infected by the disease. The infective ones have the disease and can infect others. Each host stays in one of these two states: susceptible or infective, with a function of time. The Kermack-Mckendrick epidemic model [7][8][10] adds "R" (removed) state into the simple epidemic model. The R state cannot be infected by the disease or infect others with the disease. This is called an SIR model due to the state transition can be S→I→R. In addition to these two epidemic models, there are various propagation models. Although the Kermack-Mckendrick model improved the simple model by considering a factor that some infectious hosts either recover or die after some time, this model is not suitable for modeling worm propagation because it does not consider the human countermeasures and so on. Two-factor worm model considers the effect of human countermeasures and the congestions caused by the worm scan traffic [10][11]. In the Internet, countermeasures such as cleaning, patching, and filtering against worms will both remove susceptible hosts and infectious hosts from circulation in the Kermack-Mckendrick

model. The effect of quarantine on the Internet level to constrain worm propagation was presented in [11][12]. They showed that an infectious host had a number of paths to a target due to the high connectivity of the Internet. Therefore they could prevent the widespread of a worm on the Internet level by analyzing the quarantine on the Internet. A discrete-time worm model, which considers the patching and cleaning erect during worm propagation, was presented in [13][14]. As shown previously, most of damage propagation models focus on virus and worm. Therefore these models cannot be applied to the diverse threats creating in information systems. To overcome this limitation, a new damage propagation model based on Markov process, was proposed in this paper.

6 Conclusion and Future Work

In this paper, a probabilistic model of damage propagation is proposed based on the Markov process. The proposed model can estimate the spread of damage due to not only virus or worm attacks but also diverse threats. Furthermore, a case study is presented using the reliable past data from KISA. However, more research is needed to improve the computation efficiency, such as: 'Which threat should be applied first among the many threats?' and 'How should the appropriate threshold for threat-state be decided?'. Finally, in order to design the Markov-based model, it is required to compile accurate data when threats actually occur.

References

1. Hoh Peter In, Young-Gab Kim, Taek Lee, Chang-Joo Moon, Yoonjung Jung, Injung Kim: A Security Analysis Model for Information Systems. Lecture Notes in Artificial Intelligence, Vol.3398. Springer-Verlag, Berlin Heidelberg(2004) 505-513
2. Gary Stoneburner, Alice Goguen, and Alexis Feringa: Risk Management Guide for Information Technology Systems, NIST Special Publication 800-30, NIST(2002)
3. GAO: Information Security Risk Assetment-Practices of Leading Organizations. GAO/AIMD-00-33(1999)
4. Kishor S. Trivedi: Probability and Statistics with Reliability, Queuing and Computer Science Applications. Second Edition, WILEY Interscience(2002)
5. Roy D. Yates, David J. Goodman: Probability and Stochastic Process. Second Edition, WILEY International Edition(2003)
6. KISA: Statistics and Analysis on Hacking and Virus. http://www.krcert.or.kr
7. J.C. Frauenthal: Mathematical Modeling in Epidemiology, Springer-Verlag, New York(1980)
8. D.J. Deley and J. Gani.: Epidemic Modeling : An Introduction. Cambridge University Press (1999)
9. S. Staniford, V. Paxson, and N.Weaver: How to Own the Internet in Your Spare Time. The proceedings of the 11th USENIX Security Symposium(Security02)(2002)
10. C. C. Zou, W. Gong, and D. Towsley: Worm Propagation Modeling and Analysis under Dynamic Quarantine Defense. ACM CCS Workshop on Rapid Malcode (WORM'03) (2003)

11. C. C. Zou, W. Gong, and D. Towsley: Code Red Worm Propagation Modeling and Analysis. The proceedings of the 9th ACM Conference on Computer and Communications Security(2002) 138-147
12. D.Moore, C. Shannon, G. M. Voelker, and S. Savage: Internet Quarantine: Requirements for Containing Self-Propagating Code, The proceedings of IEEE INFOCOM(2003)
13. Z. Chen, L. Gao, K. Kwiat: Modeling the Spread of Active Worms. The proceedings of IEEE INFOCOM2003(2003)
14. T. Vogt: Simulating and Optimising Worm Propagation Algorithms. http://web.lemuria.org/security/WormPropagation.pdf(2003)

Foundations of Attack Trees

Sjouke Mauw[1] and Martijn Oostdijk[2]

[1] Eindhoven University of Technology
sjouke@win.tue.nl
[2] Radboud University Nijmegen
martijno@cs.ru.nl

Abstract. Attack trees have found their way to practice because they have proved to be an intuitive aid in threat analysis. Despite, or perhaps thanks to, their apparent simplicity, they have not yet been provided with an unambiguous semantics. We argue that such a formal interpretation is indispensable to precisely understand how attack trees can be manipulated during construction and analysis. We provide a denotational semantics, based on a mapping to attack suites, which abstracts from the internal structure of an attack tree, we study transformations between attack trees, and we study the attribution and projection of an attack tree.

Keywords: attack trees, semantics, threat analysis.

1 Introduction

Attack trees (the term is introduced by Schneier in [10, 11]) form a convenient way to systematically categorize the different ways in which a system can be attacked. The graphical, structured tree notation is appealing to practitioners, yet also seems promising for tool builders attempting to partially automate the threat analysis process. As such, attack trees may turn out to be of interest to the security community at large as a standard notation for threat analysis documents.

An attack tree is a tree in which the nodes represent attacks. The root node of the tree is the global goal of an attacker. Children of a node are refinements of this goal, and leafs therefore represent attacks that can no longer be refined. A refinement can be conjunctive (aggregation) or disjunctive (choice). Figure 1 shows an example of an attack tree. In this tree, the goal of the attacker is to obtain a free lunch. The tree lists three possible ways to reach this goal. Lower levels in the tree explain how these sub-goals are refined as well. For instance, the "Eat-n-run" attack requires the attacker to order a meal and to leave the restaurant without paying. The arc connecting these two components expresses that this is a conjunctive refinement, which means that all sub-goals have to be fulfilled. Refinements without such a connecting arc are disjunctive, expressing that satisfying one sub-goal suffices.

Once the possible attacks on a system have been modeled in an attack tree, the tree can be used to analyze attributes of the security of the system. Schneier

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 186–198, 2006.
© Springer-Verlag Berlin Heidelberg 2006

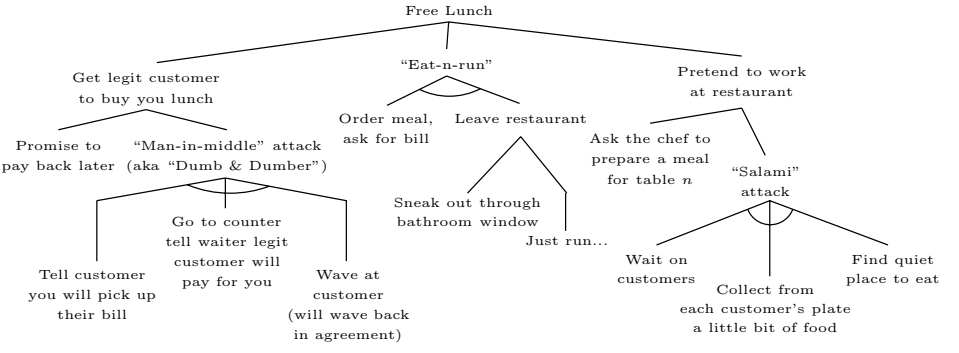


Fig. 1. Example attack tree

suggests several such attributes, for example the (im)possibility, the cost, and whether special tools are needed. The analysis proceeds in two steps: First, the value for each leaf node is determined. Second, the value in non-leaf nodes is synthesized from the value of its children. Thus, creativity on the part of the analyst is only needed in figuring out good values for the leaf nodes, as the rules for synthesis in both the conjunctive and the disjunctive case is usually determined by the nature of the attribute.

The result of an analysis can be the value of an attribute in the root node (for example the cost of the cheapest attack), but it could also be a sub-tree consisting of nodes adhering to some predicate (for example those attacks costing less than 100K Euro or those attacks that do not require use of special equipment). Also, values of different attributes can be defined (for example to determine the cheapest attack not using special equipment).

At a conceptual level attack trees are well understood and the above description is enough to work with them and even develop tool support. However, there are some questions that require a more fundamental answer: What is an attack? Is it just a collection of steps that should be performed or does it have some internal structure? Which conditions should an attribute satisfy before it allows to be synthesized bottom-up? Under what conditions may a projection of a predicate be executed bottom-up? When do two attack trees represent the same set of attacks? How should combined attributes be treated? And which extensions of the formalism (forests, directed acyclic graphs, attack graphs) are possible?

In order to be able to answer these questions, and in order to determine what computer aided threat analysis tools could look like, it is necessary to provide attack trees with foundations. Specifically, this paper provides attack trees with a semantics in terms of attack suites and defines valuations and projections in a formal way. Furthermore we show which algebraic conditions on attributes are sufficient to allow correct inference of values.

Related work. While the foundations of other sub-disciplines of computer security have had plenty of attention from formal methods researchers (e.g. access control [4], cryptographic protocols [7]), the sub-discipline of threat analysis has

had little attention thus far. Attack trees are usually attributed to Schneier. They seem, however, to have a much longer tradition. Witnessed, for instance, by work on fault trees [16, 2]. Other research considers attack graphs [9, 13], in which event sequences are the central topic of research, rather than event abstractions. This seems to be an entirely different field with no cross-references to the kind of structures we study in the current paper. This field has its own set of tools (reachability analysis, etc.) In fact, the risk analysis community seems to have brought forth many such event based formalisms, for example cause-effect diagrams. Although mostly used to describe system-internal events, these can be applied to active external attacker scenarios as well. See for instance [3]. In [12] attack trees are compared to attack nets (a threat analysis formalism based on petri nets, described in [6]) with regard to sharing of security knowledge between collaborators. As far as collaborative attack modeling is concerned, the authors prefer attack nets over attack trees. Much of their criticism on attack trees seems to be based on a lack of semantics of the formalism. Tidwell et al. extend the attack tree formalism with parameters in [15], and successfully apply these trees to model multi-stage Internet attacks. The trees are used inside intrusion detection systems. A commercial tool [5] and some rudimentary tools [1, 8] are already available for Schneier's attack trees.

Outline. This paper is structured as follows. In Section 2 we introduce the notions of attack suites and attack trees and define a mapping from attack trees to attack suites, expressing the semantics of an attack tree. Section 3 provides an alternative characterization of the semantics through rewriting. This makes it possible to transform equivalent attack trees into each other. In Section 4 we define attributes on attack trees and discuss under which conditions they can be synthesized bottom-up. Finally, we consider how attacks can be singled out that satisfy some given property. Such projections are discussed in Section 5.

2 Attack Suites and Attack Trees

The purpose of an attack tree is to define and analyze possible attacks on a system in a structured way. This structure is expressed in the node hierarchy, allowing one to decompose an abstract attack or attack goal into a number of more concrete attacks or sub-goals. Although this structure carries information on the interpretation and grouping of attacks, we will discard it when determining the meaning of an attack tree. An attack tree simply defines a collection of possible attacks which we call an *attack suite*. Each attack consists of the components required to perform this attack. A component may occur more than once in an attack, so an attack is a multi-set of attack components. These attack components are at the lowest level of abstraction that we consider and thus have no internal structure. By describing an attack as a set of attack components we will also abstract from any causal relations between the components, such as being ordered in time.

First we introduce some common notation. We use $\mathcal{P}(V)$ to denote the power set of a set V and $\mathcal{P}^+(V)$ to denote the set of all non-empty subsets of V.

Likewise, we use $\mathcal{M}(V)$ and $\mathcal{M}^+(V)$ for multi-sets. The multi-set consisting of elements a, a and b is denoted by $\{\!\{a, a, b\}\!\}$. The difference of (multi-)sets V and W is denoted by $V \setminus W$. The substitution of element x for y in (multi-)set V is denoted by $V[x/y]$. The *distributed product* of two sets of multi-sets V and W is the set defined by $V \otimes W = \{v \uplus w \mid v \in V, w \in W\}$, where $\uplus$ denotes multi-set union. The operator $\bigotimes_{i \in I}$ is the generalization of $\otimes$ (with unit element $\{\emptyset\}$). The set of end nodes of a tree T is denoted by $E(T)$.

Definition 1. *Let $\mathbb{C}$ denote a set of attack components. An* attack *is a finite non-empty multi-set of $\mathbb{C}$ and an* attack suite *is a finite set of attacks. The universe of attacks is denoted by $\mathbb{A} = \mathcal{M}^+(\mathbb{C})$ and the universe of attack suites is denoted by $\mathbb{S} = \mathcal{P}(\mathbb{A})$.*

Example 1. If we have $\mathbb{C} = \{$*open door*, *steal key*, *force lock*, *pick lock*$\}$ then the following attack suite defines three ways to illegally enter a building $\{\{\!\{$*steal key*, *open door*$\}\!\}, \{\!\{$*force lock*, *open door*$\}\!\}, \{\!\{$*pick lock*, *open door*$\}\!\}\}$.

Attack trees as defined by Schneier have two types of nodes: *and-nodes* and *or-nodes*. The children of an and-node should all be executed to reach the goal represented by the and-node, while execution of any child of an or-node suffices to reach the goal of the or-node. By considering only one type of nodes, we will follow a slightly different, but equivalent, approach. Rather than considering edges from a node to its children, we consider connections from a node to a multi-set of nodes. Such a connection is called a *bundle*. A node may contain several such bundles. The nodes in a bundle must all be executed to form an attack. Execution of any bundle of a node will suffice to reach the goal of that node. Our approach differs in one more aspect from Schneier's attack trees. We allow sharing of nodes as a means to express that a sub-attack occurs more than once. Although a node may be contained in several bundles, we will not allow the construction of cycles. So, formally speaking, we study *rooted directed acyclic bundle graphs*, but we will still call them *attack trees*.

Definition 2. *An* attack tree *is a 3-tuple $(N, \rightarrow, n_0)$, where N is a finite set of nodes, $\rightarrow$ is a finite acyclic relation of type $\rightarrow \subseteq N \times \mathcal{M}^+(N)$ and $n_0 \in N$ is the root node, such that every node in N is reachable from n_0. The universe of attack trees is denoted by $\mathbb{T}$.*

We will use the informal terminology "A is a bundle of m" to express that $m \rightarrow A$. Whenever we speak of attack suites in the context of some attack tree T, we will identify the universe of attack components $\mathbb{C}$ with T's end nodes, $E(T)$.

Next, we define the semantics of attack trees by interpreting them in the domain of attack suites. As stated before, the internal branching structure of an attack tree will not be expressed in the attack suite. The semantics only expresses which combinations of attack components (i.e. end nodes of the tree) form an attack. The attack suite defined by a node in the tree can be determined recursively from its bundles. Since the bundles define alternative attacks, the

attack suite defined by a node consists of the union of the attack suites defined by its bundles. The attack suite defined by a single bundle is determined by the attack suites of the nodes contained in the bundle. In order to construct an attack in a bundle, we must take an attack from each of the nodes in the bundle and join these together. The definition below formalizes this recursive construction. We use the distributed product, as defined above, to join the attacks within a bundle and we use the normal set union to join the bundles. The base case of the recursive definition considers the end nodes of the tree. They define an attack suite consisting of one attack which contains a single attack component.

Definition 3. *Let T be an attack tree $(N, \rightarrow, n_0)$, then the semantics of a node $[\![\_]\!] : N \to \mathbb{S}$ is defined recursively by*

$$[\![n]\!] = \begin{cases} \{\{n\}\} & \text{if } n \in E(T) \\ \bigcup\limits_{n \rightarrow A} \bigotimes\limits_{m \in A} [\![m]\!] & \text{if } n \notin E(T) \end{cases}$$

The semantics of an attack tree is defined as the semantics of its root node, $[\![T]\!] = [\![n_0]\!]$.

3 Transformations

Figure 2 illustrates that two structurally different attack trees may intuitively capture the same information. The difference in structuring can arise from a different approach towards partitioning the attacks. One may also want to simplify or rebalance an attack tree in order to change the view on the described attack suite, without changing its meaning. These two observations lead to the definition of semantics-preserving transformations of attack trees. The class of allowed transformations can be characterized by just two reduction rules. These rules are illustrated in Figure 3 and formalized in Definition 4. In the figure we consider a node which has a bundle A and possibly some other bundles (illustrated by W). Bundle A contains node m, which has a bundle B. We make a distinction between the two cases that B is the only bundle of m and that m has more bundles.

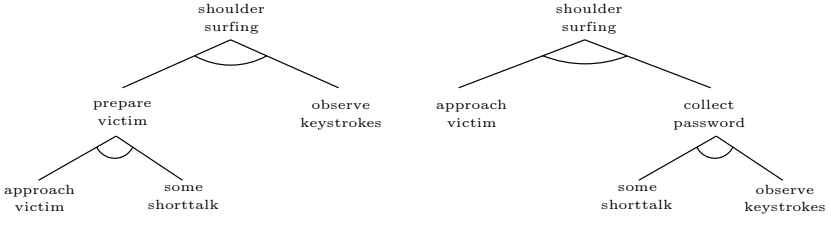


Fig. 2. Two equivalent trees

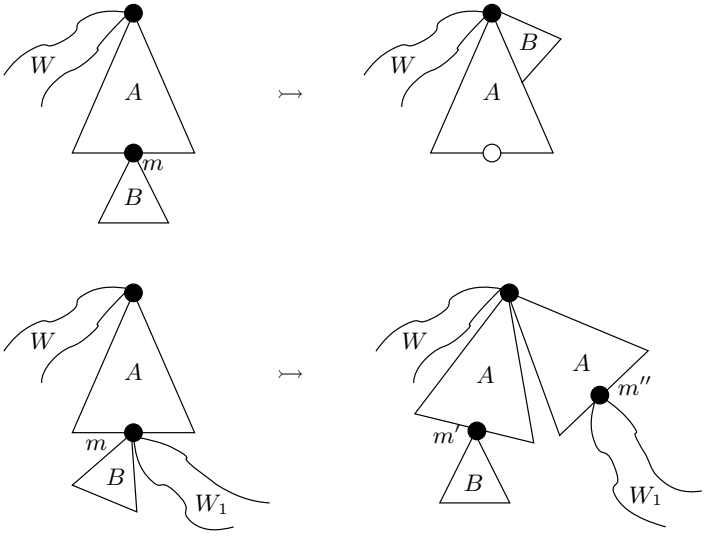


Fig. 3. Two reduction rules for attack trees (cf. Definition 4)

The first rule is based on the associativity of conjunction. If a bundle contains a node with only one sub-bundle, then this node can be deleted and its sub-bundle can be lifted one level, so as to become part of the bundle. Intuitively, this captures the fact that if we want to perform an attack that contains one sub-attack, we can simply take the components of the sub-attack and add them to the attack.

The second rule is based on the distributivity of conjunction over disjunction. If a bundle contains a node with two (or more) sub-bundles, then we can replace the bundle by two copies with the difference that the first copy only contains the first sub-bundle and the second copy only contains the second sub-bundle. Intuitively, this captures the fact that if we want to perform an attack that contains a sub-attack which can be performed in two ways, we have actually described two attacks. Please notice that the duplication of attack A in the figure is displayed in a somewhat misleading way. The picture suggests that the nodes in bundle A are also copied, but it is our intention that common nodes are shared, without duplicating nodes.

The next definition of an Abstract Reduction System on attack trees formally captures this intuition. We use the function reachable to remove all nodes and bundles that through rewriting become unreachable from the root node. The definition of this function is straightforward and will be omitted.

Definition 4. *The reduction relation* $\rightarrowtail: \mathbb{T} \to \mathbb{T}$ *is defined by the following two reduction rules. Let* T *be an attack tree* $(N, \to, n_0)$*,* $n, m \in N$ *and* $A, B \in \mathcal{M}^+(N)$ *such that* $n \to A$*,* $m \in A$*,* $m \to B$*.*

1. *if* B *is the only bundle in* m *(i.e.* $\forall_{C: m \to C} \cdot C = B$*), then* $(N, \to, n) \rightarrowtail \text{reachable}(N, \to', n)$*, where*

$$\rightarrow' = (\rightarrow \setminus \{(n, A)\}) \cup \big\{(n, (A \setminus \{\!|m|\!\}) \uplus B)\big\}$$

2. *if B is not the only bundle in m (i.e. $\exists_{C:m \to C} \cdot B \neq C$), then* $(N, \rightarrow, n) \rightarrowtail \mathrm{reachable}(N', \rightarrow', n)$, *where*

$$\begin{aligned} \rightarrow' = \big(&\rightarrow \setminus \{(n, A)\}\big) \\ &\cup \big\{(n, A[m'/m]), (n, A[m''/m]), (m', B)\big\} \\ &\cup \{(m'', B') \mid m \to B', B \neq B'\}, \\ N' = N \cup \{m', m''\} \quad &\textit{where } m', m'' \notin N \end{aligned}$$

Because we are interested in transformations (without a preferred direction), we introduce the reflexive, transitive, symmetric closure of $\rightarrowtail$, denoted by $\equiv$.

Later we will prove that these reduction rules are *sound* with respect to the semantics. Interestingly enough, the rules turn out to be complete as well. Therefore, the reduction rules defined above are not only useful for manipulating attack trees; they also provide an alternative characterization of the semantics of an attack tree. It can be easily seen that the normal forms are the "one-level attack trees" which are in one-to-one correspondence to the universe of attack suites. In order to obtain these results, we first we have to show that the reduction rules are well-behaved, i.e. that there are no infinite reduction sequences and, roughly speaking, that any two diverging reductions can be reduced to a common attack tree.

Lemma 1. *The reduction relation on attack trees is strongly terminating.*

Proof. The proof is not given here because of space restrictions.

Lemma 2. *The reduction relation on attack trees is weakly confluent.*

Proof. This follows by observing that all critical pairs are convergent. (The rules are not overlapping anyway.)

As a standard corollary of the above lemmas we have the unique normal forms property (see e.g. [14]). The set of normal forms can be determined easily.

Lemma 3. *If we denote the set of attack trees with depth $\leq d$ by $\mathbb{T}_{\leq d}$, then the set of normal forms of $\rightarrowtail$ is $\mathbb{T}_{\leq 1}$.*

Proof. This follows easily by inspection of the left-hand sides of the reduction rules. An attack tree with depth ≥ 2 still has a redex and, conversely, to have a redex an attack tree must have depth ≥ 2.

Example 2. The tree presented in Figure 1 has a normal form consisting of six bundles, each of depth one, corresponding to the following attack suite. (The lengthy attack component descriptions of Figure 1 have been replaced by mnemonic names.)

$$\begin{aligned} \{\{&\textit{promise pay later}\}, \{\textit{tell victim}, \textit{go counter}, \textit{wave}\}, \{\textit{order}, \textit{bathroom}\}, \\ &\{\textit{order}, \textit{just run}\}, \{\textit{ask chef}\}, \{\textit{wait}, \textit{collect little}, \textit{quiet place}\}\} \end{aligned}$$

By verifying that the reduction rules preserve the semantics of an attack tree, we obtain their soundness. However, we will postpone the soundness proof until Section 4, where we will prove a more general soundness property of which this is an instantiation.

Theorem 1 (Soundness). *If* $T_1 \equiv T_2$*, then* $[\![T_1]\!] = [\![T_2]\!]$.

Proof. Postponed until Corollary 1.

The following lemma helps in proving completeness. It establishes the one-to-one correspondence between normal forms and attack suites.

Lemma 4. *Let* $T_1, T_2 \in \mathbb{T}_{\leq 1}$ *such that* $[\![T_1]\!] = [\![T_2]\!]$ *then* $T_1 = T_2$.

Proof. It follows easily from the definitions that an attack suite has a unique representation in $\mathbb{T}_{\leq 1}$. Every attack in the attack suite gives rise to a bundle from the root node.

Theorem 2 (Completeness). *If* $[\![T_1]\!] = [\![T_2]\!]$*, then* $T_1 \equiv T_2$.

Proof. If we have $[\![T_1]\!] = [\![T_2]\!]$, then we can reduce T_1 and T_2 to normal form, denoted by $\mathrm{nf}(T_1)$ and $\mathrm{nf}(T_2)$, and obtain

$$[\![\mathrm{nf}(T_1)]\!] = [\![T_1]\!] = [\![T_2]\!] = [\![\mathrm{nf}(T_2)]\!]$$

Now, Lemma 4 yields equality of the normal forms: $\mathrm{nf}(T_1) = \mathrm{nf}(T_2)$. So, $T_1 \equiv T_2$.

4 Attributes

In order to calculate e.g. the cost or impact of an attack, an attack tree can be decorated with attributes. The attribute value of an attack tree can be calculated by first determining the semantics of the tree followed by calculating the attribute value of the defined attack suite. However, under some conditions it is possible to synthesize the attribute value of the attack tree without first having to reduce the tree to normal form. This can be done by calculating the attribute values of the nodes in a bottom-up way.

Before defining attributes in an attack tree, we will first define the attribution of attack suites.

Given a set V of attribute values, an attribute α is a function that assigns a value to every attack component, $\alpha\colon \mathbb{C} \to V$. An attack consists of a number of attack components that must all be executed, so we assume that the attribution of an attack can be calculated from the attributions of its attack components. Therefore, we extend α to attacks, $\alpha\colon \mathcal{M}^+(\mathbb{C}) \to V$. In the same way, because an attack suite consists of a number of attacks, we extend attributions to attack suites, $\alpha : \mathcal{P}(\mathcal{M}^+(\mathbb{C})) \to V$. In order to calculate the attribution of an attack from the attributions of its attack components we use a *conjunctive combinator* $\triangle$, while for determining the value of an attack suite from its attacks we use a *disjunctive combinator* $\triangledown$. We require that the combinators are associative and commutative and that the conjunctive combinator distributes over the disjunctive combinator. These properties follow from the structure of attack trees.

Definition 5. *Let $\mathbb{C}$ be a set of attack components. An attribute domain is a structure $(V, \triangledown, \triangle)$ where V is the set of attribute values, $\triangledown : V \times V \to V$ is the disjunctive combinator for attribute values and $\triangle : V \times V \to V$ is the conjunctive combinator for attribute values. We require that the combinators are associative and commutative:*

$$\begin{aligned}(x \triangledown y) \triangledown z &= x \triangledown (y \triangledown z)\\ x \triangledown y &= y \triangledown x\\ (x \triangle y) \triangle z &= x \triangle (y \triangle z)\\ x \triangle y &= y \triangle x\end{aligned}$$

Given an attribute domain $(V, \triangledown, \triangle)$, an attribute α is a function from $\mathbb{C}$ to V. An attribute domain is distributive if the following property holds:

$$x \triangle (y \triangledown z) = (x \triangle y) \triangledown (x \triangle z)$$

Generalization of the combinators to $\mathcal{M}^+(V) \to V$ is defined in the usual way. An attribute domain is often called a semi-ring. However in published literature, the notion of a semi-ring often occurs with the additional requirement of a unit element for one or both operators. In the setting of attack trees this is not required, since we assumed that bundles are not empty.

Definition 6. *Let S be an attack suite and α an attribute with attribute domain $(V, \triangledown, \triangle)$ then the value attributed by α to S is*

$$\alpha(S) = \bigtriangledown_{A \in S} \bigtriangleup_{c \in A} \alpha(c)$$

Example 3. The structure $(\mathbb{N}, \min, +)$ is an example of a distributive attribute domain. An attribute with this attribute domain could be interpreted as "cost of the cheapest attack". Other examples: $(\mathbb{N}, \max, +)$ "maximal damage", $(\mathbb{N}, \min, \max)$ "minimum skill level required to perform attack", $(\mathbb{B}, \wedge, \vee)$ "is the attack possible", $(\mathbb{B}, \vee, \wedge)$ "special equipment needed".

It is interesting to see that there are also examples of attributes that do not satisfy the requirements. Consider, for instance, the structure $(\mathbb{N}, +, \min)$. This could express the costs to defend against all attacks from an attack suite: to defend against one attack one only has to find the cheapest defense against any of its attack components, and to defend against an attack suite, one has to add the defense costs to all its attacks. However, this structure is not a distributive attribute domain because it does not satisfy the required distribution property. At the end of this section we will come back to this counter example.

Now that we have defined the calculation of an attribute for attack suites, we will define the attribution of an attack tree.

Definition 7. *Let T be an attack tree $(N, \rightarrow, n_0)$ and let $\alpha : E(T) \to V$ be an attribute with a distributive attribute domain. Then we define the extension of α to the nodes of the attack tree, $\alpha : N \to V$, inductively by*

$$\alpha(n) = \bigvee_{n \to A} \bigwedge_{m \in A} \alpha(m) \quad \text{for } n \notin E(T).$$

The value of an attack tree is determined by the value of its root node: $\alpha(T) = \alpha(n_0)$.

This clearly defines a unique attribution of the attack tree. Moreover, the value attributed to an attack tree is respected by the rewrite rules.

Theorem 3. *Let* T_1 *and* T_2 *be attack trees and let* $\alpha\colon E(T) \to V$ *be an attribute with distributive attribute domain, then* $T_1 \equiv T_2$ *implies that* $\alpha(T_1) = \alpha(T_2)$.

Proof. The proof is not given here because of space restrictions.

It is notable that Definition 7 looks similar to Definition 3. In fact semantics can be seen as an attribute, since $(\mathcal{P}(\mathcal{M}^+(\mathbb{C})), \cup, \otimes)$ is a distributive attribute domain. Associativity and commutativity are standard, while $x \otimes (y \cup z) = (x \otimes y) \cup (x \otimes z)$ can be verified easily. Thus, the postponed soundness proof of Theorem 1 is a corollary of Theorem 3.

Corollary 1. *The reduction rules are sound with respect to the semantics, i.e. if* $T_1 \equiv T_2$, *then* $[\![T_1]\!] = [\![T_2]\!]$.

Finally, we observe that the value of a tree is equal to the value of the attack suite that is formed by taking the semantics of the tree.

Corollary 2. *Given attribute* α *and attack tree* T, *we have* $\alpha(T) = \alpha([\![T]\!])$.

Proof. From Theorem 3 it follows that $\alpha(T) = \alpha(\mathrm{nf}(T))$. The required equality now follows from the observed correspondence between normal forms and attack suites (Lemma 4), and by comparing Definitions 6 and 7.

Clearly, all reasonable attributes encountered in practice satisfy the requirements. However, as shown in Example 3, there are also attributes that make sense at first sight, but which are not consistent with the semantics. The inconsistency shows when comparing the value of the attribute calculated on the original tree to the value of the attribute calculated on the normal form of this tree. These values can differ if e.g. the law of distributivity does not hold. Thus we can conclude that there are attributes which cannot be synthesized bottom up. The counter example shows the usefulness of our formalization. We are now able to make the distinction between *sound* attributes and attributes that are inconsistent with the algorithms intuitively sketched by Schneier (which form the basis of our semantics).

5 Projections

By manipulating attack trees one can get answers to questions like "Show all attacks that do not require special equipment", or "Which attacks incur a damage over 1000 dollars"? The last question e.g. requires an attribute *incurred damage* and a predicate on its domain, $P(n) \equiv n \geq 1000$. Taking the projection of an attack suite boils down to selecting the attacks that satisfy the predicate.

Definition 8. *Let α be an attribute with distributive attribute domain $(V, \triangledown, \triangle)$ and let $P \subseteq V$ be a predicate. Then the projection of attack suite $S \in \mathbb{S}$ onto P is defined by $\Pi^{\alpha}_{P}(S) = \{A \in S \mid P(\alpha(A))\}$.*

The definition of projections on attack trees follows from the algorithm sketched by Schneier: remove all attacks that do not satisfy the predicate.

Definition 9. *Let α be an attribute with distributive attribute domain $(V, \triangledown, \triangle)$ and let $P \subseteq V$ be a predicate. Then the projection of attack tree $T = (N, \rightarrow, n_0)$ onto P is defined by $\Pi^{\alpha}_{P}(T) = \text{reachable}((N', \rightarrow', n_0))$, where $N' = \{n \in N \mid P(\alpha(n))\} \cup \{n_0\}$ and $\rightarrow' = \{(n, A) \in \rightarrow \mid P(\alpha(A))\}$.*

Again we want to know under which conditions these two definitions are consistent with each other. More precisely, we want to know if the projection of the semantics of an attack tree is equal to the semantics of the projection of that tree. Phrased in terms of the rewriting rules, we want to know if rewriting and projection commute.

A simple example shows that this does not hold in general. Consider the attribute domain $(\mathbb{N}, \min, +)$ to calculate the cost of an attack and look at the first tree of Figure 2. Suppose that all leafs have cost 5, which implies cost 10 for the intermediate node and cost 15 for the root. Now, if we take predicate $P(n) \equiv n \neq 10$, then for the projection of this tree we have to remove the intermediate node (and its dangling leaf nodes) which gives a tree with a single attack component. However, if we first reduce the tree to normal form, the projection would not affect the tree. Clearly, this cannot be a reduct of the first projected tree.

Monotonicity of predicate P suffices to prevent such problems. We say that predicate $P \subseteq V$ is monotonic with respect to attribute domain $(V, \triangledown, \triangle)$ iff

$$P(x \triangle y) \Rightarrow P(x) \wedge P(y)$$
$$P(x \triangledown y) \Rightarrow P(x) \vee P(y)$$

We first prove an auxiliary lemma, where we denote a sequence of zero or more reductions by $\rightarrowtail^*$. After that, we state the main result for projections.

Lemma 5. *Let α be an attribute with distributive attribute domain $(V, \triangledown, \triangle)$ and let $P \subseteq V$ be a monotonic predicate. If T and T' are attack trees such that $T \rightarrowtail T'$, then $\Pi^{\alpha}_{P}(T) \rightarrowtail^* \Pi^{\alpha}_{P}(T')$.*

Proof. The proof proceeds by inspecting the two rewrite rules, while looking at the possible values of P for the nodes of interest. If P is false in the top node, then the sub-trees are all removed and the lemma trivially holds. If P is true in the top node of the first redex, then monotonicity with respect to conjunction implies that the predicate is also true in all sub-nodes, making the same reduction step possible on the projected trees. If P is true in the top node of the second redex, then monotonicity with respect to conjunction and disjunction yields the same reasoning as for the first redex.

Theorem 4. *Let α be an attribute with distributive attribute domain $(V, \triangledown, \triangle)$ and let $P \subseteq V$ be a monotonic predicate. Then we have for $T \in \mathbb{T}$ that $[\![\Pi^\alpha_P(T)]\!] = \Pi^\alpha_P([\![T]\!])$.*

Proof. Due to the correspondence between attack suites and normal forms, it suffices to prove that $\mathrm{nf}(\Pi^\alpha_P(T)) = \Pi^\alpha_P(\mathrm{nf}(T))$. Now, suppose that T reduces to $\mathrm{nf}(T)$ via the sequence $T \rightarrowtail T' \rightarrowtail \ldots \rightarrowtail \mathrm{nf}(T)$. Then by applying Lemma 5 we obtain the reduction sequence $\Pi^\alpha_P(T) \rightarrowtail^* \Pi^\alpha_P(T') \rightarrowtail^* \ldots \rightarrowtail^* \Pi^\alpha_P(\mathrm{nf}(T))$. From the fact that $\mathrm{nf}(T)$ is a normal form, we can derive that $\Pi^\alpha_P(\mathrm{nf}(T))$ is a normal form as well. Because normal forms are unique, we have the desired property $\mathrm{nf}(\Pi^\alpha_P(T)) = \Pi^\alpha_P(\mathrm{nf}(T))$.

Summarizing, we have found that the projection algorithm informally presented by Schneier can only be applied to a restricted class of predicates. Monotonicity of predicate P suffices, but it is easy to see that monotonicity of $\neg P$ is also sufficient.

6 Conclusions

The main result of our work is a formalization of the concepts informally introduced by Schneier. This formalization clarifies which manipulations of attack trees are allowed under which conditions. Such an understanding is a prerequisite for building adequate tool support. A simple experiment with building a prototype tool confirmed the feasibility of our approach.

Central to our work is the observation that an attack tree describes an attack suite. We argue that the structural information that we lose in this way is a residual of the modeling strategy, rather than an intrinsic property of the described set of attacks. Therefore, attack suites form the appropriate level of abstraction. This semantics can be characterized in two ways: by traversing the tree from the leaves to the root and by rewriting the tree to normal form. Both strategies can be easily implemented, but in practice it is more interesting to build and manipulate attack trees than to calculate their semantics. Rewriting is more suited for this purpose. The rewrite rules can be used e.g. to add structure to an unstructured attack suite or to rebalance an attack tree.

It turns out that in order to recursively calculate attributes and projections, as introduced by Schneier, certain conditions have to be met. The condition for attributes (the combination operators form a semi-ring) is rather natural and no serious restriction. The condition for sound projections is somewhat stronger, but still satisfied by Schneier's examples. As mentioned before, our formalization motivates why certain attributes and predicates are not compatible with the informal algorithms presented by Schneier.

Having formalized the basic concepts of attack trees, it is of interest to study the extension with e.g. cycles, ordered attacks and pre- and post-conditions. Furthermore, our experience with using attack trees in practice indicated the need for *defense trees* and *attack forests* (i.e. attack libraries). Finally, we mention that although we have represented an attack as a multi-set, we could have used

normal sets as well. The main difference would be the following extra requirement for attributes: $x \vartriangle x = x$.

Acknowledgments. We thank Alexander Opel, Leon Schrijvers and Martijn Wolfs for performing some initial studies with respect to implementation, use and analysis of attack trees. Hans Zantema is acknowledged for suggesting the norm used in the termination proof.

References

1. TANAT – Threat ANd Attack Tree Modeling plus Simulation, 2004. `http://www13.informatik.tu-muenchen.de:8080/tanat/`.
2. ARES corporation. Risk techniques, fault trees. Technical report. Available from `http://www.arescorporation.com/`.
3. Stefán Einarsson and Marvin Rausand. An approach to vulnerability analysis of complex industrial systems. *Risk Analysis*, 18(5):535 – 545, 1998.
4. Carl E. Landwehr. Formal models for computer security. *Computing Surveys*, 13(3):247 – 277, September 1981.
5. Amaneza Technologies Limited. A quick tour of attack tree based risk analysis using SecurITree. Technical report, 2002.
6. J.P. McDermott. Attack net penetration testing. In *Proc. 2000 workshop on New Security Paradigm*, pages 15–20. ACM, 2001.
7. Catherine Meadows. Open issues in formal methods for cryptographic protocol analysis. In *DARPA Information Survivability Conference & Exposition*, volume 1, pages 237 – 250, 2000.
8. Alexander Opel. Design and implementation of a support tool for attack trees, 2005.
9. Cynthia Phillips and Laura Painton Swiler. A graph-based system for network-vulnerability analysis. In *Proc. New Security Paradigms Workshop*, pages 71–79, 1998.
10. Bruce Schneier. Attack trees: Modeling security threats. *Dr. Dobb's journal*, December 1999.
11. Bruce Schneier. *Secrets & Lies: Digital Security in a Networked World*. Wiley, 2000.
12. Jan Stefan and Markus Schumacher. Collaborative attack modeling. In *Proc. SAC 2002*, pages 253–259. ACM, 2002.
13. L.P. Swiler, C. Philips, D. Ellis, and S. Chakarian. Computer-attack graph generation tool. In *Proc. DARPA Information Survivability Conference and Exposition*, volume 2, pages 307–321, June 2001.
14. Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.
15. T. Tidwell, R. Larson, K. Fitch, and J. Hale. Modeling internet attacks. In *Proc. of the 2001 IEEE Workshop on Information Assurance and Security*, 2001.
16. W.E. Vesely et al. Fault tree handbook. Technical Report NUREG-0492, U.S. Nuclear Regulatory Commission, January 1981.

An Algebraic Masking Method to Protect AES Against Power Attacks*

Nicolas T. Courtois[1] and Louis Goubin[1,2]

[1] Axalto Crypto Research & Advanced Security, 36-38 rue de la Princesse, BP 45, F-78430 Louveciennes Cedex, France
courtois@minrank.org
[2] PRiSM Laboratory, Versailles University, 45 avenue des Etats-Unis, F-78035 Versailles Cedex, France
Louis.Goubin@prism.uvsq.fr

Abstract. The central question in constructing a secure and efficient masking method for AES is to address the interaction between additive masking and the inverse S-box of Rijndael. All recently proposed methods to protect AES against power attacks try to avoid this problem and work by decomposing the inverse in terms of simpler operations that are more easily protected against DPA by generic methods.

In this paper, for the first time, we look at the problem in the face, and show that this interaction is not as intricate as it seems. In fact, any operation, even complex, can be directly protected against DPA of any given order, if it can be embedded in a group that has a compact representation. We show that a secure computation of a whole masked inverse can be done directly in this way, using the group of homographic transformations over the projective space (but not exactly, with some non-trivial technicalities).

Keywords: Rijndael, AES, inverse S-box, homographic transformations, linear fractional transformations, Möbius transformations, the zero-masking problem, Differential Power analysis, higher-order DPA.

1 Introduction

A secure implementation of (even a very secure) cryptographic algorithm, is by no means easy to achieve in portable cryptographic devices such as a smart cards. Indeed, it is hard to protect a secret that is entirely in the hands of a potential attacker. The nature of cryptography makes that all kinds of additional information, even very remotely correlated with the secret quantities manipulated in the cryptographic algorithm, are very likely to either directly leak information on the secret quantities, or indirectly, will help to improve some cryptographic attack. Moreover, in cryptographic devices such as smart cards, the performance

* This work was partially supported by the French Ministry of Research RNRT X-CRYPT project and by the European Commission via ECRYPT network of excellence IST-2002-507932.

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 199–209, 2006.
© Springer-Verlag Berlin Heidelberg 2006

and cost considerations make that there is little or no margin between the required level of security and the best known attack. Thus all kind of side channels are potentially fatal to the security.

The idea of side-channel attacks is very old and well known, for example it is possible to break many implementations of one-time pad by studying the output not in terms of zeros and ones, but with an oscilloscope... In 1998 practical side channel attacks for smart cards have been demonstrated by Kocher, Jaffe and Jun. They introduce a class of attacks called Power Analysis, using the power consumption traces to recover the key of an encryption scheme implemented in a smart card. The Power Analysis of first and higher order is already described by the authors in the original paper [23], and also in [24]. These attacks have been later applied to attack (and protect against such attacks) many secret key schemes, see for example [23, 29] and in particular to AES in [10, 5, 16, 1, 20, 35]. Before 2000, only first order attacks were demonstrated in practice.

In order to protect the secret key schemes against first order power attacks, two generic methods have been proposed: the transformed masking method [27, 1] and the duplication method [21, 22, 10, 11]. Moreover, in the particular case of AES, specific methods have been proposed [1, 39, 20, 6, 38, 36, 37, 31, 32]. In particular, with the progressive replacement of DES by AES as a universal encryption standard, and with slow advances in hardware protections, efficient software protections of AES against higher order side-channel attacks have become a hot topic.

In this paper we propose a new algebraic AES-specific masking method that may appear complex, yet it allows to achieve the same (very ambitious) goal without tables, and thus is suitable for both hardware and software implementations. The main contribution of this paper can be summarised as follows: a fully masked non-linear S-box of Rijndael can still be "embedded" (in a special way) in an algebraic group that has a compact representation. This allows to construct efficient and somewhat mathematically elegant protections against power analysis.

2 AES and Side-Channel Attacks

2.1 Known Side-Channel Attacks

In this paper we limit to passive non-intrusive attacks against implementations of cryptographic algorithms, such as DPA, or any other passive side-channel attack.

The Differential Power Analysis (first order) attack works as follows: one records the power consumption of several runs of a cryptographic algorithm implemented on a smart card. Then one guesses a few bits of the secret or derived key, which allows to compute for each curve, some intermediate bit(s) that appear inside the cryptographic algorithm. For example the first bit of the entry of the first S-box. Then one separates the curves in two classes, these for which this bit is expected to be 0, and those for which this bit is expected to be 1. If the guess on the key is correct, one will be able to distinguish the two classes

by various statistical techniques, if the guess was incorrect, both classes should look the same. An algorithm is susceptible to be attacked by DPA if there is an intermediate value that depends on the plaintext and on key bits or derived bits.

Higher Order DPA (of order k) just generalizes the above attack: we will use up to k points on the curves, or equivalently, up to k intermediate variables. The information obtained on each of these values (e.g. it's power trace) should be efficiently combined in order to produce an output, that is correlated to some internal value/combination of values that again depends on the plaintext and on key or derived bits.

2.2 Adversarial Model for General Side-Channel Attacks

In order to prevent all possible side-channel attacks, whatever is the leakage model, it is possible to assume that the (passive) adversary has full access to some of the intermediate results of the computation. This cannot hold for all values, and Blömer, Merchan and Krummel do define on Fig. 1. in [6] a minimal set of two assumptions that are judged necessary to be able to protect the implementation of a cryptographic algorithm against side-channel attacks. We adapt and comment on these assumptions.

1. First of all, a random number generator must be available (that unlike what we read in [6] does not have to be really random and can be pseudo-random as long as it is not resettable, i.e. at least some real entropy is gathered and used by the device). This random generator must be protected against manipulation/perturbation (not necessarily against reading).
2. The secret key (and part of it) K can be securely combined by a group operation with this random number R. This operation that manipulates K should be secure both for reading and manipulation (!). Then both R and $R \oplus K$ can go further unprotected.

   We observe that this assumption of [6] is not sufficient to protect against DPA of higher order. Our corrected assumption is as follows. One can chose random $R_1, \ldots, R_k$, then securely compute $K' = K \oplus R_1 \oplus \ldots \oplus R_k$ and then the values $K', R_1, \ldots, R_k$ can go further unprotected.

These assumptions seem to be acceptable and realistic, because:

0. smart cards may indeed protect some basic operations very carefully in hardware at the gate level,
1. it is difficult to produce really precise perturbations to random numbers that will not be removed by more randomisation,
2. we assumed that we manipulate the key only once with a random mask, and in a way that is independent of the plaintext. This makes DPA impossible and SPA can again probably be prevented by classical noise/randomisation techniques (without DPA the noise should cover the signal to recover),
3. finally even if the attacker were able to read R (which is permitted by the model), we can have again DPA attacks on $R \oplus K$, the only place the key is exposed. This is linear and as such, it is known to be fundamentally much more resistant to DPA than non-linear operations, see [34].

We will assume that the power of the attacker is restricted to observing at most k intermediate values that appear during the computation. Usually and in [6] it is also implicitly assumed that some "atomic" computations are secure and the only "observable" data are intermediate data at some level of abstraction. This assumption seems strong but we claim that as far as protecting against power attacks of any given order can be achieved, it is not necessary and can be relaxed. If we assume that the code of the algorithm is public, each part of computation is some deterministic function of a small number of intermediate values, and will be secure if k is high enough.

2.3 A Representation of Rijndael

In this paper we will view AES (and other versions of Rijndael) in an abstract way, as a functional composition of three kind of operations denoted by X, I and L:

- X In Rijndael we simply XOR a byte of an expanded key with a byte of the current state. We assume that the expanded key is also computed by a composition of basic operations X,I and L.
- I Special substitution S-box that is called Inv. Many authors identify this function with the inverse $X \mapsto 1/X$ in the finite field. We will see that Inv is not exactly the same thing as $1/X$ in $GF(256)$.
- L Linear operations (and more precisely linear or affine transformations over $GF(2)$, that are fixed and do not depend on the key or on the data).

Typically a DPA counter-measure for Rijndael is designed to protect an implementation of any cipher that is a composition of these operations.

2.4 Previously Proposed Masking Methods for AES

A natural method to protect (against side-channel attacks) a cipher using operations of type X and L is the transformed masking method [27, 1] in which any value inside the computation of the enciphering algorithm is masked with XOR by some random value. This masking method can also resist to higher-order DPA if the mask is split into several sub-masks as follows $R = R_1 \oplus \ldots \oplus R_k$, R is never computed but all sub-masks are used one after another.

The main question that arises in all proposed masking methods for AES is how to protect the AES S-box, and more precisely how to protect the Inv operation. In the past, the question has been answered with more or less success as follows:

1. Transformed Multiplicative Masking (TMM) of Akkar and Giraud [1]. Switching from additive (XOR) to multiplicative masking and back is possible and easy due to the ring structure of $GF(256)$. Unluckily, a multiplicative masking with respect to the multiplication in $GF(256)$ cannot be secure, because there is a special value $0 \in GF(256)$ that for any mask remains the same (i.e. is never masked).
2. In [39] Trichina *et al* proposed a simplified version of this method, which is not secure for exactly the same reason, see [20, 31].

3. Embedded Multiplicative Masking (EMM) of Golić and Tymen.
   One solution to the "zero-masking problem" is proposed in [20]. The method consists of a randomised embedding of $GF(256)$ in a larger ring in which zero can be represented by several possible elements (on two bytes). Then each byte taken separately has a uniform distribution.
4. A method based on univariate polynomials of Blömer, Merchan and Krummel [6]. This can be seen as a perfectly general method that can be applied to any S-box, as any function over a finite field can be seen as a univariate polynomial. Luckily, the polynomial representation $Inv(x) = x^{254}$ for any x (including 0) has only one monomial. This makes the method more efficient than in the general case and suitable for Rijndael.
5. In [38] Trichina and Korkishko propose a software-oriented masking method based on log tables.
6. In [33] Rostovtsev and Shemyakina propose to use isomorphisms of the underlying finite field.
7. Tower Fields Methods by Oswald, Trichina, *et al* [36, 37, 31, 32] are designed for hardware implementations. In these methods, the computing of Inv in $GF(2^{2k})$ is reduced to a secure computation with masked values of multiplications and inverses in $GF(2^k)$, by representing $GF(2^{2k})$ as a quadratic extension of $GF(2^k)$. Multiplications can be computed with additive masking and we are left with the problem of a secure computation of Inv at the lower level. Two versions have been proposed:
   - In [36, 37] Trichina, Korkishko and Hee Lee go down to the field $GF(16)$. At this level the problem is solved by a completely general method, as a masked computation of a combinatorial circuit with XOR and AND gates.
   - In [31, 32] Oswald, Mangard, Pramstaller and Rijmen go down to $GF(4)$ on which Inv is multivariate linear (linear over $GF(2)$, not over $GF(4)$) and easy to protect.

2.5 New Method - Defining the Target

In this paper we present a novel algebraic masking method for Inv. Our method is strictly more powerful and somewhat mathematically more elegant that all the other known methods. After the initial (insecure) proposal of multiplicative masking all the methods subsequently proposed have become more and more generic in a sense that had to **decompose** the Rijndael S-box in simpler operations and protect each of them separately. On the contrary in our new method we operate at higher level, and protect directly more complex operations by representing them as elements of some group. This is made possible by exhibiting a new algebraic structure that though a bit tricky to use, allows to protect Rijndael against side-channel analysis.

In our method, we will be able to make masked computations of Inv in one single step without "mask switching". We wish to go directly from a masked input to a masked output. Thus the operation that we wish to protect is - a fully masked Rijndael inversion defined as:

$$F_{(R,R')} : X \mapsto Inv(X \oplus R) \oplus R'$$

Though complex, this operation can be directly protected. The basic idea is as follows: if we can embed this operation in some group, we can protect it against DPA of any fixed order: we represent it as a composition of several transformations, for example k, out of which any subset of $k-1$ transformations are just a set of random uniformly distributed operations.

We will exploit the group of homographic transformations with some non-trivial mathematical and implementation technicalities: strictly speaking Inv does **not** belong to this group (while the real inverse in $GF(256)$ does). In our solution we will represent $F_{(R,R')}$ as a combination of a homographic transformation and of mapping that exchanges two points. This representation has a very interesting feature: these two type of mappings do "almost" commute (except that the two points to exchange are different). Thus we will also be able to mask completely the exchange of two points by additional homographic mappings.

3 Homographic Functions

In this paper we will work in $GF(2^n)$, for Rijndael S-boxes we have $n = 8$. The function Inv can be defined over $GF(2^n)$ in the same way as in Rijndael with the usual $0 \mapsto 0$:

$$Inv(X) = \begin{cases} X^{-1} & \text{in } GF(2^n) \text{ if } X \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

We also have the real inverse function that can be either defined as

- a function $GF(256)^* \rightarrow \overline{GF(256)^*}$ or
- as a projective function $\overline{GF(256)} \rightarrow \overline{GF(256)}$ with $\overline{GF(256)} = GF(256) \cup \{\infty\}$. This is our preferred version that will be used in this paper.

In mathematics the functions of the form $X \mapsto \frac{aX+b}{cX+d}$ are called *homographic functions* (a.k.a. linear fractional transformations or Möbius transformations, see [41]). It is well known that they can be represented by 2×2 matrices $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$. The composition of these functions is equivalent to multiplying their matrices. A cross-ratio of 4 pairwise different points $R(t, u, v, w) = \frac{t-u}{t-w} / \frac{v-u}{v-w}$ is known to be an invariant for such transformations, see [12, 4].

3.1 What Is the Difference Between Inv and $1/X$?

Unfortunately the function Inv of Rijndael is **not** strictly speaking a homographic function. It is equal to a function of the form $X \mapsto \frac{aX+b}{cX+d}$ except in one point, when 0 is mapped to 0. This "completion" with $0 \mapsto 0$ has many important and non-trivial properties, see [12]. There are three ways of defining a practical "inverse" function for the finite field $GF(256)$:

1. We can have a bijection on 255 elements $GF(256)^* \rightarrow GF(256)^*$.
2. We can have a bijection on 256 elements $Inv : GF(256) \rightarrow GF(256)$ that is used in Rijndael.
3. We can have a bijection on 257 elements $\overline{GF(256)} \rightarrow \overline{GF(256)}$ with $\overline{GF(256)} = GF(256) \cup \{\infty\}$. The "real" inverse is defined as:

$$\overline{Inv}(X) = \begin{cases} X^{-1} & \text{if } X \notin \{0, \infty\} \\ 0 \mapsto \infty & \\ \infty \mapsto 0 & \end{cases}.$$

This is an eminently interesting version that is of central interest to us. We can compose this function with other homographic permutations defined as follows:

$$H_{a,b,c,d}(X) \stackrel{def}{=} \begin{cases} \frac{aX+b}{cX+d} & \text{if } X \notin \{-\frac{d}{c}, \infty\} \\ & \\ -\frac{d}{c} \mapsto \infty & \\ \infty \mapsto \frac{a}{c} & \end{cases} \quad \text{with } det\begin{pmatrix} a & b \\ c & d \end{pmatrix} \neq 0.$$

The set of such invertible homographic mappings forms a group $\mathcal{H}$ under the usual composition law $\circ$. The matrix representation $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ in $GL_2(GF(2^n))$ is redundant and our group $\mathcal{H}$ is in fact isomorphic to a subgroup $SL_2(GF(2^n))$ of $GL_2(GF(2^n))$ in which all matrices are of determinant 1. Each element of $\mathcal{H}$ can be represented by "essentially" three elements of $GF(2^n)$.

3.2 Composition / Group Properties and Subtleties

We call an Almost-Invariant, any property that is invariant with a probability close to one. We have the following theorem:

Theorem 3.3 (Jakobsen-Knudsen-Courtois, cf. [12]). For any function $Y = F(X)$ that composes in any order

(a) N applications of Inv in $GF(2^n)$,
(b) any number of XORs with different subkeys or constants,

there exist $(a, b, c, d) \in GF(2^n)^4$ such that:

$$\mathbb{P}_{X \in GF(2^n)}\left[Y = \frac{aX+b}{cX+d} \mid Y = F(X)\right] \geq \left(1 - \frac{1}{2^n}\right)^N \geq \left(1 - \frac{N}{2^n}\right)$$

4 The New Masking Method

Let τ_{ab}, be a function that that swaps two points $a \neq b$:

$$\tau_{ab}(X) = \begin{cases} X & \text{if } X \notin \{a, b\} \\ a \mapsto b & \\ b \mapsto a & \end{cases}.$$

We observe that Inv is a restriction to $GF(256)$ of $\overline{Inv} \circ \tau_{O\infty}$. We have

$$\forall x \in GF(256) \qquad Inv\ (x) = \overline{Inv} \circ \tau_{O\infty}\ (x) = \tau_{O\infty} \circ \overline{Inv}\ (x)$$

Now, our target is to protect the whole operation as follows:

$$F_{(R,R')} : X \mapsto Inv(X \oplus R) \oplus R'$$

Though it is defined over $GF(2^n)$, nothing prevents us from extending it by deciding that $\overline{F_{(R,R')}}(\infty) = \infty$. We will be using operations in $\overline{GF(2^n)}$ to implement operations in $GF(2^n)$ and if we are able to securely implement $\overline{F_{(R,R')}}$ we obtain also a secure implementation of $F_{(R,R')}$.

Our new target is $\overline{F_{(R,R')}}$ and it can also be seen as a composition of an invertible homographic transformation and another mapping that swaps two points:

$$\overline{F_{(R,R')}} = H_{a,b,c,d} \circ \tau_{\infty R} = \tau_{\infty R'} \circ H_{a,b,c,d}$$

With a, b, c, d that can be computed explicitly as:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} R' & RR'+1 \\ 1 & R \end{pmatrix} \qquad \text{with } det \begin{pmatrix} R' & RR'+1 \\ 1 & R \end{pmatrix} = 1.$$

We will use the second decomposition (the first might be used as well). We have:

$$\overline{F_{(R,R')}} = \tau_{\infty R'} \circ H_{a,b,c,d}$$

It is important to see that it is not sufficient to find a secure implementation of $H_{a,b,c,d}$ and a secure implementation of $\tau_{\infty R'}$. This is because when the (real, not masked) value of the input of the Rijndael S-box is equal to 0, the output value of $H_{a,b,c,d}$ is always ∞ (∞ is not masked by R'). This is a projective version of the "zero-masking problem". In a secure implementation both operations have to be "jointly" protected. In particular the implementation of $\tau_{\infty R'} \circ H_{a,b,c,d}$ must hide both points that are exchanged ∞ and R', not only the point R'.

4.1 Joint Secure Implementation

At this level we will describe a method for usual DPA (1st order). We assume that in the implementation of Rijndael each entry and each output of the Inv function are protected by a couple of masks R and R' that vary from one S-box to another. We omit the description of how to protect the linear parts of the algorithm.

1. We pick a random $\alpha \in \mathcal{H}$.
2. We compute $\alpha^{-1} \in \mathcal{H}$.
3. By evaluating α^{-1} on R' and ∞, we compute two points $g, h \in \overline{GF(2^n)}$ such that:
$$\tau_{\infty R'} = \alpha \circ \tau_{gh} \circ \alpha^{-1}$$
It is possible to show that they are a random and uniformly distributed couple of distinct points in $\overline{GF(2^n)}$.

4. We compute $H' = \alpha^{-1} \circ H_{a,b,c,d}$.
5. We store the sequence α, τ_{gh}, H' in RAM as matrices of $SL_2(GF(2^n))$, except τ_{gh} that is stored as a couple of points in $\overline{GF(2^n)}$.
6. Our secure implementation of $F_{(R,R')}$ is defined as the following sequence of transformations to be applied to X in order from right to left:

$$\alpha, \quad \tau_{gh}, \quad H'$$

Security of our countermeasure against DPA. Observing each of the three transformations specified above is just a random transformation of some type that gives no information to the attacker. It is also the case for H' and $\tau_{gh} \circ H'$. This allows to see that all the intermediate values $H'(X)$ and $\tau_{gh}(H'(X))$ before the final masked output are fully de-correlated from X, from the real values inside the AES, and from the key.

5 Conclusion and Further Research

In this paper we studied the interaction between additive masking and the inverse S-box of Rijndael the happens to be less complex than it seems. It allows to construct a method to protect in one single step the whole masked inverse against all passive side-channel attacks (e.g. DPA, DEMA, etc.). This is achieved by embedding (with additional non-trivial technicalities) this operation in a group of homographic transformations over the projective space that happens to have a compact representation (essentially 3 bytes).

Further Developments. This paper is a proof of concept for a new non-trivial algebraic masking method. We develop it more in the extended version of this paper that can be found at `eprint.iacr.org/2005/204/`. The topic requires further research.

References

1. Mehdi-Laurent Akkar, C. Giraud: *An Implementation of DES and AES secure against Some Attacks*. In CHES'2001, LNCS 2162, pp. 309-318, Springer, 2001.
2. Mehdi-Laurent Akkar, Louis Goubin, *A Generic Protection against High-Order Differential Power Analysis*. In FSE'2003, LNCS 2887, Springer, pp. 192-205, 2003.
3. Mehdi-Laurent Akkar, Régis Bevan, Louis Goubin, *Two Power Analysis Attacks against One-Mask Methods*. In FSE'2004, LNCS 3017, Springer, pp. 332-347, 2004.
4. Kazuaro Aoki, Serge Vaudenay: *On the Use of GF-Inversion as a Cryptographic Primitive,* SAC 2003, LNCS 3006, pp. 234-247, Springer 2004.
5. Eli Biham, Adi Shamir: *Power Analysis of the Key Scheduling of the AES Candidates*. the second Advanced Encryption Standard (AES) Candidate Conference, March 1999. Available from `http://csrc.nist.gov/encryption/aes/` round1/Conf2/aes2conf.htm
6. Johannes Blömer, Jorge Guajardo Merchan, Volker Krummel: *Provably Secure Masking of AES,* In SAC'2004, LNCS 3357, pp. 69-83, Springer, 2005. Available on `eprint.iacr.org/2004/101`.

7. Johannes Blömer, Jean-Pierre Seifert: *Fault based cryptanalysis of the Advanced Encryption Standard,* In Proceedings of Financial Cryptography 2004, LNCS 2742, pp 162-181, Springer 2004. Available on `eprint.iacr.org/2002/075`.
8. Eric Brier, Christophe Clavier, Francis Olivier: *Optimal Statistical Power Analysis,* Available on `eprint.iacr.org/2003/152`.
9. Vincent Carlier, Hervé Chabanne, Emmanuelle Dottax, Hervé Pelletier: *Electromagnetic Side Channels of an FPGA Implementation of AES*, `eprint.iacr.org/2004/145`.
10. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, Pankaj Rohatgi: *A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards*. the second Advanced Encryption Standard (AES) Candidate Conference, March 1999. Available from `http://csrc.nist.gov/encryption/aes/round1/Conf2/aes2conf.htm`
11. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, Pankaj Rohatgi: *Towards Sound Approaches to Counteract Power-Analysis Attacks*. In Crypto 99, LNCS 1666, pp. 398-412, Springer, 1999.
12. Nicolas Courtois: *The Inverse S-box, Non-linear Polynomial Relations and Cryptanalysis of Block Ciphers,* in AES 4 Conference, Bonn May 10-12 2004, LNCS 3373, pp. 170-188, Springer, 2005.
13. Nicolas Courtois: *The Inverse S-box and Two Paradoxes of Whitening,* Long extended version of the Crypto 2004 rump session presentation, *Whitening the AES S-box,* Available at `http://www.minrank.org/invglc_rump_c04.zip`. Also explained in Appendix B of the extended version of [12].
14. Joan Daemen, Vincent Rijmen: *AES proposal: Rijndael,* The latest revised version of the proposal is available on the internet, `http://csrc.nist.gov/encryption/` aes/rijndael/Rijndael.pdf
15. Joan Daemen, Vincent Rijmen: *The Design of Rijndael. AES - The Advanced Encryption Standard,* Springer-Verlag, Berlin 2002. ISBN 3-540-42580-2.
16. Joan Daemen, Vincent Rijmen: *Resistance Against Implementation Attacks: A Comparative Study of the AES Proposals,* In 2nd AES Candidate Conference, March 1999. Available from `http://csrc.nist.gov/encryption/` aes/round1/Conf2/aes2conf.htm
17. Pierre Dusart, Gilles Letourneux, Olivier Vivolo: *Differential Fault Analysis on A.E.S.,* Rapport interne n2003-01, Laco, Université de Limoges, France. Available on `eprint.iacr.org/2003/010`.
18. Christophe Giraud: *DFA on AES,* In Proceedings of AES4 Conference, LNCS 3373, Springer, 2005. Available on `eprint.iacr.org/2003/008`.
19. Jovan Dj. Golić: *Multiplicative Masking and Power Analysis of AES,* claimed as being presented at an (internal) Gemplus Quarterly meeting, La Ciotat, France, October 30-31, 2001, In CHES 2002, LNCS 2523, pp. 198-212, Springer, 2003. Available at `http://eprint.iacr.org/2002/091/`.
20. Jovan Dj. Golić, Christophe Tymen: *Multiplicative Masking and Power Analysis of AES,* In CHES 2002, LNCS 2523, pp. 198-212, Springer, 2003. Available at `http://eprint.iacr.org/2002/091/`.
21. Louis Goubin, Jacques Patarin: *Procédé de sécurisation d'un ensemble électronique de cryptographie à clé secrète contre les attaques par analyse physique*. European Patent, Axalto, (previously SchlumbergerSema), February 4th, 1999, Publication Number: 2789535.
22. Louis Goubin, Jacques Patarin: *DES and Differential Power Analysis – The Duplication Method*. In CHES'99, LNCS 1717, pp. 158-172, Springer, 1999.

23. Paul Kocher, Joshua Jaffe, Benjamin Jun: *Introduction to Differential Power Analysis and Related Attacks*. Technical Report, Cryptography Research Inc., 1998. Available from `http://www.cryptography.com/dpa/technical/index.html`
24. Paul Kocher, Joshua Jaffe, Benjamin Jun: *Differential Power Analysis*. In Crypto 99, LNCS 1666, pp. 388-397, Springer, 1999.
25. Thomas Jakobsen, Lars Knudsen: *Attacks on Block Ciphers of Low Algebraic Degree,* Journal of Cryptology 14(3): 197-210 (2001).
26. Thomas Jakobsen, Lars R. Knudsen: *The Interpolation Attack on Block Ciphers,* FSE 97, LNCS 1267, Springer, pp.28-40, 1997.
27. Thomas S. Messerges: *Securing the AES Finalists Against Power Analysis Attacks,* FSE'00, LNCS 1978, pp. 150-164, Springer, 2001.
28. Thomas S. Messerges: *Using second-Order Power Analysis to Attack DPA Resistant software,* . In CHES'2000, LNCS 1965, pp. 238-251, Springer, 2000.
29. Thomas S. Messerges, Ezzat A. Dabbish, Robert H. Sloan: *Investigations of Power Analysis Attacks on Smartcards*. the USENIX Workshop on Smartcard Technology, pp. 151-161, May 1999. Available from `http://www.eecs.uic.edu/` tmesserg/papers.html
30. Thomas S. Messerges, Ezzat A. Dabbish, Robert H. Sloan: *Power Analysis Attacks of Modular Exponentiation in Smartcards*. In CHES'99, LNCS 1717, pp. 144-157, Springer, 1999.
31. Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller: *Secure and Efficient Masking of AES - A Mission Impossible ?,* `eprint.iacr.org/2004/134`.
32. Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller, Vincent Rijmen: *A Side-Channel Analysis Resistant Description of the AES S-box,* In FSE 2005, to appear in LNCS Springer, 2005.
33. A.G. Rostovtsev and O.V. Shemyakina; *AES side channel attack protection using random isomorphisms,* Available on `http://eprint.iacr.org/2005/087.pdf`
34. Emmanuel Prouff: *DPA attacks and S-boxes,* In FSE'05, to appear in LNCS, Springer, 2005.
35. François-Xavier Standaert, Siddika Berna Ors, Bart Preneel: *Power Analysis of an FPGA Implementation of Rijndael: Is Pipelining a DPA Countermeasure ?* In CHES'2004, LNCS 3156, pp. 30-44, Springer, 2004.
36. Elena Trichina: *Combinational Logic Design for AES SubByte Transformation on Masked Data,* . Available on `eprint.iacr.org/2003/236`.
37. Elena Trichina, Tymur Korkishko, Kyung Hee Lee: *Small Size, Low Power, Side Channel-Immune AES Coprocessor, Design and Synthesis Results,* 4th AES Conference, LNCS 3373, to appear in Springer, 2005.
38. Elena Trichina, Lesya Korkishko: *Secure and Efficient AES Software Implementation for Smart Cards,* In Proceedings of WISA 2004, LNCS 3325, Pages 425-439, Springer, 2005. Available on `eprint.iacr.org/2004/149`.
39. Elena Trichina, Domenico De Seta, Lucia Germani: *Simplifed Adaptive Multiplicative Masking for AES,* In CHES 2002,LNCS 2535, pp. 187-197, Springer, 2003.
40. Jason Waddle, David Wagner: *Towards Efficient Second-Order Power Analysis,* In CHES'2004, LNCS 3156, pp. 1-15, Springer, 2004.
41. The Wikipedia entry "Möbius transformation", freely available at `http://en.wikipedia.org/wiki/Mobius_group`

Characterisations of Extended Resiliency and Extended Immunity of S-Boxes

Josef Pieprzyk[1], Xian-Mo Zhang[1], and Jovan Dj. Golić[2]

[1] Centre for Advanced Computing - Algorithms and Cryptography,
Department of Computing, Macquarie University,
Sydney, NSW 2109, Australia
{josef, xianmo}@ics.mq.edu.au
[2] Telecom Italia Lab, Telecom Italia,
Via G. Reiss Romoli 274, 10148 Turin, Italy
jovan.golic@tilab.com

Abstract. New criteria of extended resiliency and extended immunity of vectorial Boolean functions, such as S-boxes for stream or block ciphers, were recently introduced. They are related to a divide-and-conquer approach to algebraic attacks by conditional or unconditional equations. Classical resiliency turns out to be a special case of extended resiliency and as such requires more conditions to be satisfied. In particular, the algebraic degrees of classically resilient S-boxes are restricted to lower values. In this paper, extended immunity and extended resiliency of S-boxes are studied and many characterisations and properties of such S-boxes are established. The new criteria are shown to be necessary and sufficient for resistance against the divide-and-conquer algebraic attacks by conditional or unconditional equations.

Keywords: Extended Resiliency, Extended Immunity, Divide-and-Conquer Algebraic Attacks.

1 Introduction

The concept of divide-and-conquer algebraic attacks by the *conditional* or *unconditional* equations induced from a cipher was introduced recently by Golić in [15]. The basic idea behind the resistance against the new attacks is to design the ciphers so that an attacker cannot induce non-constant equations involving certain subsets of variables within the cipher. For this reason, the notions of extended resiliency along with extended immunity as a special case were introduced in [15].[1] As a special case of extended resiliency, classical resiliency (see for instance [1, 2, 5, 6, 16, 20, 21, 23, 24]) requires additional conditions that are not necessary for resistance against the algebraic attacks by conditional or

[1] The name 'extended resiliency (immunity)' in this paper corresponds to 'algebraic immunity (resiliency)' in [15]. The name is changed in order to avoid confusion with [4, 12, 13], where 'algebraic immunity' was defined differently.

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 210–228, 2006.
© Springer-Verlag Berlin Heidelberg 2006

unconditional equations. Furthermore, the additional conditions restrict the algebraic degrees [3, 21, 22]. Therefore, although possibly useful to resist other types of attacks, such as correlation attacks, classically resilient S-boxes are not good candidates for cryptographic blocks that should resist algebraic attacks by conditional or unconditional equations. The extended resiliency (immunity) thus seems to be an appropriate platform for a study of S-boxes that are used in stream or block ciphers. These S-boxes are "provably" immune against the divide-and-conquer algebraic attacks by conditional (unconditional) equations and, unlike classically resilient S-boxes, can have high algebraic degrees.

The aim of the paper is to characterise extended resiliency and extended immunity. More precisely, in Section 2, we recall and describe the divide-and-conquer algebraic attacks by unconditional (conditional) equations induced from an S-box. The algebraic properties of S-boxes to be used in the rest of the paper are provided in Section 3. The extended resiliency and the extended immunity are characterised in Sections 4 and 5, respectively. In Section 6, the extended resiliency (immunity) is characterised in terms of the resistance against algebraic attacks by unconditional (conditional) equations. The relations between the extended immunity, the extended resiliency and the classical resiliency are summarised in Section 7. In Section 8, we demonstrate that algebraic degrees of extended resilient (immune) S-boxes can be as high as $n-1$, where n denotes the input size, and provide the corresponding constructions. In Section 9, an upper bound on extended immunity (resiliency) is analysed. Conclusions and suggestions for future work are given in Section 10. Proofs of mathematical results are provided in the Appendix.

2 Divide-and-Conquer Algebraic Attacks Based on Conditional and Unconditional Equations

Algebraic attacks [5, 7, 9, 10, 11, 12, 13, 15, 18] have recently been shown to be very powerful against certain types of both stream and block ciphers. Typically, an algebraic attack consists of the following two stages. In the first stage, the attacker finds a collection of equations that holds for some specific input, intermediate, and output variables for the cipher. In the second stage, the attacker observes the accessible variables, fixes the known variables to the observed values, and solves the resulting system of equations. The solution normally reduces the uncertainty of the unknown variables such as the secret key and in some circumstances, the attacker can determine all unknown variables breaking the cipher. The amount of work involved in this attack depends on the algebraic degree of the equations derived by the adversary. The smaller the degree of the equations the more efficient the attack is. To prevent ciphers against algebraic attacks, one would expect that the internal structure of the ciphers does not permit the adversary to derive low degree non-constant equations.

A concept of divide-and-conquer algebraic attacks is recently proposed in [15]. It suggests that algebraic attacks can be based on equations involving only subsets of input or output variables for individual nonlinear components of a cipher

such as S-boxes or lookup tables. The equations can be unconditional, involving both input and output variables, or conditioned on the output, involving only the input variables. The conditional scenario is shown to be useful for stream ciphers (e.g., based on linear feedback shift registers), while the unconditional scenario may possibly be used for both block ciphers and stream ciphers. More precisely, in iterated constructions, such equations may possibly reduce the number of intermediate variables involved in the equations and hence also the complexity of algebraic attacks. The two scenarios of induced algebraic equations are described next.

Let $F(x) = y$ be an $n \times m$ S-box, where $x = (x_1, \ldots, x_n) \in GF(2)^n$ and $y \in GF(2)^m$. [2] For a fixed integer t, $0 \le t \le n$, let $T = \{j_1, \ldots, j_t\}$ be fixed ordered t-subset of $\{1, \ldots, n\}$ and $\{i_1, \ldots, i_{n-t}\} = \{1, \ldots, n\} \setminus \{j_1, \ldots, j_t\}$, where $j_1 < \cdots < j_t$ and $i_1 < \cdots < i_{n-t}$. Set $x' = (x_{j_1}, \ldots, x_{j_t})$ and $x'' = (x_{i_1}, \ldots, x_{i_{n-t}})$. We define a subset $W(F, T)$ of $GF(2)^{t+m}$ as follows.

$$W(F,T) = \{(\alpha', \beta) \mid \alpha' \in GF(2)^t, \beta \in GF(2)^m, \\ (\exists\, \alpha'' \in GF(2)^{n-t},\ x' = \alpha', x'' = \alpha'', F(x) = \beta)\}. \tag{1}$$

Let a function h on $GF(2)^{t+m}$ satisfy

$$h(x', y) = 0, \ \text{ for all } (x', y) \in W(F, T). \tag{2}$$

Then the equation (2), over x' and y, is called an *unconditional algebraic equation* induced from $F(x) = y$ (for the fixed T). Of course, such h always exists as h can be the constant zero function. However the attackers try to find a non-constant h so as to eliminate x'' and involve only the variables in x' and y. To make this divide-and-conquer strategy ineffective, it is desirable that F does not induce non-constant unconditional algebraic equations, e.g., for relatively small values of t.

In particular, when $\beta \in F(GF(2)^n)$ in (1) is fixed, we define a subset $W(F, T, \beta)$ of $GF(2)^t$ as follows.

$$W(F,T,\beta) = \{\alpha' \mid \alpha' \in GF(2)^t, \\ (\exists\, \alpha'' \in GF(2)^{n-t},\ x' = \alpha', x'' = \alpha'', F(x) = \beta)\}. \tag{3}$$

Let a function h on $GF(2)^t$ satisfy

$$h(x') = 0, \ \text{ for all } x' \in W(F, T, \beta). \tag{4}$$

Then the equation (4), over x' only, is called a *conditional algebraic equation* induced from $F(x) = \beta$. Similarly, it may be desirable that F does not induce non-constant conditional algebraic equations, e.g., for relatively small values of t. The extended immunity (resiliency) of vectorial Boolean functions is defined in [15] in order to describe the divide-and-conquer effect of induced algebraic equations. This will be studied in more detail in Section 6 and provides a practical

[2] Here and throughout, we use a simplified notation $GF(2)^n$ for $(GF(2))^n$.

motivation for our work. On the other hand, the extended resiliency (immunity) naturally generalises the well-known notion of classical resiliency (immunity) and it is thus theoretically interesting to investigate its properties and propose new constructions.

3 Algebraic Properties of S-Boxes

In this section, we provide the necessary background including notations that are used in the next sections. We write all vectors in $GF(2)^n$ as $(0,\ldots,0,0) = \alpha_0$, $(0,\ldots,0,1) = \alpha_1$, ..., $(1,\ldots,1,1) = \alpha_{2^n-1}$, and call α_i the *binary representation* of integer i, $i = 0, 1, \ldots, 2^n - 1$. A Boolean function f is a mapping from $GF(2)^n$ to $GF(2)$. We express f as $f(x) = f(x_1,\ldots,x_n)$ where $x = (x_1,\ldots,x_n) \in GF(2)^n$. The *truth table* of f is defined as $(f(\alpha_0), f(\alpha_1), \ldots, f(\alpha_{2^n-1}))$, and the *sequence* of f is defined as $((-1)^{f(\alpha_0)}, (-1)^{f(\alpha_1)}, \ldots, (-1)^{f(\alpha_{2^n-1})})$. The scalar product of $\alpha = (a_1,\ldots,a_n)$, $\beta = (b_1,\ldots,b_n) \in GF(2)^n$, denoted by $\langle\alpha,\beta\rangle$, is defined as $\langle\alpha,\beta\rangle = a_1b_1 \oplus \cdots \oplus a_nb_n$ where $\oplus$ denotes the binary addition. We call $h(x) = \langle\alpha, x\rangle \oplus c$ an *affine function*, where $\alpha, x \in GF(2)^n$ and $c \in GF(2)$. In particular, h is called a *linear function* if $c = 0$.

Consider a mapping $F = (f_1,\ldots,f_m)$ from $GF(2)^n$ to $GF(2)^m$, where each f_j is a Boolean function on $GF(2)^n$ and is called a coordinate or component function of F. We express F as $F(x) = F(x_1,\ldots,x_n)$ where $x = (x_1,\ldots,x_n) \in GF(2)^n$. F is also called an *S-box* or a *vectorial Boolean function*. From now, we call F an $n \times m$ S-box. F is said to be *affine* if all its coordinate functions are affine, and in particular, F is said to be *linear* if all its coordinate functions are linear. For any k, $1 \le k \le m$, and any k-subset $\{j_1,\ldots,j_k\}$ of $\{1,\ldots,m\}$, where $j_1 < \cdots < j_t$, the mapping $\hat{F} = (f_{j_1},\ldots,f_{j_k})$ from $GF(2)^n$ to $GF(2)^k$ is called a *k-subfunction* of F.

Notation 1. *Let $\alpha_i \in GF(2)^n$ be the binary representation of integer i, $i = 0,\ldots,2^n-1$, and $\gamma_j \in GF(2)^m$ be the binary representation of integer j, $j = 0,\ldots,2^m-1$. For an $n \times m$ S-box F, we define a $2^n \times 2^m$ $(1,-1)$ matrix $D_F = (d_{ij})$: $d_{ij} = (-1)^{\langle F(\alpha_i),\gamma_j\rangle}$. Also we define a $2^n \times 2^m$ real-valued $(0,1)$ matrix $C_F = (c_{ij})$: $c_{ij} = 1$ if and only if $F(\alpha_i) = \gamma_j$.*

Recall that a $k \times k$ $(1,-1)$-matrix M is called a *Hadamard* matrix if $MM^T = kI_k$, where M^T is the transpose of M and I_k is the $k \times k$ identity matrix [17]. A $2^s \times 2^s$ Sylvester-Hadamard matrix, denoted by H_s, is defined by the following recursive relation: $H_0 = 1$, $H_s = \begin{bmatrix} H_{s-1} & H_{s-1} \\ H_{s-1} & -H_{s-1} \end{bmatrix}$, $s = 1, 2, \ldots$. Clearly H_s is a symmetric matrix. Denote the jth row (column) of H_m by ℓ_j (ℓ_j^T), $j = 0, 1, \ldots .2^m - 1$. It is known that ℓ_j is the sequence of a linear function $\psi_j(y) = \langle\gamma_j, y\rangle$ where $y \in GF(2)^m$.

Lemma 1. *Let F be an $n \times m$ S-box. Then $D_F H_m = 2^m C_F$.*

Lemma 2. *Let F be an $n \times m$ S-box. Let $diag(\lambda_0, \lambda_1, \ldots, \lambda_{2^m-1})$ be a diagonal matrix, where λ_j denotes the number of times that F takes value γ_j. Then $D_F^T D_F = H_m diag(\lambda_0, \lambda_1, \ldots, \lambda_{2^m-1}) H_m$.*

Recall the basic facts from linear algebra, for instance, from [19]. If an $s \times s$ matrix A with real entries, a nonzero s-dimensional column vector η with real coordinates and a real number κ satisfy $A\eta = \kappa\eta$, then κ is called the eigenvalue of matrix A corresponding to the eigenvector η or, alternatively, η is called an eigenvector of matrix A corresponding to the eigenvalue κ. For a fixed matrix A, each eigenvector corresponds to only one eigenvalue, whereas an eigenvalue does not necessarily correspond to only one eigenvector. Usually, a real square matrix does not necessarily have a real eigenvector. However, any real symmetric $s \times s$ matrix must have s linearly independent real eigenvectors.

Corollary 1. *Let F be an $n \times m$ S-box. Then the jth column ℓ_j^T of H_m is the eigenvector of $D_F^T D_F$ corresponding to the eigenvalue $2^m \lambda_j$, where λ_j denotes the number of times that F takes value γ_j.*

4 Extended Resilient S-Boxes

The concept of extended resiliency was originally proposed by Golić [15]. In this section, we derive various characterisations of the extended resiliency.

4.1 Surjective S-Boxes

Before defining the extended resiliency, we introduce necessary notations.

Notation 2. *Let F be an $n \times m$ S-box. For a subset S of $GF(2)^n$, we write $\{F(x) \mid x \in S\} = F(S)$.*

Definition 1. *Let F be an $n \times m$ S-box. F is said to be* surjective *(or* onto $GF(2)^m$*) if $F(GF(2)^n) = GF(2)^m$.*

Lemma 3. *Let F be an $n \times m$ S-box. Then the following statements are equivalent: (i) F is surjective, (ii) all eigenvalues of $D_F^T D_F$ are nonzero, and (iii) the rank of D_F is 2^m.*

Definition 2. *Functions $f_1, \ldots, f_m$ on $GF(2)^n$ are said to be* functionally independent *if for any non-constant Boolean function h on $GF(2)^m$, $h(f_1, \ldots, f_m)$ is non-constant.*

Clearly linear independence is a special case of functional independence.

Lemma 4. *Let F be an $n \times m$ S-box. Then the following statements are equivalent: (i) F is surjective, (ii) for any integer k, $1 \le k \le m$, and any surjective $m \times k$ S-box P, the $n \times k$ S-box $P(F(x))$ is surjective, and (iii) the coordinate functions of F are functionally independent.*

4.2 Extended Resiliency and Its Properties

Notation 3. *For a fixed t-subset $T = \{j_1, \ldots, j_t\}$ of $\{1, \ldots, n\}$ and a fixed vector $\alpha = (a_1, \ldots, a_t) \in GF(2)^t$, we define a subset $S(n, T, \alpha)$ of $GF(2)^n$: $S(n, T, \alpha) = \{x = (x_1, \ldots, x_n) \mid x \in GF(2)^n, x_{j_1} = a_1, \ldots, x_{j_t} = a_t\}$.* [3] *Formally, for $t = 0$, a 0-subset T is the empty set, i.e., $T = \emptyset$ and α is not defined, then $S(n, T, \alpha)$ becomes $GF(2)^n$.*

Let $\#X$ denote the number of elements in a set X. Then $\#S(n, T, \alpha) = 2^{n-t}$.

Lemma 5. *For fixed subsets $T = \{j_1, \ldots, j_t\}$, $T' = \{j_1, \ldots, j_{t-1}\}$ and fixed vectors $\alpha = (a_1, \ldots, a_t) \in GF(2)^t$ and $\alpha' = (a_1, \ldots, a_{t-1}) \in GF(2)^{t-1}$, we have $S(n, T, \alpha) \subseteq S(n, T', \alpha')$.*

Definition 3. *Let F be an $n \times m$ S-box. Then F is said to be* (n, m, t)-extended resilient *if for any t-subset T of $\{1, \ldots, n\}$ and any $\alpha \in GF(2)^t$, we have $F(S(n, T, \alpha)) = GF(2)^m$. An (n, m, t)-extended resilient S-box is also said to be* t-extended resilient *if we ignore the particular values of n and m.*

It follows that any (n, m, t)-extended resilient S-box is surjective, in particular, any $(n, m, 0)$-extended resilient S-box is equivalent to a surjective $n \times m$ S-box.

Proposition 1. *For any (n, m, t)-extended resilient S-box, it is necessary that $t \leq n - m$.*

The following claim directly follows from Lemma 5.

Lemma 6. *Let F be an $n \times m$ S-box. Then F is (n, m, t)-extended resilient if and only if F is (n, m, k)-extended resilient for $k = 0, \ldots, t$.*

Due to Lemma 6, we are able to introduce the following definition.

Definition 4. *If F is an (n, m, t)-extended resilient S-box, but is not $(n, m, t + 1)$-extended resilient, then t is called the* extended resiliency order *of F.*

Proposition 2. *Let F be an (n, m, t)-extended resilient S-box. Then $F(x)$ runs through each vector in $GF(2)^m$ at least 2^t times while x runs through $GF(2)^n$ once.*

4.3 Characterisations of Extended Resilient S-Boxes

Definition 5. *Let S be a subset of $GF(2)^n$. Then the* characteristic function *of S, denoted by χ_S, is a Boolean function on $GF(2)^n$ defined as $\chi_S(\alpha) = 1$ if and only if $\alpha \in S$.*

Theorem 1. *Let F be an $n \times m$ S-box. Then the following statements are equivalent: (i) F is an (n, m, t)-extended resilient, (ii) for any fixed t-subset T of $\{1, \ldots, n\}$ and any fixed $\alpha \in GF(2)^t$, all eigenvalues of $D_F^T diag(b_0, b_1, \ldots,$*

[3] Here and throughout, a t-subset $\{j_1, \ldots, j_t\}$ is assumed to be ordered so that $j_1 < \cdots < j_t$.

$b_{2^n-1})D_F$ *are nonzero, where* $(b_0, b_1, \ldots, b_{2^n-1})$ *denotes the truth table of the characteristic function of* $S(n, T, \alpha)$ *and each* b_j *is regarded a real number, and (iii) the rank of* $diag(b_0, b_1, \ldots, b_{2^n-1})D_F$ *is* 2^m.

The next claim follows from Lemma 4 and Definition 3.

Theorem 2. *Let* F *be an* $n \times m$ *S-box. Then the following statements are equivalent: (i)* F *is* (n, m, t)*-extended resilient, (ii) for any integer* k, $1 \le k \le m$, *and any surjective* $m \times k$ *S-box* P, *the* $n \times k$ *S-box* $P(F(x))$ *is* (n, k, t)*-extended resilient, and (iii) for any* t*-subset* $T = \{j_1, \ldots, j_t\}$ *of* $\{1, \ldots, n\}$ *and any* $\alpha = (a_1, \ldots, a_t) \in GF(2)^t$, *the coordinate functions of* $F(x)|_{x_{j_1}=a_1,\ldots,x_{j_t}=a_t}$, *i.e., the restriction of* F *to* $S(n, T, \alpha)$, *are functionally independent.*

The necessity in the following statement holds due to Theorem 2 and the sufficiency is obvious.

Corollary 2. *Let* F *be an* $n \times m$ *S-box. Then* F *is* (n, m, t)*-extended resilient if and only if for any* k, $1 \le k \le m$, *every* k*-subfunction* $\hat{F}$ *of* F *is* (n, k, t)*-extended resilient.*

Theorem 3. *Let* F *be an* $n \times m$ *S-box. Then* F *is* (n, m, t)*-extended resilient if and only if for any fixed* r, $1 \le r \le t$, *any fixed* r*-subset* $T = \{j_1, \ldots, j_r\}$ *of* $\{1, \ldots, n\}$ *and every nonzero Boolean function* g *on* $GF(2)^r$, $g(x_{j_1}, \ldots, x_{j_r})F(x)$ *is surjective, i.e.,* $\{g(x_{j_1}, \ldots, x_{j_t})F(x) | x \in GF(2)^n\} = GF(2)^m$.

The next theorem is helpful for understanding the extended resiliency.

Theorem 4. *Let* F *be an* $n \times m$ *S-box. Then the following statements are equvalent: (i)* F *is* (n, m, t)*-extended resilient, (ii) for any integer* t_0, $0 \le t_0 \le t$, *any* t_0*-subset* T_0 *of* $\{1, \ldots, n\}$ *and any* $\alpha \in GF(2)^{t_0}$, *the restriction of* $F(x)$ *to* $S(n, T_0, \alpha)$ *is* $(t - t_0)$*-extended resilient, and (iii) for any integer* t_0, $0 \le t_0 \le t$, *any* t_0*-subset* T_0 *of* $\{1, \ldots, n\}$ *and any* $\alpha \in GF(2)^{t_0}$, $F(x)$ *runs through each vector in* $GF(2)^m$ *at least* 2^{t-t_0} *times while* x *runs through* $S(n, T_0, \alpha)$ *once.*

The following statement follows from Theorem 4.

Corollary 3. *Let* F *be an* (n, m, t)*-extended resilient S-box. For any integer* $k \ge 1$, *define an* $(n+k) \times m$ *S-box* F^* *as* $F^*(\alpha, \beta) = F(\alpha)$ *for each* $\alpha \in GF(2)^n$ *and* $\beta \in GF(2)^k$. *Then* F^* *is* $(n + k, m, t)$*-extended resilient.*

5 Extended Immune S-Boxes

The extended immunity proposed by Golić [15] is more general than the extended resiliency. In this section, we derive characterisations of the extended immunity.

5.1 Extended Immunity and Its Properties

Definition 6. *Let* F *be an* $n \times m$ *S-box. Then* F *is said to be* (n, m, t)-extended immune *if for any* t*-subset* T *of* $\{1, \ldots, n\}$ *and any* $\alpha \in GF(2)^t$, *we have* $F(S(n, T, \alpha)) = F(GF(2)^n)$. *An* (n, m, t)*-extended immune S-box is also said to be* t-extended immune *if we ignore the particular values of* n *and* m.

An (n, m, t)-extended immune S-box is (n, m, t)-extended resilient if and only if $F(GF(2)^n) = GF(2)^m$. Recall that $S(n, T, \alpha)$ with $\#T = 0$ denotes $GF(2)^n$. Thus any $n \times m$ S-box is $(n, m, 0)$-extended immune.

Proposition 3. *For any (n, m, t)-extended immune S-box F, it is necessary that $t \leq n - \log_2 \#F(GF(2)^n)$.*

In particular, if F is an (n, m, t)-extended immune S-box, then the inequality $t \leq n - \log_2 \#F(GF(2)^n)$ becomes $t \leq n - m$, as in Proposition 1.

Lemma 5 and Definition 6 imply the following lemma.

Lemma 7. *Let F be an $n \times m$ S-box. Then F is (n, m, t)-extended immune if and only if F is (n, m, k)-extended immune for $k = 0, \ldots, t$.*

According to Lemma 7, we are able to introduce the following definition.

Definition 7. *If F is an (n, m, t)-extended immune S-box, but is not $(n, m, t+1)$-extended immune, then t is called the* extended immunity order *of F.*

Similarly to Proposition 2, we have the following more general statement.

Proposition 4. *Let F be an (n, m, t)-extended immune S-box. Then $F(x)$ runs through each vector in $F(GF(2)^n)$ at least 2^t times while x runs through $GF(2)^n$ once.*

5.2 Characterisations of Extended Immune S-Boxes

We start with the following simple result.

Theorem 5. *Let F be an $n \times m$ S-box. Then the following statements are equivalent: (i) F is (n, m, t)-extended immune and (ii) for any fixed t-subset T of $\{1, \ldots, n\}$ and any two vectors $\alpha, \alpha' \in GF(2)^t$, we have $F(S(n, T, \alpha)) = F(S(n, T, \alpha'))$.*

By using a similar argument in the proof of Theorem 1, we can prove Theorem 6, whereas Theorem 7 and Corollary 4 correspond to Theorem 2 and Corollary 2, respectively.

Theorem 6. *Let F be an $n \times m$ S-box. Then the following statements are equivalent: (i) F is (n, m, t)-extended immune and (ii) for any fixed t-subset T of of $\{1, \ldots, n\}$ and any fixed $\alpha \in GF(2)^t$, the eigenvalue corresponding to the eigenvector ℓ_j^T of $D_F^T diag(b_0, b_1, \ldots, b_{2^n-1}) D_F$ is nonzero if and only if the the eigenvalue corresponding to the eigenvector ℓ_j^T of $D_F^T D_F$ is nonzero, where ℓ_j^T is the jth column of H_m, $j = 0, 1, \ldots, 2^m - 1$, $(b_0, b_1, \ldots, b_{2^n-1})$ denotes the truth table of the characteristic function of $S(n, T, \alpha)$ and each b_j is regarded as a real number.*

Theorem 7. *Let F be an $n \times m$ S-box. Then the following statements are equivalent: (i) F is (n, m, t)-extended immune, (ii) for any integer k, $1 \leq k \leq m$,*

and any $m \times k$ S-box P (not necessarily surjective), $P(F(x))$ is (n,k,t)-extended immune, and (iii) for any t-subset $T = \{j_1, \ldots, j_t\}$ of $\{1, \ldots, n\}$, any $\alpha = (a_1, \ldots, a_t) \in GF(2)^t$, and any Boolean function h on $GF(2)^m$, if $h(F)$ is non-constant, then $h(G)$ is non-constant, where $G(x) = F(x)|_{x_{j_1}=a_1,\ldots,x_{j_t}=a_t}$.

Corollary 4. *Let F be an $n \times n$ S-box. Then F is (n,m,t)-extended immune if and only if for any k, $1 \le k \le m$, every k-subfunction $\hat{F}$ of F is (n,k,t)-extended immune.*

By using a similar argument as in the proof of Theorem 3, we can prove the following characterisation.

Theorem 8. *Let F be an $n \times m$ S-box. Then F is (n,m,t)-extended immune if and only if for any fixed r, $1 \le r \le t$, any fixed r-subset $T = \{j_1, \ldots, j_r\}$ of $\{1, \ldots, n\}$ and every nonzero Boolean function g on $GF(2)^r$, $\{g(x_{j_1}, \ldots, x_{j_r})F(x) | x = (x_1, \ldots, x_n) \in GF(2)^n\} = F(GF(2)^n)$.*

By the same reasoning as for Theorem 4, we can also prove the following theorem, whereas Corollary 5 corresponds to Corollary 3.

Theorem 9. *Let F be an $n \times m$ S-box. Then the following statements are equivalent: (i) F is (n,m,t)-extended immune, (ii) for any integer t_0, $0 \le t_0 \le t$, any t_0-subset T_0 of $\{1, \ldots, n\}$ and any $\alpha_0 \in GF(2)^{t_0}$, the restriction of $F(x)$ to $S(n, T_0, \alpha_0)$ is $(t - t_0)$-extended immune, and (iii) for any integer t_0, $0 \le t_0 \le t$, any t_0-subset T_0 of $\{1, \ldots, n\}$ and any $\alpha_0 \in GF(2)^{t_0}$, $F(x)$ runs through each vector in $F(GF(2)^n)$ at least 2^{t-t_0} times while x runs through $S(n, T_0, \alpha)$ once.*

Corollary 5. *Let F be an (n,m,t)-extended immune S-box. For any integer $k \ge 1$, define an $(n+k) \times m$ S-box F^* as $F^*(\alpha, \beta) = F(\alpha)$ for each $\alpha \in GF(2)^n$ and $\beta \in GF(2)^k$. Then F^* is $(n+k, m, t)$-extended immune.*

6 Characterisation of Extended Resiliency (Immunity) in Terms of Existence of Unconditional (Conditional) Equations

In this section, we characterise the extended resiliency (immunity) in terms of the resistance against divide-and-conquer algebraic attacks by unconditional (conditional) equations. For completeness, we first give a new proof for a result from [15] about the existence of conditional algebraic equations. Then, we prove a new result about the existence of unconditional algebraic equations.

Lemma 8. *For a fixed t-subset $T = \{j_1, \ldots, j_t\} \subseteq \{1, \ldots, n\}$, there is no non-constant conditional algebraic equation over $x' = (x_{j_1}, \ldots, x_{j_t})$ induced from $F(x_1, \ldots, x_n) = \beta$ for any value of $\beta \in F(GF(2)^n)$ if and only if $F(S(n, T, x')) = F(GF(2)^n)$, for every $x' \in GF(2)^t$.*

Theorem 10. *There is no non-constant conditional algebraic equation over* $x' = (x_{j_1}, \ldots, x_{j_t})$ *induced from* $F(x_1, \ldots, x_n) = \beta$ *for any* t*-subset* $T = \{j_1, \ldots, j_t\} \subseteq \{1, \ldots, n\}$ *and any value of* $\beta \in F(GF(2)^n)$ *if and only if* F *is* (n, m, t)*-extended immune.*

Accordingly, for a (n, m, t)-extended immune S-box F, if the attackers want to establish a non-constant conditional algebraic equation induced from F, they have to choose x' with dimension higher than the extended immunity order defined in Definition 7.

Lemma 9. *For a fixed* t*-subset* $T = \{j_1, \ldots, j_t\} \subseteq \{1, \ldots, n\}$*, there is no non-constant unconditional algebraic equation over* $x' = (x_{j_1}, \ldots, x_{j_t})$ *and* y *induced from* $F(x_1, \ldots, x_n) = y$ *if and only if* $F(S(n, T, x')) = GF(2)^m$*, for every* $x' \in GF(2)^t$*.*

Theorem 11. *There is no non-constant unconditional algebraic equation over* $x' = (x_{j_1}, \ldots, x_{j_t})$ *and* y *induced from* $F(x_1, \ldots, x_n) = y$ *for any* t*-subset* $T = \{j_1, \ldots, j_t\} \subseteq \{1, \ldots, n\}$ *if and only if* F *is* (n, m, t)*-extended resilient.*

Therefore, for an (n, m, t)-extended resilient S-box F, if the attackers want to establish a non-constant unconditional algebraic equation induced from F, they have to choose x' with dimension higher than the extended resiliency order defined in Definition 4.

7 Relations Between Extended Immunity, Extended Resiliency and Classical Resiliency

Classically resilient S-boxes (see for instance [1, 2, 5, 6, 8, 16, 20, 21, 23, 24]) were studied previously. The following is the definition of classical resiliency.

Definition 8. *Let* F *be an* $n \times m$ *S-box. If for any* t*-subset* T *of* $\{1, \ldots, n\}$ *and any* $\alpha \in GF(2)^t$*,* $F(x)$ *runs through each vector in* $GF(2)^m$ *exactly* 2^{n-m-t} *times while* x *runs through* $S(n, T, \alpha)$ *once, then* F *is said to be* (n, m, t)-classically resilient*.*

Classical resiliency in Definition 8 was usually called resiliency. In this paper, we call it classical resiliency so as to avoid confusion. Summarily, classical resiliency is a special case of extended resiliency and extended resiliency is a special case of extended immunity. We next establish some relations among the three.

Proposition 5. *Any affine* (n, m, t)*-extended resilient S-box is also* (n, m, t)*-classically resilient.*

Theorem 12. *Let* F *be an affine* (n, m, t)*-extended immune S-box. Then* $\#F(GF(2)^n) = 2^k$*, where* k *is an integer, and there exist a vector* $\beta \in GF(2)^m$ *and an* $m \times k$ *matrix* B *over* $GF(2)$ *such that the mapping* $P(x) = (F(x) \oplus \beta)B$ *is an* (n, k, t)*-classically resilient S-box.*

According to Theorem 12, any affine extended immune S-box can be used to construct a classically resilient S-box by an appropriate affine transformation of the output.

Theorem 13. *Let Q be an (n, m, t)-extended immune S-box and let k be an integer satisfying $k = \lfloor \log_2 \#Q(GF(2)^n) \rfloor$, where $\lfloor r \rfloor$ denotes the maximum integer less than or equal to r. For any mapping P from $Q(GF(2)^n)$ onto $GF(2)^k$, define a mapping $F = P(Q(x))$ where $x \in GF(2)^n$. Then F is an (n, k, t)-extended resilient S-box.*

According to Theorem 13, any extended immune S-box can be transformed into an extended resilient S-box by an appropriate mapping of the output.

8 Algebraic Degree of Extended Resilient S-Boxes

The algebraic degree of $n \times m$ S-box $F = (f_1, \ldots, f_m)$, denoted by $deg(F)$, is defined as $deg(F) = \min_g \{deg(g) | g = \bigoplus_{j=1}^m c_j f_j, (c_1, \ldots, c_m) \neq (0, \ldots, 0)\}$. S-boxes with high algebraic degrees are desirable for resistance against algebraic attacks.

Lemma 10. *The algebraic degree of any $n \times m$ S-box F is at most $n - 1$ if $m \geq 2$.*

From [24], an $n \times m$ S-box is (n, m, t)-classically resilient if and only if each nonzero linear combination of its coordinate functions is $(n, 1, t)$-classically resilient. Due to [22], the algebraic degree of an $(n, 1, t)$-classically resilient function is less than $n - t$ unless $t = n - 1$ (Siegenthaler's inequality). Therefore, the algebraic degree of any (n, m, t)-classically resilient S-box is less than $n - t$ (when $m \geq 2$). We will show that unlike the classical resiliency order, the extended immunity (resiliency) order does not restrict the algebraic degree.

Notation 4. *Let β_j denote the vector in $GF(2)^n$ whose jth component is zero and all other components are one, and $\beta_0 = (1, \ldots, 1) \in GF(2)^n$. Let $(x_1 \cdots x_n)/x_j$ denote the product $x_1 \cdots x_{j-1} x_{j+1} \cdots x_n$ of $n - 1$ variables.*

Lemma 11. *Let f be a Boolean function on $GF(2)^n$. For any integer j, $1 \leq j \leq n$, let $f'(x_1, \ldots, x_n) = f(x_1, \ldots, x_n) \oplus (x_1 \cdots x_n)/x_j$. Then f' differs from f only for $x = \beta_0$ and $x = \beta_j$. If $deg(f) < n - 1$ then $deg(f') = n - 1$.*

Theorem 14. *Let $F = (f_1, \ldots, f_m)$ be an (n, m, t)-extended resilient S-box, $deg(f_j) < n - 1$, $j = 1, \ldots, m$, and $t > \lfloor \log_2(m + 1) \rfloor + 1$. Let $F' = (f'_1, \ldots, f'_m)$ be an $n \times m$ S-box, where $f'_j(x_1, \ldots, x_n) = f_j(x_1, \ldots, x_n) \oplus (x_1 \cdots x_n)/x_j$. Then F' is an (n, m, t_0)-extended resilient S-box such that $t_0 = t - \lfloor \log_2(m + 1) \rfloor - 1$ and $deg(F') = n - 1$.*

Therefore, an (n, m, t_0)-extended resilient S-box F' achieves the maximum degree $n - 1$. In contrast with F', if there exists an (n, m, t_0)-classically resilient

S-box, due to Siegenthaler's inequality, its algebraic degree is less than $n - t_0$ (when $m \geq 2$). Furthermore, there is another problem: it is unknown whether this upper bound on the algebraic degree of classically resilient S-boxes can be reached except for special cases. For these reasons, classically resilient S-boxes may not be desirable with respect to algebraic attacks.

In the proof of Theorem 15, we construct (n, m, t)-extended resilient S-boxes with maximum algebraic degree $n - 1$, for any given m and t, but the number of inputs, n, has to be sufficiently large.

Theorem 15. *For any given m and t and $r \geq t + \lfloor \log_2(m+1) \rfloor + 2$, there exists an (rm, m, t)-extended resilient S-box with algebraic degree $rm - 1$.*

9 Upper Bound on Extended Immunity (Resiliency)

Recall that the classical resiliency t of an (n, m, t)-resilient function is upper-bounded by $t \leq \lfloor \frac{2^{m-1}n}{2^m-1} \rfloor - 1$ [14] and $t \leq 2\lfloor \frac{2^{m-2}(n+1)}{2^m-1} \rfloor - 1$ [2]. In this section, we indicate that the upper bound on extended immunity (resiliency) order is different from the bound on the classical resiliency order. According to Proposition 3, any (n, m, t)-extended immune S-box F satisfies $t \leq n - \log_2 \#F(GF(2)^n)$. We next show that this upper bound is tight for large t. We first provide a new proof of a result from [15] concerning Boolean functions, i.e., $m = 1$. Then we generalise this result to an arbitrary m.

Lemma 12. *Let f be a Boolean function on $GF(2)^n$ with an extended immunity order t. Then (i) $t = n$ if and only if f is constant and (ii) $t = n - 1$ if and only if $f(x) = x_1 \oplus \cdots \oplus x_n \oplus c$, where $c \in GF(2)$ is constant.*

Theorem 16. *Let $F = (f_1, \ldots, f_m)$ be an $n \times m$ S-box with an extended immunity order t. Then (i) $t = n$ if and only if F is constant and (ii) $t = n - 1$ if and only if $f_j(x) = x_1 \oplus \cdots \oplus x_n \oplus c_j$ or c_j, where $c_j \in GF(2)$ is constant, $j = 1, \ldots, m$, and there exists a value $j = j_0$ such that $f_{j_0} = x_1 \oplus \cdots \oplus x_n \oplus c_{j_0}$. (iii) For both $t = n$ and $t = n - 1$, the upper bound $t \leq n - \log_2 \#F(GF(2)^n)$ holds with equality.*

According to Theorem 16, the extended immunity of non-constant $n \times m$ S-boxes can be higher than classical resiliency. However, except for $t \leq n - m$, we do not know any other upper bound on the extended resiliency t of (n, m, t)-extended resilient S-boxes. This is an interesting problem to be studied in the future.

10 Conclusions and Future Work

In this paper, we provided different mathematical characterisations of the extended immunity and its special case - the extended resiliency. A characterisation of the extended resiliency (immunity) in terms of the existence of unconditional (conditional) equations is also provided. Relations between the extended immunity, extended resiliency and classical resiliency are examined. Constructions

showing that the extended resiliency does not restrict the algebraic degree if the number of inputs is sufficiently large are given too. More efficient constructions and the nonlinearity of extended resilient (immune) S-boxes will be studied in future work. It is also interesting to derive other, possibly sharper bounds on the extended resiliency and extended immunity. Other criteria related to algebraic attacks, such as the minimum degree of algebraic equations induced from S-boxes, in the conditional or unconditional scenarios, can also be taken into consideration.

Acknowledgment

The first two authors were supported by Australian Research Council grants DP0345366 and DP0451484.

References

1. C. H. Bennett, G. Brassard, and J. M. Robert. Privacy amplification by public discussion. *SIAM J. Computing*, 17:210–229, 1988.
2. J. Bierbrauer, K. Gopalakrishnan, and D. R. Stinson. Bounds on resilient functions and orthogonal arrays. In *Advances in Cryptology - CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 247–256. Springer-Verlag, Berlin, Heidelberg, New York, 1994.
3. P. Camion, C. Carlet, P. Charpin, and N. Sendrier. On correlation-immune functions. In *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 87–100. Springer-Verlag, Berlin, Heidelberg, New York, 1991.
4. C. Carlet. Improving the algebraic immunity of resilient and nonlinear functions and constructing bent functions. (`http://eprint.iacr.org/2004/276/`), 2004.
5. C. Carlet and E. Prouff. Vectorial functions and covering sequences. In *Finite Fields and Applications*, volume 2948 of *Lecture Notes in Computer Science*, pages 215–248. Springer-Verlag, Berlin, Heidelberg, New York, 2000.
6. J. H. Cheon. Nonlinear vector resilient functions. In *Advances in Cryptology - CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 458–469. Springer-Verlag, Berlin, Heidelberg, New York, 2001.
7. J. H. Cheon and D. H. Lee. Resistance of S-boxes against algebraic attacks. In *Proceedings of Fast Software Encryption '04*, volume 3017 of *Lecture Notes in Computer Science*, pages 83–94. Springer-Verlag, Berlin, Heidelberg, New York, 2004.
8. B. Chor, O. Goldreich, J. Håstad, J. Friedman, S. Rudich, and R. Smolensky. The bit extraction problem or t-resilient functions. In *Proc. 26th IEEE Symp. on Foundations of Computer Science*, pages 396–407, 1985.
9. N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology - CRYPTO '03*, volume 2729 of *Lecture Notes in Computer Science*, pages 176–194. Springer-Verlag, Berlin, Heidelberg, New York, 2003.
10. N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology - EUROCRYPT '03*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer-Verlag, Berlin, Heidelberg, New York, 2003.

11. N. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *Advances in Cryptology - ASIACRYPT '02*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer-Verlag, Berlin, Heidelberg, New York, 2002.
12. D. Dalai, K. Gupta, and S. Maitra. Results on algebraic immunity for cryptographically significant Boolean functions. In *Proceedings of INDOCRYPT '04*, volume 3348 of *Lecture Notes in Computer Science*, pages 92–106. Springer-Verlag, Berlin, Heidelberg, New York, 2004.
13. D. Dalai, K. Gupta, and S. Maitra. Results on algebraic immunity for cryptographically significant Boolean functions: Construction and analysis in term of algebraic immunity. In *Proceedings of Fast Software Encryption '05*, volume 3557 of *Lecture Notes in Computer Science*, pages 98–111. Springer-Verlag, Berlin, Heidelberg, New York, 2005.
14. J. Friedman. On the bit extraction problem. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 314–319, 1992.
15. J. Dj. Golić. Vectorial Boolean functions and induced algebraic equations. (`http://eprint.iacr.org/2004/225/`), 2004.
16. K. C. Gupta and P. Sarkar. Improved construction of nonlinear resilient s-boxes. In *Advances in Cryptology - ASIACRYPT '02*, volume 2501 of *Lecture Notes in Computer Science*, pages 466–483. Springer-Verlag, Berlin, Heidelberg, New York, 2002.
17. M. Hall, Jr. *Combinatorial Theory*. Ginn-Blaisdell, Waltham, 1967.
18. W. Meier, E. Pasalic, and C. Carlet. Algebraic attacks and decomposition of Boolean functions. In *Advances in Cryptology - EUROCRYPT '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 474–491. Springer-Verlag, Berlin, Heidelberg, New York, 2004.
19. M. O'Nan. *Linear Algebra*. Harcourt Brace Jovanovich, New York, 1976.
20. E. Pasalic and S. Maitra. Further constructions of resilient Boolean functions with very high nonlinearity. *IEEE Transactions on Information Theory*, 48(7):1825–1834, 2002.
21. P. Sarkar and S. Maitra. Nonlinearity bounds and constructions of resilient Boolean functions. In *Advances in Cryptology - CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 515–532. Springer-Verlag, Berlin, Heidelberg, New York, 2000.
22. T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, 30(5):776–779, 1984.
23. D. R. Stinson. Resilient functions and large sets of orthogonal arrays. *Congressus Numerantium*, 92:105–110, 1993.
24. X. M. Zhang and Y. Zheng. Cryptographically resilient functions. *IEEE Transactions on Information Theory*, 43(5):1740–1747, 1997.

Appendix: Proofs of Mathematical Results

Proof of Lemma 1. From the structure of D_F, the ith row vector l_i of D_F is the sequence of linear function $\psi(y) = \langle F(\alpha_i), y\rangle$. On the other hand, the $j$the column of H_m is ℓ_j^T - the sequence of a linear function $\psi_j(y) = \langle \gamma_j, y\rangle$. Note that the sequences of different linear functions are orthogonal. Thus, l_i and ℓ_j

will be orthogonal if $F(\alpha_i) \neq \gamma_j$, while l_i and ℓ_j will be identical if $F(\alpha_i) = \gamma_j$. According to the definition of C_F, this proves the lemma. □

Proof of Lemma 2. Due to Lemma 1, we have $H_m D_F^T D_F H_m = 2^{2m} C_F^T C_F$. Due to the definition of C_F, each row of C_F has precisely one nonzero entry. Thus any two columns of C_F are orthogonal. On the other hand, the number of ones in the jth column is equal to the number of times that F takes γ_j. Thus $C_F^T C_F = diag(\lambda_0, \lambda_1, \ldots, \lambda_{2^m-1})$. Summarising the above, we have proved $H_m D_F^T D_F H_m = 2^{2m} diag(\lambda_0, \lambda_1, \ldots, \lambda_{2^m-1})$. Since H_m is a $2^m \times 2^m$ Hadamard matrix, it follows that $D_F^T D_F = H_m diag(\lambda_0, \lambda_1, \ldots, \lambda_{2^m-1}) H_m$. □

Proof of Corollary 1. Since H_m is a $2^m \times 2^m$ Hadamard matrix, due to Lemma 2, we have $D_F^T D_F H_m = 2^m H_m diag(\lambda_0, \lambda_1, \ldots, \lambda_{2^m-1})$ or, equivalently, $D_F^T D_F \ell_j^T = 2^m \lambda_j \ell_j^T$, $j = 0, 1, \ldots, 2^m - 1$. □

Proof of Lemma 3. According to Corollary 1, (i) and (ii) are equivalent. According to the well known fact from linear algebra, (ii) holds if and only if the rank of $D_F^T D_F$ is 2^m. Further, $D_F^T D_F$ and D_F have the same rank. This proves the equivalence of (ii) and (iii). □

Proof of Lemma 4. Assume that (i) holds. Consequently, $P(F(GF(2)^n))= P(GF(2)^m)= GF(2)^k$. This proves (i) $\Longrightarrow$ (ii). Assume that (ii) holds. Clearly, any non-constant function h on $GF(2)^m$ is a surjective $m \times 1$ S-box. Due to (ii), $h(F)$ is a surjective $n \times 1$ S-box and then non-constant. By virtue of Definition 2, we thus proved (ii) $\Longrightarrow$ (iii). Assume that (iii) holds. We now prove (i) by contradiction. Assume that F does not take a value $\beta \in GF(2)^m$. Define a non-constant function h on $GF(2)^m$ as $h(y) = 1$ if and only if $y = \beta$. Clearly, $h(F(x))$ is the constant zero function. This contradicts (iii), so that (i) is true. This proves (iii) $\Longrightarrow$ (i). □

Proof of Proposition 2. Fix a t-subset T of $\{1, \ldots, n\}$. The 2^t sets $S(n, T, \alpha)$, $\alpha \in GF(2)^t$, are disjoint and partition $GF(2)^n$. When x runs once through $S(n, T, \alpha)$ for any fixed $\alpha \in GF(2)^t$, $F(x)$ runs at least once through each vector in $GF(2)^m$. Accordingly, $F(x)$ runs through each vector in $GF(2)^m$ at least 2^t times when x runs once through $GF(2)^n$. □

Proof of Theorem 1. Let a column vector ζ be the sequence of a Boolean function g on $GF(2)^n$. It is easy to verify that $diag(b_0, b_1, \ldots, b_{2^n-1})\zeta$ is the sequence of the restriction $g(x_1, \ldots, x_n)|_{x_{j_1}=a_1,\ldots,x_{j_t}=a_t}$. For this reason, the matrix, from Notation 1, corresponding to the restriction $F(x_1, \ldots, x_n)|_{x_{j_1}=a_1,\ldots,x_{j_t}=a_t}$ is identical with $diag(b_0, b_1, \ldots, b_{2^n-1}) D_F$. The theorem then follows from Lemma 3. □

Proof of Theorem 3. We only need to prove the theorem for $r = t$ because a function g on $GF(2)^r$ with $1 \leq r < t$ can be regarded as a function on $GF(2)^t$ that does not depend on some $t - r$ variables. Assume that F is (n, m, t)-extended resilient. For any fixed t-subset $T = \{j_1, \ldots, j_t\}$ of $\{1, \ldots, n\}$

and any given nonzero Boolean function g on $GF(2)^t$, there exists a vector $\alpha \in GF(2)^t$ satisfying $g(\alpha) = 1$. Then $\{g(x_{j_1}, \ldots, x_{j_t})F(x)|x \in S(n,T,\alpha)\}$ $=F(S(n,T,\alpha))$. Since F is (n,m,t)-extended resilient, $F(S(n,T,\alpha)) = GF(2)^m$. Thus $g(x_{j_1}, \ldots, x_{j_t})F(x)$ is surjective. Conversely, assume that F satisfies the property from the theorem. We now prove that F is (n,m,t)-extended resilient. For a given t-subset $T = \{j_1, \ldots, j_t\}$ of $\{1, \ldots, n\}$ and any given $\alpha = (a_1, \ldots, a_t)$ $\in GF(2)^t$, we define a nonzero Boolean function g on $GF(2)^t$ such that $g(y) = 1$ if and only if $y = \alpha$. Therefore, $\{g(x_{j_1}, \ldots, x_{j_t})F(x)|x \in GF(2)^n\}$ $=\{g(x_{j_1}, \ldots, x_{j_t})F(x)|x \in S(n,T,\alpha)\}=$ $F(S(n,T,\alpha))$. Due to the assumption, $\{g(x_{j_1}, \ldots, x_{j_t})F(x)|x \in GF(2)^n\} = GF(2)^m$. Hence $F(S(n,T,\alpha)) = GF(2)^m$. Since both T with $\#T = t$ and $\alpha \in GF(2)^t$ are arbitrary, we have proved that F is (n,m,t)-extended resilient. □

Proof of Theorem 4. By Definition 3, it is easy to see that (i) $\Longrightarrow$ (ii). Due to Proposition 2, (ii) $\Longrightarrow$ (iii). Assume that (iii) holds. We let $t_0 = t$. Then it follows that F is (n,m,t)-extended resilient. This proves (iii) $\Longrightarrow$ (i). □

Proof of Theorem 7. Assume that (i) holds. For any subset $S(n,T,\alpha)$ of $GF(2)^n$ with $\#T = t$, since F is (n,k,t)-extended immune, $P(F(S(n,T,\alpha)))$ $= P(F(GF(2)^n))$. Thus $P(F(x))$ is (n,k,t)-extended immune. We have thus proved (i) $\Longrightarrow$ (ii). Assume now that (ii) holds. Let h be a function on $GF(2)^m$ such that $h(F)$ is non-constant, i.e., $\#h(F(GF(2)^n)) = 2$. As, due to (ii), $h(F)$ is $(n,1,t)$-extended immune, for any subset $S(n,T,\alpha)$ of $GF(2)^n$ with $\#T = t$ and $\alpha \in GF(2)^t$, we have $\#h(F(S(n,T,\alpha))=\#h(F(GF(2)^n)) = 2$. This means that $h(G)$ is non-constant, where G is defined in the Theorem. We have thus proved that (ii) $\Longrightarrow$ (iii). Finally, assume that (iii) holds. We prove (i) by contradiction. Assume that (i) does not hold. Then there exists a subset $S(n,T,\alpha)$ of $GF(2)^n$ with $\#T = t$ and $\alpha \in GF(2)^t$ such that $F(S(n,T,\alpha)) \neq F(GF(2)^n)$. This implies that F is non-constant. Let $\beta \in F(GF(2)^n) \setminus F(S(n,T,\alpha))$. We choose a non-constant function h on $GF(2)^m$ such that $h(y) = 1$ if and only if $y = \beta$. Since F takes value β and F is non-constant, from the definition of h, it follows that $h(F)$ takes both values 1 and 0. However, since $\beta \notin F(S(n,T,\alpha))$, $h(G)$ is the zero function, where G is defined in the Theorem. This contradicts (iii), so that (i) is true. Thus we have proved (iii) $\Longrightarrow$ (i). □

Proof of Lemma 8. For a fixed t-subset T and any given value of $\beta \in F(GF(2)^n)$, due to (4), there exists a non-constant algebraic equation over x' induced from $F(x) = \beta$ if and only if $\#W(F,T,\beta) < 2^t$. Consequently, there is no non-constant algebraic equation over x' induced from $F(x) = \beta$ if and only if $\#W(F,T,\beta) = 2^t$, or equivalently, for each $x' \in GF(2)^t$, there exists $x'' \in GF(2)^{n-t}$ such that $F(x) = \beta$. Since this is true for an arbitrary $\beta \in F(GF(2)^n)$, it then follows that there is no non-constant algebraic equation over x' induced from $F(x) = \beta$ for any $\beta \in F(GF(2)^n)$ if and only if $F(S(n,T,x')) = F(GF(2)^n)$, for every $x' \in GF(2)^t$. □

Proof of Theorem 10. As a t-subset T is arbitrary, the theorem directly follows from Lemma 8 and Definition 6. □

Proof of Lemma 9. For a fixed t-subset T, due to (2), there is no non-constant algebraic equation over x' and y induced from $F(x) = y$ if and only if $\#W(F,T) = 2^{t+m}$, or equivalently, for each $x' \in GF(2)^t$, $\#F(S(n,T,x')) = 2^m$, that is, $F(S(n,T,x')) = GF(2)^m$, for every $x' \in GF(2)^t$. □

Proof of Theorem 11. As a t-subset T is arbitrary, the theorem directly follows from Lemma 9 and Definition 3. □

Proof of Proposition 5. Let F be an affine (n,m,t)-extended resilient S-box. Let $T = \{j_1, \ldots, j_t\}$ be a subset of $\{1, \ldots, n\}$ and $\alpha = (a_1, \ldots, a_t) \in GF(2)^t$. Due to Theorem 2, all the coordinate functions of $F(x)|_{x_{j_1}=a_1,\ldots,x_{j_t}=a_t}$ are functionally independent and then also linearly independent. Since $F(x)|_{x_{j_1}=a_1,\ldots,x_{j_t}=a_t}$ is affine, due to linear algebra, $F(x)|_{x_{j_1}=a_1,\ldots,x_{j_t}=a_t}$ runs through each vector in $GF(2)^m$ exactly 2^{n-m-t} times while x runs through $S(n,T,\alpha)$ once. This proves that F is (n,m,t)-classically resilient. □

Proof of Theorem 12. Since F is affine, there exists a vector $\beta \in GF(2)^m$ such that $F(x) \oplus \beta$ is linear. By linear algebra, $\#F(GF(2)^n) = 2^k$ for an integer k and $F(GF(2)^n) \oplus \beta$ is a k-dimensional subspace U of $GF(2)^m$. Therefore, there exists an $m \times k$ matrix B over $GF(2)$ such that $\{\gamma B | \gamma \in U\}$ is identical with $GF(2)^k$. Set $P(x) = (F(x) \oplus \beta)B$. According to Theorem 7, P is an (n,k,t)-extended resilient S-box. On the other hand, $P(GF(2)^n) = \{\gamma B | \gamma \in U\} = GF(2)^k$. Thus P is a linear (n,k,t)-extended resilient S-box. According to Proposition 5, P is an (n,k,t)-classically resilient S-box. □

Proof of Theorem 13. Clearly, the condition $\#Q(GF(2)^n) \geq 2^k$ guarantees the existence of a mapping from $Q(GF(2)^n)$ onto $GF(2)^k$. Accordingly, if P is such a mapping, then $P(Q(GF(2)^n)) = GF(2)^k$. For any t-subset T of $\{1, \ldots, n\}$ and any $\alpha \in GF(2)^t$, since Q is (n,m,t)-extended immune, $Q(S(n,T,\alpha)) = Q(GF(2)^n)$. As P is a mapping from $Q(GF(2)^n)$ onto $GF(2)^k$, $P(Q(S(n,T,\alpha))) = P(Q(GF(2)^n)) = GF(2)^k$. □

Proof of Lemma 10. We prove the lemma by contradiction. Assume for contradiction that there exists an $n \times m$ S-box $F = (f_1, \ldots, f_m)$ with $m \geq 2$ such that $deg(F) = n$. Then we have $deg(f_1) = deg(f_2) = n$ and hence both f_1 and f_2 contain the product term $x_1 \cdots x_n$. This term cancels out in $f_1 \oplus f_2$ and therefore $deg(f_1 \oplus f_2) < n$. Further, by definition, this implies that $deg(F) < n$, so that we have a contradiction. □

Proof of Lemma 11. The first part is straightforward to verify. Secondly, if $deg(f) < n-1$, then the term $(x_1 \cdots x_n)/x_j$ remains in the algebraic normal form of f' and hence $deg(f') = n-1$. □

Proof of Theorem 14. Due to Lemma 11, each f'_j contains exactly one term of degree $n-1$, i.e., $(x_1 \cdots x_n)/x_j$. Since $m \leq n$ and $\{(x_1 \cdots x_n)/x_j \mid 1 \leq j \leq m\}$ are linearly independent Boolean functions, any nonzero linear combination of $f'_1, \ldots, f'_m$ must contain a nonzero linear combination of $(x_1 \cdots x_n)/x_j$, j

$= 1, \ldots, m$, that cannot be eliminated. This implies that $deg(F') = n-1$. For any fixed subset $T_0 = \{j_1, \ldots, j_{t_0}\}$ of $\{1, \ldots, n\}$ and any fixed vector $\alpha_0 \in GF(2)^{t_0}$, due to Theorem 4, $F(x)$ runs through each vector in $GF(2)^m$ at least 2^{t-t_0} times while x runs through $S(n, T_0, \alpha_0)$ once. According to Lemma 11, F' is obtained from F by changing exactly $m+1$ of its values, $F(\beta_j)$, $j = 1, \ldots, m$, and $F(\beta^*)$. Since $t-t_0 = \lfloor \log_2(m+1) \rfloor + 1$, we have $t - t_0 > \log_2(m+1)$. Thus $2^{t-t_0} - (m+1) > 0$ and hence $F'(x)$ runs through each vector in $GF(2)^m$ at least once while x runs through $S(n, T_0, \alpha_0)$ once, or in other words, $F'(S(n, T_0, \alpha_0)) = GF(2)^m$. □

Proof of Theorem 15. Let P be a permutation on $GF(2)^m$ and let $r \geq 1$. Let $F(x) = P(z_1) \oplus \cdots \oplus P(z_r)$, where $z_1, \ldots, z_r \in GF(2)^m$ and $x = (z_1, \ldots, z_r) \in GF(2)^{rm}$. We first prove that F is $(rm, m, r-1)$-extended resilient with $deg(F) \leq m$. We note that F is a surjective $rm \times m$ S-box. Rewrite x as $x = (x_1, \ldots, x_{rm})$. Choose any subset $T = \{j_1, \ldots, j_{r-1}\}$ of $\{1, \ldots, rm\}$ and any $\alpha \in GF(2)^{r-1}$. Then there must exist an index i, $1 \leq i \leq r$, such that the sets of variables in z_i and $(x_{j_1}, \ldots, x_{j_{r-1}})$ are disjoint. Since P is a permutation on $GF(2)^m$, then we have $GF(2)^m \supseteq F(S(rm, T, \alpha)) \supseteq P(GF(2)^m) = GF(2)^m$. This proves that F is $(rm, m, r-1)$-extended resilient. Due to the construction, the algebraic degree of F cannot exceed m.

In particular, we choose r satisfying $r - 1 \geq t + \lfloor \log_2(m+1) \rfloor + 1$. Let F' be an $rm \times m$ S-box obtained from F as in Theorem 14. Then according to Theorem 14, $deg(F') = rm - 1$ and F' is (rm, m, t_0)-extended resilient with $t_0 = (r-1) - \lfloor \log_2(m+1) \rfloor - 1 \geq t$. By virtue of Lemma 6, F' is then also (rm, m, t)-extended resilient. □

Proof of Lemma 12. If f is constant, then $t = n$ by the definition of extended immunity. If $t = n$, then the upper bound $t \leq n - \log_2 \#f(GF(2)^n)$ implies that $\#f(GF(2)^n) = 1$, which means that f is constant. This proves (i).

As for (ii), to prove the sufficiency, note that the restriction of f to any set $S(n, T_i, \alpha)$, where $T_i = \{1, \ldots, n\} \setminus \{i\}$ and $\alpha \in GF(2)^{n-1}$, is a non-constant affine function of the remaining variable x_i, so that $f(S(n, T_i, \alpha)) = GF(2)$. This means that $t \geq n-1$. However, as f is non-constant, (i) implies that $t = n-1$.

To prove the necessity in (ii), assume that $t = n-1$. Then, firstly, from (i) it follows that $f(GF(2)^n) = GF(2)$. For any fixed i, $1 \leq i \leq n$, f can be expressed as $f(x) = x_i g(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \oplus h(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ where both g and h are Boolean functions on $GF(2)^{n-1}$. We next prove that g is the constant one by contradiction. Assume that there exists some vectotr $\alpha \in GF(2)^{n-1}$ such that $g(\alpha) = 0$. Since h does not depend on x_i, we have $\#f(S(n, T_i, \alpha)) = 1$, where T_i is as above. This contradicts the fact that $f(S(n, T_i, \alpha)) = f(GF(2)^n) = GF(2)$ and hence proves that g is the constant one. Thus $f(x) = x_i \oplus h(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$, which means that the algebraic normal form of f contains the linear term x_i and x_i appears as a linear term only. Since this holds for each $i \in \{1, \ldots, n\}$, f can be expressed as $f(x) = x_1 \oplus \cdots \oplus x_n \oplus c$, where $c \in GF(2)$ is constant. □

Proof of Theorem 16. The claim (i) is proved in the same way as for $m = 1$ in Lemma 12.

As for (ii), to prove the sufficiency, assume that F has the form specified. Since f_j is either constant or identical to l or $l \oplus 1$, where $l(x) = x_1 \oplus \cdots \oplus x_n$, and f_{j_0} is not constant, it follows that, for each $x \in GF(2)^n$, the value of $f_{j_0}(x)$ uniquely determines the values of the remaining functions $f_j(x)$. Therefore, $\#F(GF(2)^n) = 2$. By the same argument, we also obtain that $\#F(S(n, T, \alpha)) = 2$, for any $(n-1)$-subset T of $\{1, \ldots, n\}$, because the restriction of f_{j_0} to $S(n, T, \alpha)$ is a non-constant affine function of x_i. Consequently, $F(S(n, T, \alpha)) = F(GF(2)^n)$ and $\#F(GF(2)^n) = 2$. Hence, in view of (i), we get $t = n - 1$.

To prove the necessity in (ii), assume that $t = n-1$. In view of (i), this means that F is a non-constant $(n, m, n-1)$-extended immune function. Corollary 4 then implies that each f_j is $(n, 1, n-1)$-extended immune, i.e., has an extended immunity order $n-1$ or n. According to Lemma 12, each f_j has the form $f_j = x_1 \oplus \cdots \oplus x_n \oplus c_j$ or c_j, where $c_j \in GF(2)$ is constant. Since F is non-constant, there must exist a value $j = j_0$ such that f_{j_0} is non-constant.

Finally, as for (iii), (i) and (ii) imply that for $t = n$ and $t = n - 1$, we have $\#F(GF(2)^n) = 1$ and $\#F(GF(2)^n) = 2$, respectively, so that the upper bound holds with equality in both cases. □

Integral Cryptanalysis of Reduced FOX Block Cipher

Wenling Wu[1], Wentao Zhang[2], and Dengguo Feng[1]

[1] State Key Laboratory of Information Security,
Institute of Software, Chinese Academy of Sciences,
Beijing 100080, P.R. China
{wwl, feng}@is.iscas.ac.cn
[2] State Key Laboratory of Information Security,
Graduate University of Chinese Academy of Sciences,
Beijing 100049, P.R. China
zhangwt@gscas.ac.cn

Abstract. FOX is a family of block ciphers presented recently, which is based upon some results of provable security and has high performances on various platforms. In this paper, we construct some distinguishers between 3-round FOX and a random permutation of the blocks space. By using integral attack and collision-searching techniques, the distinguishers are used to attack 4, 5, 6 and 7-round FOX64, 4 and 5-round FOX128. The attack is more efficient than previous integral attacks on FOX. The complexity of improved integral attack is $2^{77.6}$ on 4-round FOX128, $2^{205.6}$ against 5-round FOX128 respectively. For FOX64, the complexity of improved integral attack is $2^{45.4}$ on 4-round FOX64, $2^{109.4}$ against 5-round FOX64, $2^{173.4}$ against 6-round FOX64, $2^{237.4}$ against 7-round FOX64 respectively. Therefore, 4-round FOX64/64, 5-round FOX64/128, 6-round FOX64/192, 7-round FOX64/256 and 5-round FOX128/256 are not immune to the attack in this paper.

1 Introduction

FOX [9] is a new family of block ciphers, which is designed upon the request of the MediaCrypt AG company. The high level of FOX adopts a modified structure of Lai-Massey Scheme [14], which has been proven to have good pseudorandomness properties in the Luby-Rackoff paradigm [15] and decorrelation inheritance properties proposed by Vaudenay [19]. The round function of FOX uses SPS (Substitution-Permutation-Substitution) structure with three layers of subkey addition, SPS structure has already been proven to have powerful ability to resist differential and linear cryptanalysis. The design rationale of diffusion primitives in FOX is presented in [10]. The key schedule of FOX is very complex compared with other existing block ciphers, each subkey of FOX is related to the seed key and it's very difficult to acquire information about seed key or other subkeys from some certain subkeys. The complex key

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 229–241, 2006.
© Springer-Verlag Berlin Heidelberg 2006

schedule, high-level structure with provable security and powerful round function make FOX appear to be a strong block cipher. Since FOX is a new cipher published last year, all we know about its security analysis are limited to the designer's results and the integral cryptanalysis presented in [17]. The security of FOX against differential and linear cryptanalysis is easy to estimate for the good property of its S-box, SPS transformation and high level structures. The designers also analyze the security of FOX against differential-linear cryptanalysis [1,16], boomerang [20] and rectangle attacks [2], truncated and higher-order differentials [12], impossible differentials [3], partitioning cryptanalysis [7] and interpolation attack [11], algebraic attack [6,18], slide attack [4,5] and related-cipher attacks [21]. Integral attack [13] is one of the most effective attack method against AES, which had been used to analyze the security of other ciphers [8,23]. It's pointed in [9] that integral attack has a complexity of 2^{72} encryptions against 4-round FOX64, 2^{136} against 5-round FOX64, 2^{200} against 6-round FOX64. The authors also claimed that integral attack has a complexity of 2^{136} encryptions against 4-round FOX128, and 5-round FOX128 is immune to integral attack. In this paper, we combine collision technique [22] and integral attack to analyze the security of FOX. The improved integral attack on FOX is more efficient than known integral attacks. The improved integral attack has a complexity of $2^{77.6}$ encryptions against 4-round FOX128, $2^{205.6}$ against 5-round FOX128 respectively. For FOX64, the improved integral attack has a complexity of $2^{45.4}$ encryptions against 4-round FOX64, $2^{109.4}$ against 5-round FOX64, $2^{173.4}$ against 6-round FOX64, $2^{237.4}$ against 7-round FOX64 respectively.

This paper is organized as follows: Section 2 briefly introduces the structure of FOX128. 3-round distinguishers are presented in Section 3. In Section 4, we show how to use the 3-round distinguishers to attack 4 and 5 rounds of FOX128. In Section 5, we briefly introduce the attacks on 4, 5, 6 and 7 rounds of FOX64. and Section 6 concludes the paper.

2 Description of FOX

FOX has two versions, both have a variable number of rounds which depends on keysize: the first one FOX64/k/r has a 64-bit blocksize with a variable key length which is a multiple of 8 and up to 256 bits. The second one FOX128/k/r uses a 128-bit blocksize with the same possible key lengths. For FOX64 with k=128 and FOX128 with k=256, the designers advise that round number are both 16. Due to space limitation, we only introduce FOX128 briefly. Details are shown in [9].

2.1 Round Function f64

The round function **f64** consists of three main parts: a *substitution* part denoted **sigma8**, a *diffusion* part denoted **MU8**, and a *round key addition* part (see Fig.1). Formally, the i-th round function $f64^i$ takes a 64-bit input

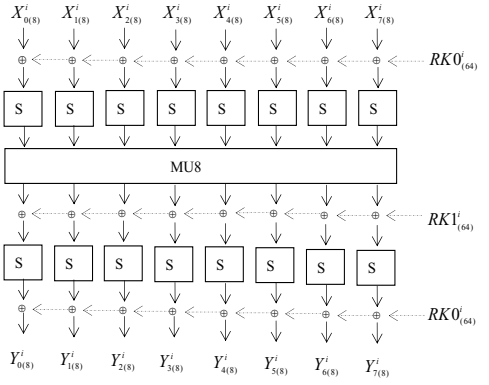


Fig. 1. The i-th round function

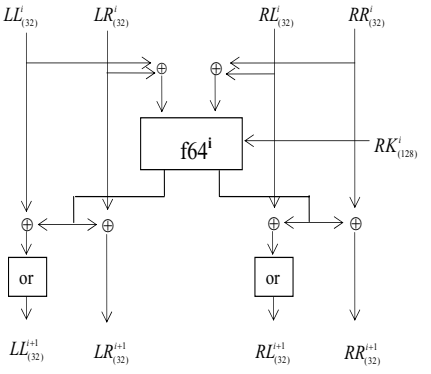


Fig. 2. The i-th round transformation

$X^i_{(64)} = X^i_{0(8)}||X^i_{1(8)}||\cdots||X^i_{7(8)}$, a 128-bit round key $RK^i_{(128)} = RK0^i_{(64)}||RK1^i_{(64)}$ and returns

$$Y^i_{(64)} = Y^i_{0(8)}||\cdots||Y^i_{7(8)} = sigma8(MU8(sigma8(X^i_{(64)} \oplus RK0^i_{(64)}))$$

$$\oplus RK1^i_{(64)}) \oplus RK0^i_{(64)}.$$

The mapping **sigma8** consists of 8 parallel computations of a non-linear bijective mapping (see Ref.[9]). **MU8** considers an input $(Z_{0(8)}||\cdots||Z_{7(8)})$ as a vector $(Z_{0(8)}||\cdots||Z_{7(8)})^T$ over $GF(2^8)$ and multiply it with a matrix to obtain an output vector of the same size. The matrix is the following:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & a \\ 1 & a & b & c & d & e & f & 1 \\ a & b & c & d & e & f & 1 & 1 \\ b & c & d & e & f & 1 & a & 1 \\ c & d & e & f & 1 & a & b & 1 \\ d & e & f & 1 & a & b & c & 1 \\ e & f & 1 & a & b & c & d & 1 \\ f & 1 & a & b & c & d & e & 1 \end{pmatrix}$$

where $a = \alpha + 1$, $b = \alpha^7 + \alpha$, $c = \alpha$, $d = \alpha^2$, $e = \alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2$ and $f = \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha$. α is a root of the irreducible polynomial $m(x) = x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + 1$.

2.2 Encryption of FOX128

FOX128 is the 15-times iteration of *round transformation* **elmor128**, followed by the application of last round transformation called **elmid128**. **elmor128** (illustrated in Fig.2) is built as an Extended Lai-Massey scheme combined with

two transformations ***or***. The i-th round transformation—**elmor128** transforms a 128-bit input $LL^i_{(32)}||LR^i_{(32)}||RL^i_{(32)}||RR^i_{(32)}$ and a 128-bit round key $RK^i_{(128)}$ in a 128-bit output $LL^{i+1}_{(32)}||LR^{i+1}_{(32)}||RL^{i+1}_{(32)}||RR^{i+1}_{(32)}$. Let $LL^i_{(32)} \oplus LR^i_{(32)}||RL^i_{(32)} \oplus RR^i_{(32)} = X^i_{(64)} = X^i_{0(8)}||X^i_{1(8)}||\cdots||X^i_{7(8)})$ and $f64^i(X^i_{(64)}, RK^i_{(128)}) = \phi_L||\phi_R$. Then,

$$LL^{i+1}_{(32)}||LR^{i+1}_{(32)}||RL^{i+1}_{(32)}||RR^{i+1}_{(32)} = or(LL^i_{(32)} \oplus \phi_L)||LR^i_{(32)} \oplus \phi_L||or$$

$$(RL^i_{(32)} \oplus \phi_R)||RR^i_{(32)} \oplus \phi_R.$$

The **elmid128** function is a slightly modified version of **elmor128**, namely the two transformations ***or*** are replaced by two identity transformations. The transformation ***or*** is a function taking a 32-bit input $X_{(32)} = X_{0(16)}||X_{1(16)}$ and returning a 32-bit output $Y_{(32)} = Y_{0(16)}||Y_{1(16)}$ which is in fact a one-round Feistel scheme with the identity function as round function; it is defined as $Y_{0(16)}||Y_{1(16)} = X_{1(16)}||(X_{0(16)} \oplus X_{1(16)})$. The encryption C_{128} by FOX128 of a 128-bit plaintext P_{128} is defined as:

$$C_{128} = elmid128(elmor128(\ldots elmor128(P_{128}, RK^1_{(128)}), \ldots, RK^{15}_{(128)})RK^{16}_{(128)}).$$

where $RK^1_{(128)}, \ldots, RK^{16}_{(128)}$ are round subkeys produced by the key schedule algorithm from the user key. In this paper, subkeys are assumed to be independent of each other.

3 3-Round Distinguishers

Choose plaintexts $P_{(128)} = LL^1_{(32)}||LR^1_{(32)}||RL^1_{(32)}||RR^1_{(32)}$ as follows:

$$LL^1_{(32)} = LR^1_{(32)} = c||c||c||c, \qquad RL^1_{(32)} = RR^1_{(32)} = c||c||c||x.$$

where x take values in $\{0,1\}^8$, c is a constant in $\{0,1\}^8$. Thus, the input of the first round function $f64^1$ is $X^1_{(64)} = 0||0\ldots||0$. Let $f64^1(0||0\ldots||0) = (a_0||a_1\ldots a_7)$, where $a_i(0 \le i \le 7)$ are entirely determined by round subkey $RK^1_{(128)}$, so $a_i(0 \le i \le 7)$ are constants when the user key is fixed. Then the output of the 1st round transformation can be represented as follows:

$$\begin{aligned}
LL^2_{(32)} &= a_2 \oplus c||a_3 \oplus c||a_0 \oplus a_2||a_1 \oplus a_3,\\
LR^2_{(32)} &= a_0 \oplus c||a_1 \oplus c||a_2 \oplus c||a_3 \oplus c,\\
RL^2_{(32)} &= a_6 \oplus c||a_7 \oplus x||a_4 \oplus a_6||a_5 \oplus a_7 \oplus x \oplus c,\\
RR^2_{(32)} &= a_4 \oplus c||a_5 \oplus c||a_6 \oplus c||a_7 \oplus x.
\end{aligned}$$

Therefore, we have the input of the 2nd round function $f64^2$ is $X^2_{(64)} = X^2_{0(8)}||X^2_{1(8)}\ldots||X^2_{7(8)}$, where each byte of the input can be expressed as:

$$
\begin{aligned}
X^2_{0(8)} &= a_0 \oplus a_2, & X^2_{4(8)} &= a_4 \oplus a_6,\\
X^2_{1(8)} &= a_1 \oplus a_3, & X^2_{5(8)} &= a_5 \oplus a_7 \oplus c \oplus x,\\
X^2_{2(8)} &= a_0 \oplus c, & X^2_{6(8)} &= a_4 \oplus c,\\
X^2_{3(8)} &= a_1 \oplus c, & X^2_{7(8)} &= a_5 \oplus c.
\end{aligned}
$$

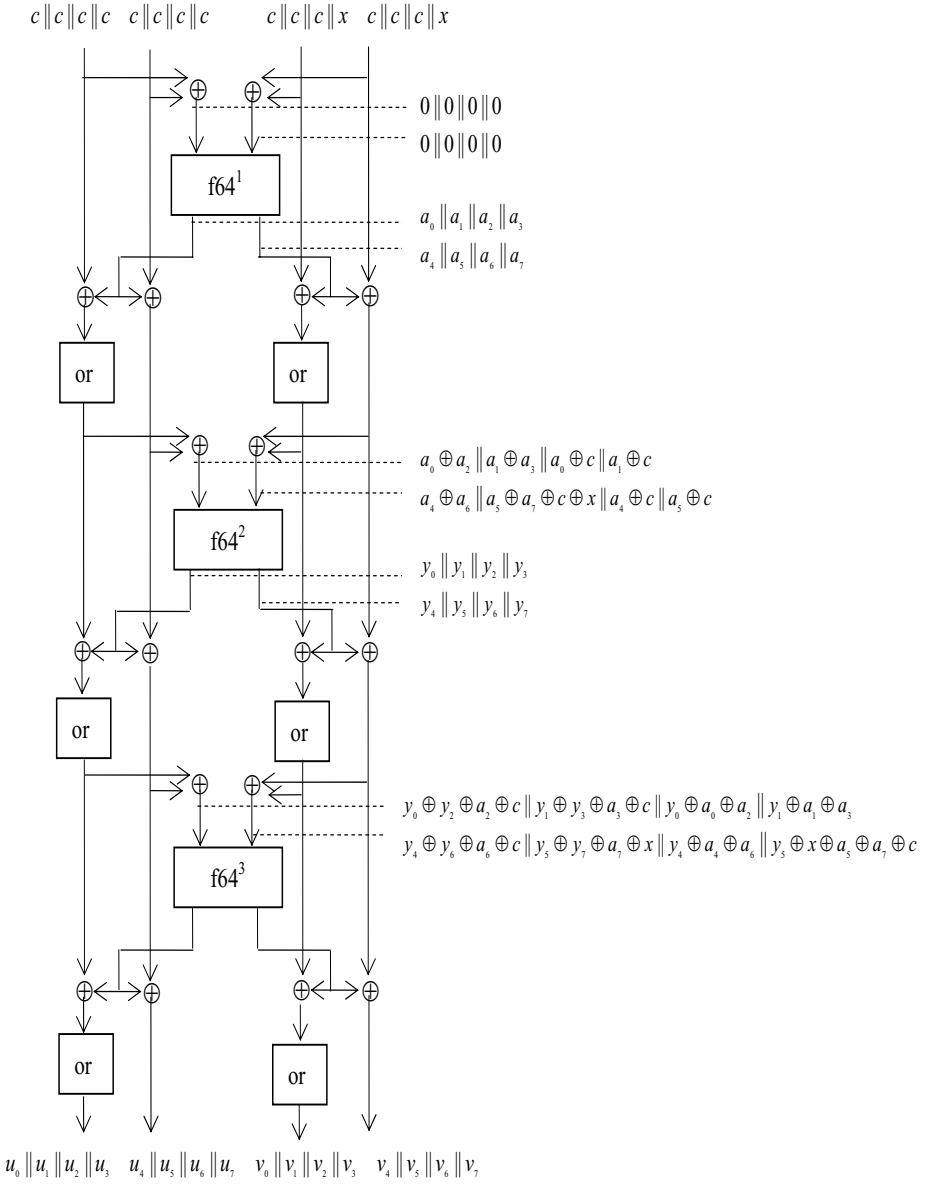


Fig. 3. 3-Round Distinguishers of FOX128

Let $f64^2(X^2_{(64)}) = (y_0||y_1 \dots y_7)$, then the output of the 2nd round transformation can be represented as follows:

$$
\begin{aligned}
LL^3_{(32)} &= y_2 \oplus a_0 \oplus a_2 || y_3 \oplus a_1 \oplus a_3 || y_0 \oplus y_2 \oplus a_0 \oplus c || y_1 \oplus y_3 \oplus a_1 \oplus c,\\
LR^3_{(32)} &= y_0 \oplus a_0 \oplus c || y_1 \oplus a_1 \oplus c || y_2 \oplus a_2 \oplus c || y_3 \oplus a_3 \oplus c,\\
RL^3_{(32)} &= y_6 \oplus a_4 \oplus a_6 || y_7 \oplus a_5 \oplus a_7 \oplus c \oplus x || y_4 \oplus y_6 \oplus a_4 \oplus c || y_5 \oplus y_7 \oplus a_5 \oplus c,\\
RR^3_{(32)} &= y_4 \oplus a_4 \oplus c || y_5 \oplus a_5 \oplus c || y_6 \oplus a_6 \oplus c || y_7 \oplus a_7 \oplus x.
\end{aligned}
$$

Thus, we have the input of the 3rd round function $f64^3$ is $X^3_{(64)} = X^3_{0(8)}||X^3_{1(8)} \dots ||X^3_{7(8)}$, where each byte of the input can be expressed as:

$$
\begin{aligned}
X^3_{0(8)} &= y_0 \oplus y_2 \oplus a_2 \oplus c, & X^3_{4(8)} &= y_4 \oplus y_6 \oplus a_6 \oplus c,\\
X^3_{1(8)} &= y_1 \oplus y_3 \oplus a_3 \oplus c, & X^3_{5(8)} &= y_5 \oplus y_7 \oplus a_7 \oplus x,\\
X^3_{2(8)} &= y_0 \oplus a_0 \oplus a_2, & X^3_{6(8)} &= y_4 \oplus a_4 \oplus a_6,\\
X^3_{3(8)} &= y_1 \oplus a_1 \oplus a_3, & X^3_{7(8)} &= y_5 \oplus x \oplus a_5 \oplus a_7 \oplus c.
\end{aligned}
$$

By observing the high-level structure of FOX128, we get

$$\mathbf{or}^{-1}(LL^4_{(32)}) \oplus LR^4_{(32)} = X^3_{0(8)}||X^3_{1(8)}||X^3_{2(8)}||X^3_{3(8)},$$

$$\mathbf{or}^{-1}(RL^4_{(32)}) \oplus RR^4_{(32)} = X^3_{4(8)}||X^3_{5(8)}||X^3_{6(8)}||X^3_{7(8)}.$$

From the definition of $\mathbf{or}^{-1}$, we have

$$\mathbf{or}^{-1}(LL^4_{(32)}) = LL^4_{0(8)} \oplus LL^4_{2(8)}||LL^4_{1(8)} \oplus LL^4_{3(8)}||LL^4_{0(8)}||LL^4_{1(8)},$$

$$\mathbf{or}^{-1}(RL^4_{(32)}) = RL^4_{0(8)} \oplus RL^4_{2(8)}||RL^4_{1(8)} \oplus RL^4_{3(8)}||RL^4_{0(8)}||RL^4_{1(8)}.$$

Thus, we have the following from the above equations.

$$
\begin{aligned}
LL^4_{0(8)} \oplus LL^4_{2(8)} \oplus LR^4_{0(8)} &= y_0 \oplus y_2 \oplus a_2 \oplus c,\\
LL^4_{1(8)} \oplus LL^4_{3(8)} \oplus LR^4_{1(8)} &= y_1 \oplus y_3 \oplus a_3 \oplus c,\\
LL^4_{0(8)} \oplus LR^4_{2(8)} &= y_0 \oplus a_0 \oplus a_2,\\
LL^4_{1(8)} \oplus LR^4_{3(8)} &= y_1 \oplus a_1 \oplus a_3,\\
RL^4_{0(8)} \oplus RL^4_{2(8)} \oplus RR^4_{0(8)} &= y_4 \oplus y_6 \oplus a_6 \oplus c,\\
RL^4_{1(8)} \oplus RL^4_{3(8)} \oplus RR^4_{1(8)} &= y_5 \oplus y_7 \oplus a_7 \oplus x,\\
RL^4_{0(8)} \oplus RR^4_{2(8)} &= y_4 \oplus a_4 \oplus a_6,\\
RL^4_{1(8)} \oplus RR^4_{3(8)} &= y_5 \oplus x \oplus a_5 \oplus a_7 \oplus c.
\end{aligned}
$$

Further we have the following:

$$
\begin{aligned}
&LL^4_{2(8)} \oplus LR^4_{0(8)} \oplus LR^4_{2(8)} = y_2 \oplus a_0 \oplus c,\\
&LL^4_{3(8)} \oplus LR^4_{1(8)} \oplus LR^4_{3(8)} = y_3 \oplus a_1 \oplus c,\\
&LL^4_{0(8)} \oplus LR^4_{2(8)} = y_0 \oplus a_0 \oplus a_2,\\
&LL^4_{1(8)} \oplus LR^4_{3(8)} = y_1 \oplus a_1 \oplus a_3,\\
&RL^4_{2(8)} \oplus RR^4_{0(8)} \oplus RR^4_{2(8)} = y_6 \oplus a_4 \oplus c,\\
&RL^4_{3(8)} \oplus RR^4_{1(8)} \oplus RR^4_{3(8)} = y_7 \oplus a_5 \oplus c,\\
&RL^4_{0(8)} \oplus RR^4_{2(8)} = y_4 \oplus a_4 \oplus a_6,
\end{aligned}
$$

Now we analyze the property of $y_i (0 \le i \le 7)$. Let $y = \mathbf{s}(x \oplus a_5 \oplus a_7 \oplus c \oplus RK0^2_{5(8)})$, then $y_i = \mathbf{s}(y \oplus b_i) \oplus RK0^2_{i(8)}$, here $b_i (0 \le i \le 7)$ are entirely determined by $a_i (0 \le i \le 7)$, c and $RK^2_{(128)}$, so $b_i (0 \le i \le 7)$ are constants when the user key is fixed.

Since $\mathbf{s}$ is a permutation, $y = \mathbf{s}(x \oplus a_5 \oplus a_7 \oplus c \oplus RK0^2_{5(8)})$ differs when x takes different values and the user key is fixed. As a consequence, $y_i = \mathbf{s}(y \oplus b_i) \oplus RK0^2_{i(8)}$ will have different values when x takes different values and the user key is fixed. Thus, from the above discussion we know that $LL^4_{2(8)} \oplus LR^4_{0(8)} \oplus LR^4_{2(8)}$, $LL^4_{3(8)} \oplus LR^4_{1(8)} \oplus LR^4_{3(8)}$, $LL^4_{0(8)} \oplus LR^4_{2(8)}$, $LL^4_{1(8)} \oplus LR^4_{3(8)}$, $RL^4_{2(8)} \oplus RR^4_{0(8)} \oplus RR^4_{2(8)}$, $RL^4_{3(8)} \oplus RR^4_{1(8)} \oplus RR^4_{3(8)}$ and $RL^4_{0(8)} \oplus RR^4_{2(8)}$ each will have different values when x takes different values. Therefore, we get the following theorem.

Theorem 1. *Let* $P_{(128)} = LL^1_{(32)} || LR^1_{(32)} || RL^1_{(32)} || RR^1_{(32)}$ *and* $P^*_{(128)} = LL^{1*}_{(32)} || LR^{1*}_{(32)} || RL^{1*}_{(32)} || RR^{1*}_{(32)}$ *be two plaintexts of 3-round FOX128,* $C_{(128)} = LL^4_{(32)} || LR^4_{(32)} || RL^4_{(32)} || RR^4_{(32)}$ *and* $C^*_{(128)} = LL^{4*}_{(32)} || LR^{4*}_{(32)} || RL^{4*}_{(32)} || RR^{4*}_{(32)}$ *be the corresponding ciphertexts.* $RR^i_{l(8)} (0 \leqslant l \leqslant 3)$ *denotes the* $(l+1)^{th}$ *byte of* $RR^i_{(32)}$. *If* $LL^1_{(32)} = LR^1_{(32)} = LL^{1*}_{(32)} = LR^{1*}_{(32)}, RL^1_{(32)} = RR^1_{(32)}$, $RL^{1*}_{(32)} = RR^{1*}_{(32)}$, $RR^1_{j(8)} = RR^{1*}_{j(8)} (j = 0, 1, 2)$, $RR^1_{3(8)} \neq RR^{1*}_{3(8)}$, *then* $C_{(128)}$ *and* $C^*_{(128)}$ *satisfy the following inequalities:*

$$LL^4_{2(8)} \oplus LR^4_{0(8)} \oplus LR^4_{2(8)} \neq LL^{4*}_{2(8)} \oplus LR^{4*}_{0(8)} \oplus LR^{4*}_{2(8)} \quad (1)$$

$$LL^4_{3(8)} \oplus LR^4_{1(8)} \oplus LR^4_{3(8)} \neq LL^{4*}_{3(8)} \oplus LR^{4*}_{1(8)} \oplus LR^{4*}_{3(8)} \quad (2)$$

$$LL^4_{0(8)} \oplus LR^4_{2(8)} \neq LL^{4*}_{0(8)} \oplus LR^{4*}_{2(8)} \quad (3)$$

$$LL^4_{1(8)} \oplus LR^4_{3(8)} \neq LL^{4*}_{1(8)} \oplus LR^{4*}_{3(8)} \quad (4)$$

$$RL^4_{2(8)} \oplus RR^4_{0(8)} \oplus RR^4_{2(8)} \neq RL^{4*}_{2(8)} \oplus RR^{4*}_{0(8)} \oplus RR^{4*}_{2(8)} \quad (5)$$

$$RL^4_{3(8)} \oplus RR^4_{1(8)} \oplus RR^4_{3(8)} \neq RL^{4*}_{3(8)} \oplus RR^{4*}_{1(8)} \oplus RR^{4*}_{3(8)} \quad (6)$$

$$RL^4_{0(8)} \oplus RR^4_{2(8)} \neq RL^{4*}_{0(8)} \oplus RR^{4*}_{2(8)} \quad (7)$$

From the above discussion, we have $RL^4_{1(8)} \oplus RR^4_{3(8)} = y_5 \oplus x \oplus a_5 \oplus a_7 \oplus c$, and y_5 will have different values when x take different values. So we can get the following corollary, which is similar to the integral distinguisher presented in [9] and [17].

Corollary 1. *Let* $Pj_{(128)} = LLj^1_{(32)}||LRj^1_{(32)}||RLj^1_{(32)}||RRj^1_{(32)} (0 \le j \le 255)$ *be 256 plaintexts of 3-round FOX128,* $Cj_{(128)} = LLj^4_{(32)}||LRj^4_{(32)}||RLj^4_{(32)}||RRj^4_{(32)}$ *be the corresponding ciphertexts. If* $LLj^1_{(32)} = LRj^1_{(32)}, RLj^1_{(32)} = RRj^1_{(32)}$, $RLj_{l(8)} (l = 0, 1, 2)$ *are constants, and* $RLj_{3(8)}$ *take all possible values between 0 and 255, then* $Cj_{(128)} (0 \le j \le 255)$ *satisfy:*

$$\bigoplus_{j=0}^{255} (RLj^4_{1(8)} \oplus RRj^4_{3(8)}) = 0 \tag{8}$$

4 Attacks on Reduced-Round FOX128

4.1 Attacking 4-Round FOX128

This section explains the attack on 4-round FOX128 in detail. The last round omit the **or** transformation. First we recover 72 bits subkey $RK0^4_{(64)}$ and $RK1^4_{0(8)}$.

Choose plaintext $P_{(128)} = LL^1_{(32)}||LR^1_{(32)}||RL^1_{(32)}||RR^1_{(32)}$, and let $C_{(128)} = LL^5_{(32)}||LR^5_{(32)}||RL^5_{(32)}||RR^5_{(32)}$ be the corresponding ciphertext. The input of the fourth round function $f64^4$ is $LL^5_{(32)} \oplus LR^5_{(32)}||RL^5_{(32)} \oplus RR^5_{(32)}$. We can calculate the value of $LL^4_{2(8)} \oplus LR^4_{2(8)}$, because $LL^4_{2(8)} \oplus LR^4_{2(8)} = LL^5_{2(8)} \oplus LR^5_{2(8)}$. If we guess the value of $LR^4_{0(8)}$, then we can guess $LL^4_{2(8)} \oplus LR^4_{2(8)} \oplus LR^4_{0(8)}$. From the structure of $f64^4$, it is known that the value of $LR^4_{0(8)}$ is entirely determined by the input $LL^5_{(32)} \oplus LR^5_{(32)}||RL^5_{(32)} \oplus RR^5_{(32)}$ and subkey $RK0^4_{(64)}$, $RK1^4_{0(8)}$. Thus using the inequality (1) of Theorem 1, we construct the following algorithm to recover $RK0^4_{(64)}$ and $RK1^4_{0(8)}$.

Algorithm 1

Step 1. Choose 166 plaintexts $Pj_{(128)} = LLj^1_{(32)}||LRj^1_{(32)}||RLj^1_{(32)}||RRj^1_{(32)} (0 \le j \le 165)$ as follows:

$$\begin{aligned} LLj^1_{(32)} &= (c||c||c||c), \\ LRj^1_{(32)} &= (c||c||c||c), \\ RLj^1_{(32)} &= (c||c||c||j), \\ RRj^1_{(32)} &= (c||c||c||j). \end{aligned}$$

where c is a constant, $0 \le j \le 165$. The corresponding ciphertexts are $Cj_{(128)} = LLj^5_{(32)}||LRj^5_{(32)}||RLj^5_{(32)}||RRj^5_{(32)}$.

Step 2. For each possible value of $RK0^4_{(64)}||RK1^4_{0(8)}$, first compute the first output byte $Yj^4_{0(8)}$ of $f64^4(LLj^5_{(32)} \oplus LRj^5_{(32)}||RLj^5_{(32)} \oplus RRj^5_{(32)})$, and then compute

$$\triangle_j = Yj^4_{0(8)} \oplus LLj^5_{2(8)} \oplus LRj^5_{2(8)} \oplus LRj^5_{0(8)}.$$

Check if there is a collision among $\triangle_j$. If so, discard the value of $RK0^4_{(64)}||RK1^4_{0(8)}$. Otherwise, output $RK0^4_{(64)}||RK1^4_{0(8)}$.

Step 3. For the output values of $RK0^4_{(64)}||RK1^4_{0(8)}$ in Step 2, choose some other plaintexts, and repeat Step 2.

The probability of at least one collision occurs when we throw 166 balls into 256 buckets at random is larger than $1 - e^{-166(166-1)/2\times 2^8} \geq 1 - 2^{-76}$. So the probability of passing the test of Step 2 is less than 2^{-76}. Because the right subkey candidates must pass the test of Step 2, the number of subkey candidates passing Step 2 is about $1 + (2^{72} \times 2^{-76}) \approx 1.06$. Then, only two plaintexts are needed in Step 3. The data complexity of this attack is about 168 chosen plaintexts. And the main time complexity of Algorithm 1 is in Step 2, the time of computing each $\triangle_j$ is less than 1-round encryption, so the time complexity is less than $2^{72} \times 168/4 = 42 \times 2^{72}$ encryptions.

Next we recover $RK1^4_{1(8)}$. The steps are very similar to Algorithm 1, except $RK0^4_{(64)}$ is known here. So the number of candidates is 2^8, only 64 chosen plaintexts are needed (we can use the data chosen in Algorithm 1). Using the inequality (2) in Theorem 1, we can recover $RK1^4_{1(8)}$ by computing

$$\triangle_j = Yj^4_{1(8)} \oplus LLj^5_{3(8)} \oplus LRj^5_{3(8)} \oplus LRj^5_{1(8)}.$$

and the attack requires $2^8 \times 64/4 = 2^{12}$ encryptions.

Knowing $RK0^4_{(64)}$ and $RK1^4_{0(8)}$, using inequality (3) in Theorem 1 and the plaintexts chosen in Algorithm 1, we can recover $RK1^4_{2(8)}$ by computing

$$\triangle_j = Yj^4_{0(8)} \oplus Yj^4_{2(8)} \oplus LLj^5_{0(8)} \oplus LRj^5_{2(8)}$$

and the attack requires 2^{12} encryptions.

Similarly, knowing $RK0^4_{(64)}$ and $RK1^4_{1(8)}$, using inequality (4) in Theorem 1 and the plaintexts chosen in Algorithm 1, we can recover $RK1^4_{3(8)}$ by computing

$$\triangle_j = Yj^4_{1(8)} \oplus Yj^4_{3(8)} \oplus LLj^5_{1(8)} \oplus LRj^5_{3(8)}$$

and the attack requires 2^{12} encryptions.

Furthermore, using inequality (5) in Theorem 1 and the plaintexts chosen in Algorithm 1, we can recover $RK1^4_{4(8)}$ by computing

$$\triangle_j = Yj^4_{4(8)} \oplus RLj^5_{2(8)} \oplus RRj^5_{2(8)} \oplus RRj^5_{0(8)}$$

and the attack requires 2^{12} encryptions.

And using inequality (6) in Theorem 1 and the plaintexts chosen in Algorithm 1, we can recover $RK1^4_{5(8)}$ by computing

$$\triangle_j = Yj^4_{5(8)} \oplus RLj^5_{3(8)} \oplus RRj^5_{3(8)} \oplus RRj^5_{1(8)}$$

and the attack requires 2^{12} encryptions.

Knowing $RK0^4_{(64)}$ and $RK1^4_{4(8)}$, using inequality(7) in Theorem 1 and the plaintexts chosen in Algorithm 1, we can recover $RK1^4_{6(8)}$ by computing

$$\triangle_j = Yj^4_{4(8)} \oplus RLj^5_{0(8)} \oplus Yj^5_{6(8)} \oplus RRj^5_{2(8)}$$

and the attack requires 2^{12} encryptions.

We can't use similar approach to recover $RK1^4_{7(8)}$, fortunately integral technique can be used here. Knowing $RK0^4_{(64))}$ and $RK1^4_{5(8)}$, using equation (8) of Corollary 1, we can construct the following algorithm to recover $RK1^4_{7(8)}$.

Algorithm 2
Step 1. Choose 256 plaintexts $Pj_{(128)} = LLj^1_{(32)}||LRj^1_{(32)}||RLj^1_{(32)}||RRj^1_{(32)} (0 \leq j \leq 255)$ as follows:

$$\begin{aligned} LLj^1_{(32)} &= (c||c||c||c), \\ LRj^1_{(32)} &= (c||c||c||c), \\ RLj^1_{(32)} &= (c||c||c||j), \\ RRj^1_{(32)} &= (c||c||c||j). \end{aligned}$$

where c is a constant, $0 \leq j \leq 255$. The corresponding ciphertexts are $Cj_{(128)} = LLj^5_{(32)}||LRj^5_{(32)}||RLj^5_{(32)}||RRj^5_{(32)}$.

Step 2. For each possible value of $RK1^4_{7(8)}$, first compute $Yj^4_{7(8)}$, and then compute

$$\Delta = \bigoplus_{j=0}^{255} (RL^5_{1(8)} \oplus RR^5_{3(8)} \oplus Yj^4_{5(8)} \oplus Yj^4_{7(8)}).$$

Check if $\Delta = 0$. If not, discard the value of $RK1^4_{7(8)}$. Otherwise, output $RK1^4_{7(8)}$.

Step 3. For the output values of $RK1^4_{7(8)}$ in Step 2, choose another group of plaintexts, and repeat Step 2 until the key candidate is unique.

Wrong values will pass Step 2 successfully with probability 2^{-8}. Thus Algorithm 2 requires about 2^9 chosen plaintexts, and the time complexity is less than $2^9 \times 2^8/4 = 2^{15}$ encryptions. The data in Algorithm 1 can be repeatedly used here, so the data complexity for recovering $RK^4_{(128))}$ is about 2^9, and the time complexity is about $42 \times 2^{72} + 6 \times 2^{12} + 2^{15}$.

Now we have recovered $RK^4_{(128)}$ using 2^9 chosen plaintexts and $42 \times 2^{72} + 6 \times 2^{12} + 2^{15}$ encryptions. By decrypting the 4th round, we can recover $RK^3_{(128)}$, the time complexity is less than $2^{73} + 6 \times 2^{12} + 2^{15}$. Similarly, we can recover $RK^2_{(128))}$ and $RK^1_{(128))}$, the time complexity are both less than $2^{73} + 6 \times 2^{12} + 2^{15}$. Therefore, the attack on the 4-round FOX128 requires 2^9 chosen plaintexts and about $2^{77.6}$ encryptions.

4.2 Attacking 5-Round FOX128

We could extend the previous attack on 5-round FOX128, using a key exhaustive search on the fifth subkey $RK^5_{(128)}$. The attack requires about $2^{205.6}$ encryptions, which is less expensive than a key exhaustive search.

5 Attacks on Reduced-Round FOX64

Similar to FOX128, we can get the following theorem for FOX64.

Theorem 2. *Let* $P_{(64)} = L^1_{(32)}||R^1_{(32)}$ *and* $P^*_{(64)} = L^{1*}_{(32)}||R^{1*}_{(32)}$ *be two plaintexts of 3-round FOX64,* $C_{(64)} = L^4_{(32)}||R^4_{(32)}$ *and* $C^*_{(64)} = L^{4*}_{(32)}||R^{4*}_{(32)}$ *be the corresponding ciphertexts,* $L_{l(8)}$ *denote the* $(l+1)^{th}$ *byte of* $L_{(32)}$*. If* $L^1_{(32)} = R^1_{(32)}, L^{1*}_{(32)} = R^{1*}_{(32)}$*, and* $L^1_{l8)} = L^{1*}_{l(8)} (l = 0, 1, 2)$*,* $L^1_{3(8)} \neq L^{1*}_{3(8)}$*, then* $C_{(64)}$ *and* $C^*_{(64)}$ *satisfy:*

$$L^4_{2(8)} \oplus R^4_{2(8)} \oplus R^4_{0(8)} \neq L^{4*}_{2(8)} \oplus R^{4*}_{2(8)} \oplus R^{4*}_{0(8)}, \tag{9}$$

$$L^4_{3(8)} \oplus R^4_{3(8)} \oplus R^4_{1(8)} \neq L^{4*}_{3(8)} \oplus R^{4*}_{3(8)} \oplus R^{4*}_{1(8)} \tag{10}$$

$$L^4_{0(8)} \oplus R^4_{2(8)} \neq L^{4*}_{0(8)} \oplus R^{4*}_{2(8)} \tag{11}$$

Corollary 2. *Let* $Pj_{(64)} = Lj^1_{(32)}||Rj^1_{(32)} (0 \leq j \leq 255)$ *be 256 plaintexts of 3-round FOX64,* $Cj_{(64)} = Lj^4_{(32)}||Rj^4_{(32)}$ *be the corresponding ciphertexts. If* $Lj^1_{(32)} = Rj^1_{(32)}$*,* $Lj_{l(8)} (l = 0, 1, 2)$ *are constants, and* $Lj_{3(8)}$ *take all possible values between 0 and 255, then* $Cj_{(64)} (0 \leq j \leq 255)$ *satisfy:*

$$\bigoplus_{j=0}^{255} (Lj^4_{1(8)} \oplus Rj^4_{3(8)}) = 0 \tag{12}$$

Using Theorem 2 and Corollary 2, we can construct algorithms similar to those in Section 4, and get four subkeys of 4-round FOX64. The attack requires 2^9 chosen plaintexts, and the time complexity is about $2^{45.4}$ 4-round FOX64 encryptions.

Similarly, we can get subkeys of 5 (6,7)-round FOX64 just through guessing the overall key bits behind the fourth round, then using the attack procedure for 4-round FOX64. The time complexity on 5, 6 and 7-round FOX64 is about $2^{109.4}$, $2^{173.4}$ and $2^{237.4}$ respectively.

6 Concluding Remarks

Since FOX is a new cipher published last year, all we know about its security analysis are limited to [9] and [17]. In this paper, we combine collision technique and integral attack to analyze the security of FOX. The improved integral attack on FOX is more efficient than known integral attacks. The complexity of improved integral attack is $2^{77.6}$ on 4-round FOX128, $2^{205.6}$ against 5-round FOX128, respectively. For FOX64, the complexity of improved integral attack is about $2^{45.4}$ on 4-round FOX64, $2^{109.4}$ against 5-round FOX64, $2^{173.4}$ against 6-round FOX64, $2^{237.4}$ against 7-round FOX64, respectively. The results show that 4-round FOX64/64, 5-round FOX64/128, 6-round FOX64/192, 7-round FOX64/256 and 5-round FOX128/256 are not immune to improved integral attack presented in this paper.

In Section 4 of [17], the author claim *if we test all the* 2^{32} *possible values for* $X^0_{0(32)}$*, the probability to distinguish the four rounds FOX64 from a random*

permutation is 1. This result is not correct, so the 4-round distinguisher presented in [17] is not as effective as that claimed. Hence, we only compare the performance of known integral attacks on FOX in [9] and that of this paper in the following table.

Table 1. The summary of known integral attacks on FOX

Name	round	Time	Notes
FOX64	4	2^{72}	Ref.[9]
FOX64	4	$2^{45.4}$	this paper
FOX64	5	2^{136}	Ref.[9]
FOX64	5	$2^{109.4}$	this paper
FOX64	6	2^{200}	Ref.[9]
FOX64	6	$2^{173.4}$	this paper
FOX64	7	$2^{237.4}$	this paper
FOX128	4	2^{136}	Ref.[9]
FOX128	4	$2^{77.6}$	this paper
FOX128	5	$2^{205.6}$	this paper

Acknowledgment

We would like to thank anonymous referees for their helpful comments and suggestions. This research is supported by the National Natural Science Foundation of China under Grant No. 60373047; the National Basic Research 973 Program of China under Grant No.2004CB318004; and the National High-technique 863 Program of China under Grant No.2003AA144030.

References

1. E. Biham, A. Biryukov, and A. Shamir. Enhancing differential-linear cryptanalysis. *Advances in Cryptology-ASIACRYPT'02,* LNCS 2501, pp.254-266. Springer-Verlag, 2002.
2. E. Biham, O. Dunkelman, and N. Keller. The rectangle attack - rectangling the Serpent. *Advances in Cryptology-EUROCRYPT'01,* LNCS 2045, pp.340-357. Springer-Verlag, 2001.
3. E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. *Advances in Cryptology-EUROCRYPT'99,* LNCS 2595, pp.12-23. Springer-Verlag, 1999.
4. A. Biryukov and D. Wagner. Slide attacks. *Fast Software Encryption-FSE'99*, LNCS 1636, pp.245-259. Springer-Verlag, 1999.
5. A. Biryukov and D. Wagner. Advanced slide attacks. *Advances in Cryptology-EUROCRYPT'00,* LNCS 1807, pp.589-606. Springer-Verlag, 2000.
6. N. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. *Advances in Cryptology-ASIACRYPT'02,* LNCS 2595, pp.267-287. Springer-Verlag, 2002.
7. C. Harpes and J. Massey. Partitioning cryptanalysis. *Fast Software Encryption-FSE'97*, LNCS 1267, pp.13-27. Springer-Verlag, 1997.

8. Y. Hu, Y. Zhang, and G. Xiao. Integral cryptanalysis of SAFER++. *Electronics Letters,* Vol.35, No.17, pp.1458-1459.
9. P. Junod and S. Vaudenay. FOX: a new family of block ciphers. *Selected Areas in Cryptography-SAC 2004*, LNCS 3357, pp.114-129. Springer-Verlag, 2005.
FOX Specifications Version 1.2 appeared on http://crypto.junod.info
10. P. Junod and S. Vaudenay. Perfect diffusion primitives for block ciphers—building efficient MDS matrices. *Selected Areas in Cryptography-SAC 2004*, LNCS 3357, pp.84-99. Springer-Verlag, 2005.
11. T. Jakobsen and L. Knudsen. The interpolation attack against block ciphers. *Fast Software Encryption-FSE'99*, LNCS 1267, pp.28-40. Springer-Verlag,1999.
12. L. Knudsen. Truncated and higher order differentials. *Fast Software Encryption-FSE'95*, LNCS 2595, pp.196-211. Springer-Verlag, 1995.
13. L. Knudsen and D. Wagner. Integral cryptanalysis(extended abstract). *Fast Software Encryption-FSE 2002*, LNCS 2595, pp.112-127. Springer-Verlag, 2002.
14. X. Lai and J. Massey. A proposal for a new block encryption standard. *Advances in Cryptology-EUROCRYPT'90*, LNCS 473, pp.389-404. Springer-Verlag, 1990.
15. M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, Vol.17, No.2, pp.373-386.
16. K. Lanford and E. Hellman. Differential-linear cryptanalysis. *Advances in Cryptology-CRYPTO'94,* LNCS 893, pp.17-25. Springer-Verlag, 1994.
17. Marine Minier. An integral cryptanalysis against a five rounds version of FOX. *Western European Workshop on Research in Cryptography-WEWoRC,* July 5-7, Leuven, Belgium, 2005.
18. S. Murphy and M. Robshaw. Comments on the security of the AES and the XSL technique. *Electronic Letters,* Vol.39, No.1, pp.36-38.
19. S. Vaudenay. On the Lai-Massey scheme. *Advances in Cryptology-ASIACRYPT'99*, LNCS 1716, pp.8-19. Springer-Verlag, 1999.
20. D. Wagner. The boomerang attack. *Fast Software Encryption-FSE'99*, LNCS 1636, pp.157-170. Springer-Verlag, 1999.
21. Hongjun Wu. Related-cipher attacks. *Information and Communications Security-ICICS 2002,* LNCS 2513, pp.447-455. Springer-Verlag, 2002.
22. Wenling Wu, Dengguo Feng, and Hua Chen. Collision attack and pseudorandomness of reduced-round Camellia. *Selected Areas in Cryptography-SAC 2004*, LNCS 3357, pp.256-270. Springer-Verlag, 2005.
23. Y.Yeom, S.Park, and I.Kim. On the security of Camellia against the square attack. *Fast Software Encryption-FSE'02*, LNCS 2356, pp.89-99. Springer-Verlag, 2002.

Hybrid Symmetric Encryption Using Known-Plaintext Attack-Secure Components

Kazuhiko Minematsu and Yukiyasu Tsunoo

NEC Corporation, 1753 Shimonumabe, Nakahara-Ku,
Kawasaki 211-8666, Japan
k-minematsu@ah.jp.nec.com

Abstract. This paper describes a hybrid symmetric cipher that combines a strongly-secure function, e.g., a pseudorandom function (PRF), which is secure against any Chosen-Plaintext Attack, and a weak PRF, which is only secure against any Known-Plaintext Attack. Although this kind of composition is potentially faster than the modes of PRFs, it has not been extensively studied. Our main contribution is in proposing a new block cipher scheme that is suitable for hybrid composition. We describe efficient hybrid constructions of pseudorandom permutation and strong pseudorandom permutation for an arbitrarily large block size using our new scheme.

1 Introduction

In 1988, Luby and Rackoff [13] began studying the secure composition of cryptographic components. They showed that only a few iterations of a Feistel round could be secure against any Chosen-Plaintext Attack (CPA) or Chosen-Ciphertext Attack (CCA), if the underlying round functions were pseudorandom functions [8] (PRFs), i.e., secure against any CPA. Following Luby and Rackoff's work, many researchers have studied the compositions of various cryptographic systems.

In this paper, we discuss hybrid[1] symmetric encryptions combining a component secure against any Known-Plaintext Attack (KPA), which is called a weak PRF (WPRF), and a stronger component such as PRF. WPRFs were studied by many researchers [1, 6, 23, 24] and widely accepted as one of the weak cryptographic primitives. Since KPA is weaker attack than CPA, a WPRF is reasonably assumed to be faster than a PRF. For example, it was pointed out [6] that the WPRF based on the Decisional Diffie-Hellman (DDH) assumption could be more efficient than the DDH-based PRF proposed by Naor and Reingold [22]. Consider a mode that invokes a PRF. If almost all invocations of the PRF can be securely substituted with those of a WPRF, then the resulting mode would be much faster than the original mode based only on PRF. In practice, such hybrid modes can be seen as modes of operation for multiple cryptographic components that have different security-levels, such as a strongly secure block cipher and its reduced-round version, or a (strong) block cipher and a (weak) stream cipher.

[1] In this paper, "hybrid" means combining strong and weak primitives.

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 242–260, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Although the idea of using multiple components can be seen in previous studies, for instance Bear and Lion [2], none of them used WPRFs as their components.

The basic idea is that the cascade of a PRF and a WPRF is PRF. This is intuitively correct, since outputs of a PRF, which correspond to the WPRF's inputs, should be close to random in terms of computational indistinguishability. We first prove that this idea actually holds true, and propose a hybrid construction of a PRF with large output (and small input) based on this idea. Such a PRF can be used as a stream cipher accepting an initial vector (IV).

These results are also beneficial to hybrid block ciphers. We propose a new scheme for block ciphers that slightly differs from Feistel. It provides a pseudorandom permutation (PRP) that has double length (i.e., a $2n$-bit block cipher composed of n-bit block components) using one invocation of a PRP and a WPRF, and universal hash function [32] (UH)-based mixing. As it might be impossible to build a double length PRP using two WPRF invocations, our construction is optimal (in terms of the number of n-bit PRP invocation). Moreover, we show that such a hybrid composition is, in a sense, difficult with the original Feistel. Double length PRP has been extensively studied and many schemes have been proposed [13, 25, 27, 12, 19, 28]. However, to our knowledge, our scheme is the first construction that does not need two invocations of a PRF (or PRP).

In addition, our scheme is useful for building a large block cipher. Using our hybrid block cipher scheme combined with our hybrid large output PRF, we build an mn-bit strong PRP (SPRP), which is secure against any combination of CPA and Chosen-Ciphertext Attack (CCA). A large block SPRP has desirable properties for storage encryption [35]. Our construction requires two invocations of an n-bit SPRP, $(m-2)$ invocations of an n-bit WPRF, and two Feistel rounds with UHs, for all $m > 2$. Therefore, its throughput will be close to that of the WPRF we intend to use, and the underlying n-bit SPRP's throughput will not be a problem with a large block size. For a comparison, NR mode [25], which is a highly sophisticated mode to provide a large block SPRP, requires m invocations of an n-bit SPRP and two mixing layers to provide a mn-bit block SPRP. These examples illustrate that our hybrid block cipher construction is highly optimized for both small and large block sizes.

All our security analyses are based on the standard security notions of symmetric cryptography introduced by Bellare et al. [3] and a natural extension of this to deal with KPA, which is the same as the previous studies [1, 6, 22]. We also use a framework that was proposed by Maurer [17] to perform a rigorous security analysis.

2 Preliminaries

2.1 Random Functions and Their Composition

Definition 1. *Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets. Random function (RF) $\mathbf{F} : \mathcal{X} \to \mathcal{Y}$ is a random variable distributed over all functions $\mathcal{X}$ to $\mathcal{Y}$[2]. If $\mathbf{F}$ is distributed*

[2] If $\mathbf{F}$ has key K, uniformly distributed over $\mathcal{K}$, then there is function $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that $\mathbf{F}(x) = f(K, x)$ and $\Pr[\mathbf{F}(x) = y] = |\{k \in \mathcal{K} : f(k, x) = y\}|/|\mathcal{K}|$.

over all permutations on $\mathcal{X}$, it is called a random permutation (RP) on $\mathcal{X}$. A uniform random function (URF) : $\{0,1\}^n \to \{0,1\}^m$ is an RF with uniform distribution on all functions $\{0,1\}^n$ to $\{0,1\}^m$ and denoted by $\mathbf{R}_{n,m}$. A uniform random permutation (URP) on $\{0,1\}^n$ is an RP with uniform distribution on all n-bit permutations and denoted by $\mathbf{P}_n$.

Note that, in this paper, the word "random" does not imply uniformity. It only means it is probabilistic. We used bold symbols for RFs. When $\mathbf{F}$ and $\mathbf{G}$ are two RFs that have the same input/output space, we say they are *compatible*.

For simplicity, most of our results deal with cases when n-bit block components are used. We will use the following composition operators.

Definition 2. *Let $\mathbf{F} : \mathcal{X} \to \mathcal{Y}$, and $\mathbf{G} : \mathcal{Y} \to \mathcal{Z}$. Let $\mathbf{F} \circ \mathbf{G} : \mathcal{X} \to \mathcal{Z}$ such that $\mathbf{F} \circ \mathbf{G}(x) = \mathbf{G}(\mathbf{F}(x))$[3] , and let $\mathbf{F} \triangleleft \mathbf{G} : \mathcal{X} \to \mathcal{Y} \times \mathcal{Z}$ such that $\mathbf{F} \triangleleft \mathbf{G}(x) = (\mathbf{F}(x), \mathbf{G}(\mathbf{F}(x)))$ for all $x \in \mathcal{X}$.*

2.2 Security Measures and Their Properties

We breifly describe our security measures in a standard framework introduced by Bellare et al.[3]. Let $\mathbf{F}, \mathbf{G}$ be two compatible RFs. Let $\mathbf{D}$ be an attacker that can access the encryption oracle (EO). Here, EO has implemented $\mathbf{H}$, which is equivalent to either $\mathbf{F}$ or $\mathbf{G}$. $\mathbf{D}$ determines whether $\mathbf{H}$ is $\mathbf{F}$ or $\mathbf{G}$ after a predetermined number of queries and answers. The advantage of $\mathbf{D}$ is defined as

$$V(\mathbf{F}, \mathbf{G}|\mathbf{D}) \stackrel{\text{def}}{=} |\Pr[\mathbf{D}\text{'s guess is } \mathbf{F}|\mathbf{H} = \mathbf{F}] - \Pr[\mathbf{D}\text{'s guess is } \mathbf{F}|\mathbf{H} = \mathbf{G}]|. \quad (1)$$

Definition 3. *The CPA-advantage (KPA-advantage) is defined as the maximal advantage of all attackers using CPA (KPA). That is,*

$$\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{F},\mathbf{G}}(q,\tau) \stackrel{\text{def}}{=} \max_{\mathbf{D}:(q,\tau)\text{-}CPA} V(\mathbf{F}, \mathbf{G}|\mathbf{D}),\ \mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{F},\mathbf{G}}(q,\tau) \stackrel{\text{def}}{=} \max_{\mathbf{D}:(q,\tau)\text{-}KPA} V(\mathbf{F}, \mathbf{G}|\mathbf{D}).$$

Here, (q,τ)-CPA denotes a CPA that uses q queries with time complexity τ[4]*. Similarly, (q,τ)-KPA denotes a KPA that uses q independent and uniformly random queries with time complexity τ. Especially, let $\mathbf{R}$ be a URF compatible to $\mathbf{F}$. Then, $\mathrm{Adv}^{\mathrm{prf}}_{\mathbf{F}}(q,\tau) \stackrel{\text{def}}{=} \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{F},\mathbf{R}}(q,\tau)$ and $\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,\tau) \stackrel{\text{def}}{=} \mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{F},\mathbf{R}}(q,\tau)$. If $\mathbf{F}$ is an RP, we have $\mathrm{Adv}^{\mathrm{prp}}_{\mathbf{F}}(q,\tau) \stackrel{\text{def}}{=} \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{F},\mathbf{P}}(q,\tau)$ where $\mathbf{P}$ is the URP.*

Finally, we will define the CCA-advantage. This provides the security against an attacker who can adaptively choose a plaintext (a ciphertext) and receive the ciphertext (the plaintext). It can be defined as a variant of the CPA-advantage.

Definition 4. *Let $\mathbf{F}$ and $\mathbf{G}$ be two RPs on $\mathcal{X}$. The inverse of $\mathbf{F}$ is denoted by $\mathbf{F}^{-1}$. Let $\langle\mathbf{F}\rangle$ be the RF:$\mathcal{X} \times \{0,1\} \to \mathcal{X}$ such that $\langle\mathbf{F}\rangle(x_i, d_i) = \mathbf{F}(x_i)$ if*

[3] Note that the definition of $\circ$ is different from the standard one.

[4] The time complexity includes the worst case execution time and the program size, in some fixed RAM computation model.

$d_i = 0$ *and* $\mathbf{F}^{-1}(x_i)$ *if* $d_i = 1$. *The CCA-advantage is defined as* $\mathrm{Adv}^{\mathrm{cca}}_{\mathbf{F},\mathbf{G}}(q,\tau) \stackrel{\mathrm{def}}{=} \mathrm{Adv}^{\mathrm{cpa}}_{\langle\mathbf{F}\rangle,\langle\mathbf{G}\rangle}(q,\tau)$ *and we have* $\mathrm{Adv}^{\mathrm{sprp}}_{\mathbf{F}}(q,\tau) \stackrel{\mathrm{def}}{=} \mathrm{Adv}^{\mathrm{cca}}_{\mathbf{F},\mathbf{P}}(q,\tau)$.

If $\mathbf{F}$ has a small CPA-advantage for some sufficiently large q and τ in distinguishing $\mathbf{F}$ from URF, it is called a pseudorandom function (PRF). In addition, if $\mathbf{F}$ is invertible, it is called a pseudorandom permutation (PRP). Similarly, if $\mathbf{F}$ has a small KPA-advantage, it is called a weak PRF [24] (WPRF), and if $\mathbf{F}$ is an RP and has a small CCA-advantage, it is called a strong PRP (SPRP).

It is well known that triangle inequality holds for the CPA, KPA, and CCA-advantages. More precisely, we have $\mathrm{Adv}^{***}_{\mathbf{F},\mathbf{H}}(q,\tau) \le \mathrm{Adv}^{***}_{\mathbf{F},\mathbf{G}}(q,\tau) + \mathrm{Adv}^{***}_{\mathbf{G},\mathbf{H}}(q,\tau)$ for $*** \in \{\mathrm{cpa},\mathrm{kpa},\mathrm{cca}\}$.

Let $\mathbf{F},\mathbf{G}$ be compatible RFs with n-bit input, and let $\mathbf{R}$ be the URF with n-bit output. The following equation plays an important role in our analysis.

$$\mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{F},\mathbf{G}}(q,\tau) = \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{R}\triangleleft\mathbf{F},\mathbf{R}\triangleleft\mathbf{G}}(q,\tau'), \text{ where } \tau = \tau + O(nq). \tag{2}$$

This is natural, since all adaptive attacks are useless in distinguishing $\mathbf{R}\triangleleft\mathbf{F}$ from $\mathbf{R}\triangleleft\mathbf{G}$. Actually, the difference between $\mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{F},\mathbf{G}}(q,\tau)$ and $\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{R}\triangleleft\mathbf{F},\mathbf{R}\triangleleft\mathbf{G}}(q,\tau)$ only depends on the time for generating uniformly random plaintexts (for $\mathbf{F}$ and $\mathbf{G}$). We assume that the time for generating q uniformly random plaintexts needs $O(nq)$ time. Hereafter, $\mathcal{X}$ denotes $\{0,1\}^n$ and τ' denotes $\tau + O(nq)$.

Lemma 1. *For any* $\mathbf{F}$ *and* $\mathbf{G}$, $\mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{F},\mathbf{G}}(q,\tau) \le \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{F},\mathbf{G}}(q,\tau)$. *Moreover, let* $\mathbf{E}$ *be an RF that can be cascaded to* $\mathbf{F}$ *and* $\mathbf{G}$, *and* $\mathbf{R}$ *be the URF compatible with* $\mathbf{E}$. *Then,* $\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{E}\triangleleft\mathbf{F},\mathbf{E}\triangleleft\mathbf{G}}(q,\tau) \le 2\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{E},\mathbf{R}}(q,\tau) + \mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{F},\mathbf{G}}(q,\tau')$.

Proof. The first claim is obvious. For the second, we have

$$\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{E}\triangleleft\mathbf{F},\mathbf{E}\triangleleft\mathbf{G}}(q,\tau) \le \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{E}\triangleleft\mathbf{F},\mathbf{R}\triangleleft\mathbf{F}}(q,\tau) + \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{R}\triangleleft\mathbf{F},\mathbf{R}\triangleleft\mathbf{G}}(q,\tau) + \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{R}\triangleleft\mathbf{G},\mathbf{E}\triangleleft\mathbf{G}}(q,\tau).$$

Combining the above inequality with Eq. (2) proves the second claim.

2.3 Monotone Event Sequence and Conditional Equivalences

We will use a methodology developed by Maurer [17, 18] to analyze information-theoretic security, i.e., the maximum advantage without computational restrictions. Here, let us briefly describe his notations. Consider event a_i defined for i input/output pairs of $\mathbf{F}$. Let $\overline{a_i}$ be the negation of a_i. We assumed a_i was monotone, i.e., a_i never occurred if $\overline{a_{i-1}}$ occurred. For instance, a_i is monotone if this indicates that all i outputs are distinct. An infinite sequence of monotone events $\mathcal{A} = a_0 a_1 \dots$ is called a monotone event sequence (MES). Here, a_0 denotes some tautological event. Note that $\mathcal{A} \wedge \mathcal{B} = (a_0 \wedge b_0)(a_1 \wedge b_1)\dots$ is an MES if $\mathcal{A} = a_0 a_1 \dots$ and $\mathcal{B} = b_0 b_1 \dots$ are both MESs. For any sequence of random variables, $X_1, X_2, \dots$, let X^i denote $(X_1, \dots, X_i)$. After this, $\mathrm{dist}(X^i)$ will denote an event where $X_1, X_2, \dots, X_i$ are distinct.

Let MESs $\mathcal{A}$ and $\mathcal{B}$ be defined for $\mathbf{F} : \mathcal{X} \to \mathcal{Y}$ and $\mathbf{G} : \mathcal{X} \to \mathcal{Y}$, respectively. Let $X_i \in \mathcal{X}$ and $Y_i \in \mathcal{Y}$ be the i-th input and output. Let $P^{\mathbf{F}}$ be the probability space defined by $\mathbf{F}$. For example, $P^{\mathbf{F}}_{Y_i|X^iY^{i-1}}(y^i, x^i)$ means $\Pr[Y_i = y_i | X^i = x^i, Y^{i-1} = y^{i-1}]$ where $Y_j = \mathbf{F}(X_j)$ for $j = 1, \dots$.

Definition 5. *Let us say* $\mathbf{F}$ *and* $\mathbf{G}$ *are equivalent and write* $\mathbf{F} \equiv \mathbf{G}$ *if* $P^{\mathbf{F}}_{Y_i|X^iY^{i-1}} = P^{\mathbf{G}}_{Y_i|X^iY^{i-1}}$*, which means* $P^{\mathbf{F}}_{Y_i|X^iY^{i-1}}(y^i, x^i) = P^{\mathbf{G}}_{Y_i|X^iY^{i-1}}(y^i, x^i)$ *for all* $x^i \in \mathcal{X}^i$, $y^i \in \mathcal{Y}^i$ *and for all* $i \geq 1$.

Definition 6. *We write* $\mathbf{F}^{\mathcal{A}} \equiv \mathbf{G}^{\mathcal{B}}$ *if* $P^{\mathbf{F}}_{Y_ia_i|X^iY^{i-1}a_{i-1}} = P^{\mathbf{G}}_{Y_ib_i|X^iY^{i-1}b_{i-1}}$ [5] *holds, which means* $P^{\mathbf{F}}_{Y_ia_i|X^iY^{i-1}a_{i-1}}(y^i, x^i) = P^{\mathbf{G}}_{Y_ib_i|X^iY^{i-1}b_{i-1}}(y^i, x^i)$ *holds for all* (y^i, x^i) *such that both* $P^{\mathbf{F}}_{a_{i-1}|X^{i-1}Y^{i-1}}(y^{i-1}, x^{i-1})$ *and* $P^{\mathbf{G}}_{b_{i-1}|X^{i-1}Y^{i-1}}(y^{i-1}, x^{i-1})$ *are positive for all* $i \geq 1$.

Definition 7. *We write* $\mathbf{F}|\mathcal{A} \equiv \mathbf{G}|\mathcal{B}$ *if* $P^{\mathbf{F}}_{Y_i|X^iY^{i-1}a_i} = P^{\mathbf{G}}_{Y_i|X^iY^{i-1}b_i}$ *holds. Moreover, let* $\mathcal{C} = c_0c_1\ldots$ *be an MES defined for* $\mathbf{F}$*. We write* $\mathbf{F}^{\mathcal{A}}|\mathcal{C} \equiv \mathbf{G}^{\mathcal{B}}$ *if* $P^{\mathbf{F}}_{Y_ia_i|X^iY^{i-1}a_{i-1}c_i} = P^{\mathbf{G}}_{Y_ib_i|X^iY^{i-1}b_{i-1}}$ *holds.*

Note that if $\mathbf{F}^{\mathcal{A}} \equiv \mathbf{G}^{\mathcal{B}}$, then $\mathbf{F}|\mathcal{A} \equiv \mathbf{G}|\mathcal{B}$ (but not vice versa).

Definition 8. *For* $\mathcal{A}$ *defined for* $\mathbf{F}$, $\nu(\mathbf{F}, \overline{a_q})$ *denotes the maximal probability of* $\overline{a_q}$ *for any* (q, ∞)*-CPA that interacts with* $\mathbf{F}$*. Similarly,* $\mu(\mathbf{F}, \overline{a_q})$ *denotes the maximal probability of* $\overline{a_q}$ *for any non-adaptive* (q, ∞)*-CPA.*

Clearly, $\mu(\mathbf{F}, \overline{a_q}) \leq \nu(\mathbf{F}, \overline{a_q})$ holds. In addition, $\mu(\mathbf{F}, \overline{a_q})$ equals $\max_{x^q \in \mathcal{X}^q} P^{\mathbf{F}}_{\overline{a_q}|X^q}$, which often makes the analysis of $\mu(\mathbf{F}, \overline{a_q})$ much easier than that of $\nu(\mathbf{F}, \overline{a_q})$.

These equivalences are crucial to the proof of information-theoretic security. For example, if $\mathbf{F}^{\mathcal{A}} \equiv \mathbf{G}^{\mathcal{B}}$, then $\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{F},\mathbf{G}}(q, \infty) \leq \nu(\mathbf{F}, \overline{a_q})$ (Theorem 6 in Appendix A). Moreover, one can turn the analysis of adaptive attacks (i.e., $\nu(*, *)$) into that of non-adaptive attacks (i.e., $\mu(*, *)$) under some additional conditions. We will use a number of Maurer's results including Theorem 6, as these often provide a rigorous security proof, which can not be obtained with other methods[6]. For completeness, these results are cited in Appendix A.

Caveat. Maurer's methodology [17] can only be applied to an information-theoretic setting. In most cases information-theoretic proofs can be easily converted into computational ones, but this is not always the case [18, 21]. However, we do not encounter such difficulties in this paper. His methodology can also be applied to *random systems*, i.e., stateful random functions. We will use some random systems in our proofs for convenience, but in practice, none of our hybrid modes require underlying components to be stateful.

2.4 Why We Should Care About Hybrid Modes?

As we mentioned, the focus of this paper is modes for a PRF combined with a WPRF. However, why do we need such hybrid modes as we already have PRFs? The main advantage of hybrid modes is throughput, since a WPRF is

[5] Here, $P^{\mathbf{F}}_{Y_ia_i|X^iY^{i-1}a_{i-1}}(y^i, x^i)$ is $\Pr[Y_i = y_i, a_i|a_{i-1}, X^{i-1} = x^{i-1}, Y^{i-1} = y^{i-1}]$.

[6] For example, let $\mathbf{C}$ be a $2n$-bit 3-round Feistel with PRFs. Classical analysis requires $q^2/2^{2n}$ as a term appearing in the upper bound of $\mathrm{Adv}^{\mathrm{prp}}_{\mathbf{C}}(q, \tau)$, while Maurer [17] showed this was redundant.

naturally assumed to be faster than a compatible PRF. The following examples demonstrate that this assumption actually holds true in some cases.

Example 1. Let $\mathbf{M}[\mathbf{F}_1, \mathbf{F}_2]$ denote a $2n$-bit 2-round Feistel, where $\mathbf{F}_i$ is the i-th round function:$\mathcal{X} \to \mathcal{X}$ for $i = 1, 2$ (recall that $\mathcal{X}$ denotes $\{0,1\}^n$). The first round of $\mathbf{M}[\mathbf{F}_1, \mathbf{F}_2]$ is left-to-right. That is, the input to $\mathbf{F}_1$ is the left half of the input to $\mathbf{M}[\mathbf{F}_1, \mathbf{F}_2]$. It is well known that the 3-round Feistel where each round function is an independent PRF is PRP. However, the following lemma shows that the 2-round Feistel, which can never be a PRP, can be KPA-secure.

Lemma 2. $\mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{M}[\mathbf{F}_1,\mathbf{F}_2],\mathbf{P}_{2n}}(q,\tau) \leq \mathrm{Adv}^{\mathrm{prf}}_{\mathbf{F}_1}(q,\tau) + \mathrm{Adv}^{\mathrm{prf}}_{\mathbf{F}_2}(q,\tau) + \frac{q^2}{2^n}$.

This lemma is proved by a simple non-adaptive analysis similar to Maurer [15].

Actually, a similar result can be obtained for a generalized Feistel. For example, the type I transformation [33] on mn-bit block input requires $2m-1$ rounds to achieve $2^{n/2}$-bit CPA-security [20] (i.e., the CPA-advantage is negligibly small if $q \ll 2^{n/2}$), whereas m rounds are sufficient to attain $2^{n/2}$-bit KPA-security. However, we omitted the formal descriptions of these results here.

As another example, it was pointed out [6] that the WPRF based on the DDH assumption could be more efficient than the DDH-based PRF construction proposed by Naor and Reingold [22]. These examples illustrate that well-designed hybrid modes can be faster than modes that only use PRFs.

Basically, we can only use WPRFs as building blocks. A mode proposed by Damgård and Nielsen [6] can convert any WPRF into a pseudorandom generator (PRG), which implies that any WPRF can be converted into a PRF using the PRG-to-PRF conversion [7]. However, this takes too much computation time and hence is rather impractical. Still, our proposals can be seen as modes of WPRF if this KPA-to-CPA conversion is incorporated into them[7].

3 Hybrid Construction of Large Output PRF

The basic idea behind hybrid modes is that the cascade of a PRF and a WPRF is a PRF. If the WPRF has large output, then the cascade would have almost the same throughput as that of the WPRF. This idea is intuitively correct. Actually, a similar idea was originally proposed by Aiello, Rajagopalan, and Venkatesan[1], without complete security proof (in fact, they only proved Lemma 3). In this section, we will describe our hybrid construction of a large output PRF called ARV[8], and prove it is secure.

Definition 9. *Let* $\mathbf{F}_1 : \mathcal{X} \to \mathcal{X}$ *be a PRF and* $\mathbf{F}_2 : \mathcal{X} \to \mathcal{X}$ *be a WRPF.* ARV *is defined as* $\mathrm{ARV}_m[\mathbf{F}_1, \mathbf{F}_2] \stackrel{\mathrm{def}}{=} \mathbf{F}_1 \circ L_{\mathbf{F}_2,m}$ *(see Def.2 for the def. of* $\circ$*) , where* $L_{\mathbf{F}_2,m} : \mathcal{X} \to \mathcal{X}^m$ *is* $L_{\mathbf{F}_2,m}(x) = (\mathbf{F}_{2,1}(x), \mathbf{F}_{2,2}(x), \ldots, \mathbf{F}_{2,m}(x))$ *for any* $x \in \mathcal{X}$*. Here,* $\mathbf{F}_{2,1}, \ldots, \mathbf{F}_{2,m}$ *are independently-keyed* m *RFs that are equivalent to* $\mathbf{F}_2$*.*

[7] The term "KPA-to-CPA" conversion was also used in [6]. However, it was not intended as a conversion of WPRF into PRF.

[8] Even though Aiello et al.'s proposal was slightly different from ours, we still called it ARV.

Before analyzing ARV, we have to formally prove that the cascade of PRF and WPRF is PRF.

Theorem 1. *Let* $\mathbf{G} = \mathbf{C} \circ \mathbf{F}$ *and* $\mathbf{G}' = \mathbf{C} \triangleleft \mathbf{F}$*, where* $\mathbf{C} : \mathcal{X} \to \mathcal{X}$*, and* $\mathbf{F} : \mathcal{X} \to \mathcal{X}^m$*. Then,*

$$\mathrm{Adv}^{\mathrm{prf}}_{\mathbf{G}}(q,\tau) \le \mathrm{Adv}^{\mathrm{prf}}_{\mathbf{G}'}(q,\tau) \le \mathrm{Adv}^{\mathrm{prf}}_{\mathbf{C}}(q,\tau) + \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,\tau') + \frac{q^2}{2^{n+1}}. \quad (3)$$

Proof. The first inequality is obvious. For the second inequality, let $\mathbf{H}_1$ and $\mathbf{H}_2$ be $\mathbf{R}_{n,n} \triangleleft \mathbf{F}$, and $\mathbf{R}_{n,n} \triangleleft \mathbf{R}_{n,mn}$, respectively. We then obtain $\mathrm{Adv}^{\mathrm{prf}}_{\mathbf{G}'}(q,\tau) \le \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{G}',\mathbf{H}_1}(q,\tau) + \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{H}_1,\mathbf{H}_2}(q,\tau) + \mathrm{Adv}^{\mathrm{prf}}_{\mathbf{H}_2}(q,\tau)$. It is easy to see that $\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{G}',\mathbf{H}_1}(q,\tau) \le \mathrm{Adv}^{\mathrm{prf}}_{\mathbf{C}}(q,\tau)$ and $\mathrm{Adv}^{\mathrm{prf}}_{\mathbf{H}_2}(q,\tau) \le \mathrm{Adv}^{\mathrm{prf}}_{\mathbf{H}_2}(q,\infty) < \frac{q^2}{2^{n+1}}$. Finally, $\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{H}_1,\mathbf{H}_2}(q,\tau) \le \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,\tau')$ follows from Eq. (2).

Now, the remaining task is to show the KPA-advantage of $L_{\mathbf{F}_2,m}$. This is easily derived from triangle inequality.

Lemma 3. *(in [1, 6]) Let* $\mathbf{F} : \mathcal{X} \to \mathcal{X}$*. Then,* $\mathrm{Adv}^{\mathrm{wprf}}_{L_{\mathbf{F},m}}(q,\tau) \le m \cdot \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,\tau)$.

From Theorem 1 and Lemma 3, it is obvious that using $\mathbf{F}_1(x_i)$ as part of the i-th output does not compromise the security of ARV. That is, we can use $\triangleleft$ instead of $\circ$. We thus have the following corollary.

Corollary 1. *Let* $\mathrm{ARV}^+_m[\mathbf{F}_1,\mathbf{F}_2] \stackrel{\mathrm{def}}{=} \mathbf{F}_1 \triangleleft L_{\mathbf{F}_2,m}$*. Then,* $\mathrm{Adv}^{\mathrm{prf}}_{\mathrm{ARV}_m[\mathbf{F}_1,\mathbf{F}_2]}(q,\tau) \le \mathrm{Adv}^{\mathrm{prf}}_{\mathrm{ARV}^+_m[\mathbf{F}_1,\mathbf{F}_2]}(q,\tau) \le \mathrm{Adv}^{\mathrm{prf}}_{\mathbf{F}_1}(q,\tau) + m \cdot \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}_2}(q,\tau') + \frac{q^2}{2^{n+1}}$.

An advantage of ARV$^+$ over ARV is that the former guarantees an improved throughput for any small m whenever $\mathbf{F}_2$ is faster than $\mathbf{F}_1$.

Smaller key size. Althought the key size of $L_{\mathbf{F},m}$ is large (i.e., m keys), a mode of WPRF proposed by Damgård and Nielsen [6] reduces the key size to $2\log_2 m$. However, we will not discuss the key scheduling issue in this paper.

4 Hybrid Construction of PRP

4.1 Hybrid Double Length PRP is Difficult Within 3-Round Feistel

In this section, we deal with the hybrid construction of a PRP. Our first target is a double length PRP (DLPRP). More specifically, we want to build a PRP on $\mathcal{X}^2$ using a PRF: $\mathcal{X} \to \mathcal{X}$ and a WPRF: $\mathcal{X} \to \mathcal{X}$. It is well known that the cascade of a light-weight mixing and two Feistel rounds where round functions are two independent PRFs is a DLPRP (this was first pointed out by Lucks [14]). To implement the light-weight mixing, no cryptographic functions are needed: one Feistel round using ϵ-AXU [25] with an adequately small ϵ is enough. Here, ϵ-AXU is defined as follows. There have been many proposals for practical and efficient ϵ-AXUs, for instance MMH [9].

Definition 10. *Let* $\mathbf{F} : \mathcal{X} \to \mathcal{Y}$. *If* $\mathbf{F}$ *is* ϵ*-almost XOR universal (*ϵ*-AXU), then* $\Pr\{\mathbf{F}(x) \oplus \mathbf{F}(x') = y\} \leq \epsilon$ *for all* $(x, x') \in \mathcal{X}^2$ *such that* $x \neq x'$ *and all* $y \in \mathcal{Y}$.

Can we substitute one of two PRFs (in DLPRP described above) with some WPRF and maintain the cipher's security? If the answer is yes, we can build a hybrid DLPRP using one invocation of a PRF and a WPRF, and a light-weight mixing round[9].

Unfortunately, the answer is not that clear. At least we found that, some special WPRF could be used as the last round function. The following theorem has a typical example.

Theorem 2. *Let* $\mathbf{G}$ *be* $\mathbf{E} \circ \mathbf{M}[\mathbf{F}_1, \mathbf{F}_2]$, *where* $\mathbf{E}$ *is a* $2n$*-bit RP and* $\mathbf{M}[\mathbf{F}_1, \mathbf{F}_2]$ *is a* 2*-round Feistel (see Ex. 1). Let* S_i *be the* i*-th input to* $\mathbf{F}_1$ *and let* a_i *be* $\mathrm{dist}(S^i)$. *Note that* S_i *is also the left half of the* i*-th output of* $\mathbf{E}$. *Let us assume that* $\mathbf{F}_2(x) = \mathbf{H}(\hat{x})$ *holds for all* x, *where* $\mathbf{H} : \{0,1\}^{n-1} \to \mathcal{X}$, *and* $\hat{x}$ *is the first* $n-1$ *bits of* x. *Then,* $\mathrm{Adv}^{\mathrm{prp}}_{\mathbf{G}}(q, \tau)$ *is at most* $\mu(\mathbf{E}, \overline{a_q}) + \mathrm{Adv}^{\mathrm{prf}}_{\mathbf{F}_1}(q, \tau) + \mathrm{Adv}^{\mathrm{prf}}_{\mathbf{H}}(q, \tau) + q^2/2^n$. *If* $\mathbf{E}$ *is one right-to-left Feistel round with* ϵ*-AXU, then* $\mu(\mathbf{E}, \overline{a_q}) \leq \epsilon q^2/2$.

Proof. The proof is an extension of 3-round Feistel's proof. See Appendix B.

If $\mathbf{H}$ is a PRF, then $\mathbf{F}_2$ is obviously a WPRF, but not a PRF[10]. However, we could not find a way of evaluating the CPA-advantage of the cipher unless $\mathbf{F}_2$ was such a special WPRF (or a PRF). The reason is, roughly saying, that the information of S_i, which is a key to find a collision among S^{i+1}, may not be sufficiently hidden unless $\mathbf{F}_2$ is a PRF or a special WPRF described above. Moreover, the construction in Theorem 2 is not a hybrid one, but only a mode of two PRFs. For now, we think a general security proof based only on the CPA-advantage of $\mathbf{F}_1$ and KPA-advantage of $\mathbf{F}_2$ is intractable.

4.2 New Scheme Providing Hybrid DLPRP

Here, we propose a new block cipher scheme that slightly differs from Feistel.

Definition 11. *For an RP on* $\mathcal{X}$, $\mathbf{C}$, *and* $\mathbf{F} : \mathcal{X} \to \mathcal{X}^{m-1}$, *let* $\mathbf{N}_m[\mathbf{C}, \mathbf{F}]$ *be an RP on* $\mathcal{X}^m$ *defined as* $\mathbf{N}_m[\mathbf{C}, \mathbf{F}](x_l, x_r) = (\mathbf{C}(x_l), \mathbf{F}(\mathbf{C}(x_l)) \oplus x_r)$, *where* $x_l \in \mathcal{X}$ *and* $x_r \in \mathcal{X}^{m-1}$.

Here, $\mathbf{N}_m[\mathbf{C}, \mathbf{F}]$ is clearly invertible if $\mathbf{C}$ is invertible. Note that $\mathbf{N}_m[*, *]$ is unbalanced (i.e., a message is divided into two submessages of unequal lengths) for all $m > 2$. Let us say $\mathbf{N}_m[*, *]$ is $(n, (m-1)n)$ unbalanced. Now, let us show that the double length scheme, $\mathbf{N}_2[*, *]$, has quite a unique property: it provides an efficient hybrid DLPRP that accepts *any* WPRF. The first step in proving this is in analyzing an ideal setting (i.e., when $\mathbf{N}_2[\mathbf{P}_n, \mathbf{R}_{n,n}]$ is used).

[9] Building a DLPRP using two WPRF calls seems impossible, though we have not formally proved this so far.

[10] Interestingly, this kind of WPRF was proposed in Lucks's "Faster Luby-Rackoff Cipher" [14], although he only proved its non-adaptive CPA-security.

Theorem 3. *Let* $\mathbf{G}$ *be* $\mathbf{E}\circ\mathbf{N}_2[\mathbf{P}_n, \mathbf{R}_{n,n}]$*, where* $\mathbf{E}$ *is an RP on* $\mathcal{X}^2$*. Let* S_i *denote the i-th input to* $\mathbf{P}_n$*, which corresponds to the left half of the i-th output of* $\mathbf{E}$*. We then obtain*

$$\mathrm{Adv}^{\mathrm{prp}}_{\mathbf{G}}(q,\infty) \le \mu(\mathbf{E}, \overline{a_q}) + \frac{q^2}{2^{n+2}}, \quad \text{where } a_q \text{ denotes } \mathrm{dist}(S^q). \tag{4}$$

Moreover, if $\mathbf{E}$ *is one right-to-left Feistel round with* ϵ*-AXU,* $\mathbf{H} : \mathcal{X} \to \mathcal{X}$ *(see left of Fig. 1), then* $\mathrm{Adv}^{\mathrm{prp}}_{\mathbf{G}}(q,\infty)$ *is at most* $\frac{q^2}{2}\left(\epsilon + \frac{1}{2^{n+1}}\right)$*.*

Proof. The core of the proof is in the following lemma. This is proved in Appendix C.

Lemma 4. *For any RF:*$\mathcal{X}^2 \to \mathcal{X}^2$*, let* (S_i, T_i) *be the i-th input, where* $S_i, T_i \in \mathcal{X}$*. Similarly,* (U_i, V_i) *denotes the i-th output, where* $U_i, V_i \in \mathcal{X}$*. For the case of* $\mathbf{N}_2[\mathbf{P}_n, \mathbf{R}_{n,n}]$*,* $U_i = \mathbf{P}_n(S_i)$ *and* $V_i = \mathbf{R}_{n,n}(\mathbf{P}_n(S_i))\oplus T_i$*. Let* a_i *and* b_i *be* $\mathrm{dist}(S^i)$ *and* $\mathrm{dist}(U^i)$*, respectively. For two MESs,* $\mathcal{A} = a_0 a_1 \ldots$ *and* $\mathcal{B} = b_0 b_1 \ldots$*,*

$$\mathbf{N}_2[\mathbf{P}_n, \mathbf{R}_{n,n}]^{\mathcal{A}\wedge\mathcal{B}\wedge\mathcal{C}} \equiv \mathbf{P}_{2n}^{\mathcal{A}\wedge\mathcal{B}} \tag{5}$$

holds for some MES $\mathcal{C} = c_0 c_1 \ldots$ *defined for* $\mathbf{N}_2[\mathbf{P}_n, \mathbf{R}_{n,n}]$*.*

Let us abbreviate $\mathbf{N}_2[\mathbf{P}_n, \mathbf{R}_{n,n}]$ to $\mathbf{N}_2^*$. Using Lemmas 4 and 5, we obtain

$$(\mathbf{E}\circ\mathbf{N}_2^*)^{\mathcal{A}\wedge\mathcal{B}\wedge\mathcal{C}} \equiv (\mathbf{E}\circ\mathbf{P}_{2n})^{\mathcal{A}\wedge\mathcal{B}}, \tag{6}$$

where MESs are defined for $\mathbf{N}_2^*$ and $\mathbf{P}_{2n}$ (see the left of Fig. 1). Let $\widehat{\mathbf{P}}_{2n}$ be a random system compatible to $\mathbf{P}_{2n}$, which always behaves as if some distinct inputs are given to $\mathbf{P}_{2n}$, no matter what the actual inputs are. We then have

$$\mathrm{Adv}^{\mathrm{prp}}_{\mathbf{E}\circ\mathbf{N}_2^*}(q,\infty) \le \nu(\mathbf{E}\circ\mathbf{P}_{2n}, \overline{a_q}\vee\overline{b_q}) \le \mu(\mathbf{E}, \overline{a_q}) + \mu(\widehat{\mathbf{P}}_{2n}, \overline{b_q}). \tag{7}$$

In Eq. (7), the first inequality follows from the equivalence $\mathbf{E}\circ\mathbf{P}_{2n} \equiv \mathbf{P}_{2n}$, and Lemma 4, and Theorem 6. For the last inequality, note that $\mathbf{P}_{2n}^{\mathcal{B}}|\mathcal{A} \equiv \widehat{\mathbf{P}}_{2n}^{\mathcal{B}}$ holds, since $\mathcal{A}$ indicates that inputs to $\mathbf{P}_{2n}$ are distinct and $\mathcal{B}$ is defined for outputs. Applying $\mathbf{P}_{2n}^{\mathcal{B}}|\mathcal{A} \equiv \widehat{\mathbf{P}}_{2n}^{\mathcal{B}}$ to Lemma 10 proves the last inequality. Let us analyze $\mu(\widehat{\mathbf{P}}_{2n}, \overline{b_q})$, which corresponds to the probability of a collision occurring in the left halves of $\widehat{\mathbf{P}}_{2n}$'s outputs. For any $i \neq j$, we obtain

$$\begin{aligned} P^{\widehat{\mathbf{P}}_{2n}}(U_i = U_j) &= \sum_{u,v_i,v_j\in\mathcal{X},v_i\neq v_j} P^{\widehat{\mathbf{P}}_{2n}}(U_i = U_j = u, V_i = v_i, V_j = v_j) \\ &= \sum_{u,v_i,v_j\in\mathcal{X},v_i\neq v_j} \frac{1}{2^{2n}}\cdot\frac{1}{2^{2n}-1} = 2^n\cdot\frac{2^n(2^n-1)}{2\cdot 2^{2n}\cdot 2^{2n}-1} < \frac{1}{2^{n+1}}. \end{aligned}$$

This means $\mu(\widehat{\mathbf{P}}_{2n}, \overline{b_q})$ is less than $\binom{q}{2}\frac{1}{2^{n+1}} < \frac{q^2}{2^{n+2}}$. Substituting $\mu(\widehat{\mathbf{P}}_{2n}, \overline{b_q})$ with $\frac{q^2}{2^{n+2}}$ in Eq. (7) proves the first claim. As the proof of the second claim is easily derived by the first claim and a trivial collision analysis of $\mathbf{E}$, we omitted it.

The CPA-security of $\mathbf{E} \circ \mathbf{N}_2[\mathbf{C}, \mathbf{F}]$ is easy to prove, when both $\mathbf{C}$ and $\mathbf{F}$ are CPA-secure. Here, we will present a stronger result: we only need the KPA-security of $\mathbf{F}$ and CPA-security of $\mathbf{C}$.

Theorem 4. *Let* $\mathbf{G} = \mathbf{E} \circ \mathbf{N}_2[\mathbf{C}, \mathbf{F}]$*, where* $\mathbf{E}$ *is an RP on* $\mathcal{X}^2$*,* $\mathbf{C}$ *is an RP on* $\mathcal{X}$*, and* $\mathbf{F} : \mathcal{X} \to \mathcal{X}$*. Then,*

$$\mathrm{Adv}^{\mathrm{prp}}_{\mathbf{G}}(q, \tau) \leq \mu(\mathbf{E}, \overline{a_q}) + \mathrm{Adv}^{\mathrm{prp}}_{\mathbf{C}}(q, \tau) + \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q, \tau') + \frac{5q^2}{2^{n+2}}, \qquad (8)$$

where a_q *denotes an event where* q *inputs to* $\mathbf{C}$ *are distinct.*

Proof. Using triangle inequality, $\mathrm{Adv}^{\mathrm{prp}}_{\mathbf{G}}(q, \tau)$ is no more than $\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{G}, \mathbf{G}'}(q, \tau) + \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{G}', \mathbf{G}^*}(q, \tau) + \mathrm{Adv}^{\mathrm{prp}}_{\mathbf{G}^*}(q, \tau)$, where $\mathbf{G}'$ and $\mathbf{G}^*$ denote $\mathbf{E} \circ \mathbf{N}_2[\mathbf{P}_n, \mathbf{F}]$ and $\mathbf{E} \circ \mathbf{N}_2[\mathbf{P}_n, \mathbf{R}_{n,n}]$, respectively. Note that $\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{G}', \mathbf{G}^*}(q, \tau) \leq \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{P}_n \triangleleft \mathbf{F}, \mathbf{P}_n \triangleleft \mathbf{R}_{n,n}}(q, \tau)$ and thus we can use Lemma 1. This observation and Theorem 3 complete the proof.

As Theorem 4 shows, if $\mathbf{C}$ is a PRP and $\mathbf{F}$ is a WPRF, the cascade of light-weight mixing and $\mathbf{N}_2[\mathbf{C}, \mathbf{F}]$ is a DLPRP. Unlike a 3-round Feistel, no additional conditions are needed for $\mathbf{F}$.

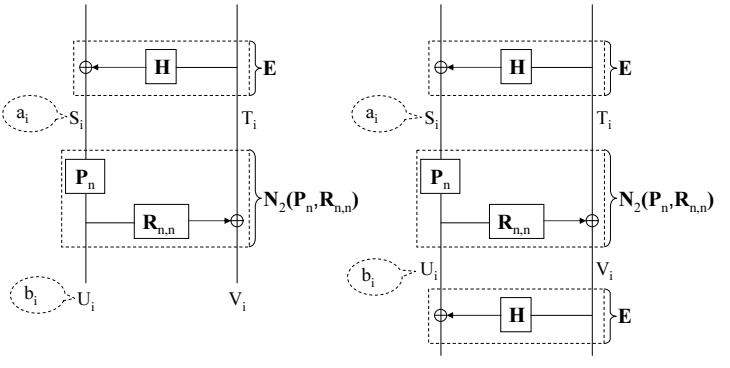


Fig. 1. Our Double Length PRP (left) and Double Length SPRP (right)

4.3 Achieving Large Block Size

One notable property of our scheme is that it offers a very efficient way of extending block size. We can use $\mathbf{N}_m[*, *]$ to build an mn-bit block cipher. Here, we need a WPRF:$\mathcal{X} \to \mathcal{X}^{m-1}$ for the second argument of $\mathbf{N}_m[*, *]$. Such an RF can be composed from any WPRF:$\mathcal{X} \to \mathcal{X}$, as shown in Lemma 3.

Corollary 2. *Let* $\mathbf{C}$ *be an RP on* $\mathcal{X}$*, and let* $\mathbf{F}$ *be an RF*: $\mathcal{X} \to \mathcal{X}$*. Moreover, let* $\mathbf{E}$ *be a right-to-left,* $(n, (m-1)n)$ *unbalanced Feistel round with* ϵ*-AXU. Then,*

$$\mathrm{Adv}^{\mathrm{prp}}_{\mathbf{E} \circ \mathbf{N}_m[\mathbf{C}, L_{\mathbf{F}, m-1}]}(q, \tau) \leq \mathrm{Adv}^{\mathrm{prp}}_{\mathbf{C}}(q, \tau) + (m-1) \cdot \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q, \tau') + q^2 \left(\frac{\epsilon}{2} + \frac{5}{2^{n+2}} \right).$$

Proof. The logic of the proof is the same as with DLPRP. Consider $\mathbf{E} \circ \mathbf{G}$, where $\mathbf{G}$ is an mn-bit RP. Assume S_i is the leftmost n-bit of the i-th input to $\mathbf{G}$, and U_i is the leftmost n-bit of the i-th output of $\mathbf{G}$. Let a_i and b_i denote $\mathrm{dist}(S^i)$ and $\mathrm{dist}(U^i)$. We can then prove that $\mathrm{Adv}^{\mathrm{prp}}_{\mathbf{E}\circ\mathbf{N}_m[\mathbf{P}_n,\mathbf{R}_{n,(m-1)n}]}(q,\infty)$ is at most $\mu(\mathbf{E},\overline{a_q})+q^2/2^{n+2}$ in almost the same way as with the proof of Theorem 3. From this and Theorem 4,

$$\mathrm{Adv}^{\mathrm{prp}}_{\mathbf{E}\circ\mathbf{N}_m[\mathbf{C},L_{\mathbf{F},m-1}]}(q,\tau) \le \mu(\mathbf{E},\overline{a_q}) + \mathrm{Adv}^{\mathrm{prp}}_{\mathbf{C}}(q,\tau) + \mathrm{Adv}^{\mathrm{wprf}}_{L_{\mathbf{F},m-1}}(q,\tau') + \frac{5q^2}{2^{n+2}} \tag{9}$$

is obtained. Here, $\mu(\mathbf{E},\overline{a_q}) \le \epsilon q^2/2$ holds, if $\mathbf{E}$ is an unbalanced Feistel with ϵ-AXU. From this observation, and Eq. (9), and Lemma 3, Corollary 2 is proved.

To implement an efficient large block cipher with our scheme, the domain of ϵ-AXU needs to be easily expanded. Most practical AXUs have this property.

5 Hybrid Construction of SPRP

Similar to the 3-round Feistel, our hybrid PRP is completely vulnerable to CCA. However, small additions to our scheme can yield an SPRP. This is a very similar approach to that presented by Naor and Reingold [25]. Our SPRP construction is based on the following theorem.

Theorem 5. *Let* $\mathbf{Q}$ *be* $\mathbf{E}_1 \circ \mathbf{N}_2[\mathbf{P}_n,\mathbf{R}_{n,n}] \circ \mathbf{E}_2^{-1}$*, where* $\mathbf{E}_1$ *and* $\mathbf{E}_2$ *are independent RPs on* $\mathcal{X}^2$*. For any* $\mathbf{E}_1 \circ \mathbf{G} \circ \mathbf{E}_2^{-1}$*, where* $\mathbf{G}$ *is an RP on* $\mathcal{X}^2$*, let* (S_i,T_i) *be the* i*-th input to* $\mathbf{G}$*. Similarly, let* (U_i,V_i) *be the* i*-th output of* $\mathbf{G}$*. Let* a_i *and* b_i *denote* $\mathrm{dist}(S^i)$ *and* $\mathrm{dist}(U^i)$*, respectively. Then,*

$$\mathrm{Adv}^{\mathrm{sprp}}_{\mathbf{Q}}(q,\infty) \le \mu(\mathbf{E}_1,\overline{a_q}) + \mu(\mathbf{E}_2,\overline{b_q}). \tag{10}$$

In addition, let $\mathbf{Q}'$ *be* $\mathbf{E}\circ\mathbf{N}_2[\mathbf{P}_n,\mathbf{R}_{n,n}]\circ\mathbf{E}$*, where* $\mathbf{E}$ *is a* $2n$*-bit right-to-left Feistel with* ϵ*-AXU,* $\mathbf{H}:\mathcal{X}\to\mathcal{X}$ *(see the right of Fig. 1). Then* $\mathrm{Adv}^{\mathrm{sprp}}_{\mathbf{Q}'}(q,\infty) \le \epsilon q^2$*.*

Proof. See Appendix D.

As well as Sect. 4.3, Theorem 5 can easily be generalized to an mn-bit block size. In this case, the second argument of $\mathbf{N}_m[*,*]$ has to be a PRF:$\mathcal{X}\to\mathcal{X}^{m-1}$.

Corollary 3. *Let* $\mathbf{G}$ *be* $\mathbf{E}\circ\mathbf{N}_m[\mathbf{C},\mathbf{F}]\circ\mathbf{E}$*, where* $\mathbf{E}$ *is an* mn*-bit right-to-left* $(n,(m-1)n)$ *unbalanced Feistel with* ϵ*-AXU,* $\mathbf{H}:\mathcal{X}^{m-1}\to\mathcal{X}$*, and* $\mathbf{C}$ *is an RP on* $\mathcal{X}$ *and* $\mathbf{F}:\mathcal{X}\to\mathcal{X}^{m-1}$*. Then,*

$$\mathrm{Adv}^{\mathrm{sprp}}_{\mathbf{G}}(q,\tau) \le \mathrm{Adv}^{\mathrm{sprp}}_{\mathbf{C}}(q,\tau) + \mathrm{Adv}^{\mathrm{prf}}_{\mathbf{F}}(q,\tau) + \epsilon q^2. \tag{11}$$

Proof. Let $\mathbf{Q}$ be $\mathbf{E}\circ\mathbf{N}_m[\mathbf{P}_n,\mathbf{R}_{n,(m-1)n}]\circ\mathbf{E}$. Then, $\mathrm{Adv}^{\mathrm{sprp}}_{\mathbf{Q}}(q,\infty) \le \epsilon q^2$ can be proved in the same way as with the proof of Theorem 5, as we can use the same MESs as Theorem 5 (i.e., collisions in the leftmost n-bit of $\mathbf{N}_m[\mathbf{P}_n,\mathbf{R}_{n,(m-1)n}]$'s inputs and outputs). Corollary 3 follows from this and triangle inequality.

Suppose that an SPRP on $\mathcal{X}$, $\mathbf{C}$, and a WPRF, $\mathbf{F} : \mathcal{X} \to \mathcal{X}$, are available. Here, we first generate two independent versions of $\mathbf{C}$, $\mathbf{C}_1$ and $\mathbf{C}_2$, and build an mn-bit hybrid block cipher $\mathbf{E} \circ \mathbf{N}_m[\mathbf{C}_1, \mathrm{ARV}^+_{m-2}[\mathbf{C}_2, \mathbf{F}]] \circ \mathbf{E}$, where $\mathbf{E}$ is an $(n, (m-1)n)$ unbalanced Feistel with ϵ-AXU (see Fig. 2 in Appendix E). From Corollaries 1 and 3, this cipher is proved to be SPRP. It only requires two invocations of SPRP, $(m-2)$ invocations of WPRF, and two invocations of ϵ-AXU:$\mathcal{X}^{m-1} \to \mathcal{X}$ for any $m > 2$.

6 Summary

For comparison, we considered the NR mode [26]. It uses m invocations of an SPRP on $\mathcal{X}$ and two mixing layers on $\mathcal{X}^m$ to provide an mn-bit block SPRP. These mixing layers are composed of independent AXUs and slightly more complicated than ours. For $m > 2$, NR mode is very close to the best if an n-bit SPRP is the only cryptographic component available (and a fast AXU is available[11]), since it would be impossible to have an m-block pseudorandom output without using m SPRP invocations. Furthermore, it is easily verified that if one mixing layer is omitted from the NR mode, the resulting mode, which is denoted by the NR$^-$ mode, is a PRP, if the underlying component is a PRP. This is a highly optimized large block PRP construction based on small PRP and AXU.

Table 1. The number of component calls for hybrid and previous modes. All components are n-bit block, except for AXU. $\mathrm{AXU}_{\alpha,\beta}$ denotes AXU: $\{0,1\}^\alpha \to \{0,1\}^\beta$.

DLPRP	PRP	WPRF	$\mathrm{AXU}_{2n,n}$	others
Hybrid (left of Fig.1)	1	1	1	-
3-round Feistel	2	0	1	-
mn-bit PRP ($m > 2$)	PRP	WPRF	$\mathrm{AXU}_{n(m-1),n}$	others
Hybrid (left of Fig.2)	1	$m-1$	1	-
NR$^-$ mode	m	0	1	some additional AXU calls
mn-bit SPRP ($m > 2$)	SPRP	WPRF	$\mathrm{AXU}_{n(m-1),n}$	others
Hybrid (right of Fig.2)	2	$m-2$	2	-
NR mode	m	0	2	some additional AXU calls

As Table 1 shows, our hybrid modes performs quite well for both small and large blocks. In addition, they have comparable parallelism to that of the NR (or NR$^-$) mode, due to the high parallelism of $L_{\mathbf{F},m}$. The implementation cost of hybrid mode is naturally higher than that of the NR mode, but the additional cost would be small, or, at least smaller than the implemention cost of another PRF, since WPRF is a "cheaper" primitive than PRF.

Several options can be considered to implement our proposals. A promising approach is to combine the AES and a stream cipher that accepts IVs and is

[11] If a mode without AXU is desirable, EME or CMC modes [10, 11] are used.

faster than AES. For example, some stream ciphers proposed for the recent ECRYPT project [34] have this property. Such a stream cipher can be used as $\mathbf{F}$ in the $L_{\mathbf{F},t}$ construction. We can even use it directly as a substitute for $L_{\mathbf{F},t}$. In addition, we can save the implementation cost if the stream cipher is based on AES (an example of this is LEX [4]). Of course, we have to carefully check if our stream cipher is adequately secure. In this case, stream ciphers must be secure against attacks using many random IVs and corresponding (short) keystreams. These attacks are classified as a kind of resynchronization attack [5] and well-considered stream ciphers would be immune from them.

References

1. W. Aiello, S. Rajagopalan, and R. Venkatesan. "High-Speed Pseudorandom Number Generation with Small Memory." *Fast Software Encryption*, FSE'99, LNCS 1636, pp. 290-304, 1999.
2. R. Anderson and E. Biham. "Two Practical and Provably Secure Block Ciphers: BEAR and LION." *Fast Software Encryption*, FSE'96, LNCS 1039, pp. 113-120, 1996.
3. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. " A Concrete Security Treatment of Symmetric Encryption." *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, FOCS '97, pp. 394-403, 1997.
4. A. Biryukov. "New 128-bit Key Stream Cipher LEX." *ECRYPT Stream Cipher Project Report*, available from `http://www.ecrypt.eu.org/stream/`
5. J. Daemen, R. Govaerts, J. Vandewalle. "Resynchronization Weaknesses in Synchronous Stream Ciphers." *Advances in Cryptology*- EUROCRYPT'93, LNCS 765, pp. 159-167, 1993.
6. I. Damgård and J. Nielsen. "Expanding Pseudorandom Functions; or: From Known-Plaintext Security to Chosen-Plaintext Security." *Advances in Cryptology*- CRYPTO'02, LNCS 2442, pp. 449-464, 2002.
7. O. Goldreich, S. Goldwasser, and S. Micali. "How to Construct Random Functions." *Journal of the ACM*, Vol. 33, No. 4, pp. 792-807, 1986.
8. O. Goldreich. "Modern Cryptography, Probabilistic Proofs and Pseudorandomness." *Springer*.
9. S. Halevi and H. Krawczyk. "MMH:Software Message Authentication in the Gbit/second rates." *Fast Software Encryption*, FSE'97, LNCS 1267, pp. 172-189, 1997.
10. S. Halevi and P. Rogaway. "A Tweakable Enciphering Mode." *Advances in Cryptology* - CRYPTO'03, LNCS 2729, pp. 482-499, 2003.
11. S. Halevi and P. Rogaway. "A Parallelizable Enciphering Mode." *Topics in Cryptology*- CT-RSA'04, LNCS 2964, pp. 292-304, 2004.
12. T. Iwata and K. Kurosawa. "On the Universal Hash Functions in Luby-Rackoff Cipher." *Information Security and Cryptology*- ICISC'02, pp. 226-236, LNCS 2587, 2003.
13. M. Luby and C. Rackoff. "How to Construct Pseudo-random Permutations from Pseudo-random functions." *SIAM J. Computing*, Vol. 17, No. 2, pp. 373-386, 1988.
14. S. Lucks. "Faster Luby-Rackoff Ciphers." *Fast Software Encryption*, FSE'96, LNCS 1039, pp. 189-203. 1996.
15. U. Maurer. "A Simplified and Generalized Treatment of Luby-Rackoff Pseudorandom Permutation Generators." *Advances in Cryptology*- EUROCRYPT'92, pp. 239-255, 1992.

16. U. Maurer and J. L. Massey. "Cascade Ciphers: The Importance of Being First." *J. Cryptology* Vol. 6, num.1, pp. 55-61, 1993.
17. U. Maurer. "Indistinguishability of Random Systems." *Advances in Cryptology*- EUROCRYPT'02, LNCS 2332, pp. 110-132, 2002.
18. U. Maurer and K. Pietrzak. "Composition of Random Systems: When Two Weak Make One Strong." *Theory of Cryptography* - TCC'04, LNCS 2951, pp. 410-427, 2004.
19. M. Minier and H. Gilbert. "New Results on the Pseudorandomness of Some Block-cipher Constructions." *Fast Software Encryption*, FSE'01, LNCS 2355, pp. 248-266, 2002.
20. S. Moriai and S. Vaudenay. "On the Pseudorandomness of Top-Level Schemes of Block Ciphers." *Advances in Cryptology* - ASIACRYPT'00, LNCS 1976, pp. 289-302, 2000.
21. S. Myers. "Black-Box Composition Does Not Imply Adaptive Security." *Advances in Cryptology*- EUROCRYPT'04, LNCS 3027, pp. 189-206, 2004.
22. M. Naor and O. Reingold. "Number-theoretic Constructions of Efficient Pseudo-random Functions." 38 *th Annual Symposium on Foundations of Computer Science*, FOCS'97, pp. 458-467, 1997.
23. M. Naor and O. Reingold. "From Unpredictability to Indistinguishability: A Simple Construction of Pseudo-Random Functions from MACs (Extended Abstract)." *Advances in Cryptology* - CRYPTO'98, LNCS 1462, pp. 267-282, 1998.
24. M. Naor and O. Reingold. "Synthesizers and their application to the parallel construction of pseudo-random functions." *J. of Computer and Systems Sciences*, Vol. 58 (2), pp. 336-375, 1999.
25. M. Naor and O. Reingold. "On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited." *Journal of Cryptology*, Vol. 12 (1), pp. 29-66, 1999.
26. M. Naor and O. Reingold. "The NR Mode of Operation." *Manuscript*, available from `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/nr-mode.ps`
27. S. Patel, Z. Ramzan, and G. Sundaram. "Towards Making Luby-Rackoff Ciphers Optimal and Practical." *Fast Software Encryption*, FSE'99, LNCS 1636, pp. 171-185, 1999.
28. J. Patarin. "Security of Random Feistel Schemes with 5 or More Rounds." *Advances in Cryptology* - CRYPTO'04, LNCS 3152, pp. 106-122, 2004.
29. S. Vaudenay. "Feistel Ciphers with L_2-Decorrelation." *Selected Areas in Cryptography* - SAC'98, LNCS 1556, pp. 1-14, 1998.
30. S. Vaudenay. "Provable Security for Block Ciphers by Decorrelation." *15th Annual Symposium on Theoretical Aspects of Computer Science*- STACS '98, LNCS 1373, pp. 249-275, 1998.
31. S. Vaudenay. "Adaptive-Attack Norm for Decorrelation and Super-Pseudorandomness." *Selected Areas in Cryptography* - SAC'00, LNCS 1758, pp. 49-61, 2000.
32. M. Wegman and L. Carter. "New Hash Functions and Their Use in Authentication and Set Equality." *Journal of Computer and System Sciences*, Vol. 22, pp. 265-279, 1981.
33. Y. Zheng, T. Matsumoto, and H. Imai. "On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses." *Advances in Cryptology* - CRYPTO'89, LNCS 435, pp. 461-480, 1990.
34. ECRYPT Stream Cipher Project. `http://www.ecrypt.eu.org/stream/`
35. Security in Storage Working Group, An IEEE Information Assurance Activity. `http://www.siswg.org`

A Theorems and Lemmas Proved by Maurer [17]

Let us now describe some of Maurer's results [17]. They were used in our analysis.

Theorem 6. *(Theorem* 1 (i) *of [17]) Let* $\mathcal{A}$ *and* $\mathcal{B}$ *be MESs defined for* $\mathbf{F}$ *and* $\mathbf{G}$*. If* $\mathbf{F}^{\mathcal{A}} \equiv \mathbf{G}^{\mathcal{B}}$ *or* $\mathbf{F}|\mathcal{A} \equiv \mathbf{G}$*, then* $\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{F},\mathbf{G}}(q,\infty) \leq \nu(\mathbf{F}, \overline{a_q})$*.*

Theorem 7. *(Theorem* 7 *of [17]) Let* $\mathbf{G}$ *be* $\mathbf{E} \circ \mathbf{M}[\mathbf{R}^{(1)}, \mathbf{R}^{(2)}]$*, where* $\mathbf{E}$ *is an RP on* $\mathcal{X}^2$*,* $\mathbf{R}^{(1)}$ *and* $\mathbf{R}^{(2)}$ *are independent URFs:*$\mathcal{X} \to \mathcal{X}$*. Here,* $\mathbf{M}[*,*]$ *is a* $2n$*-bit* 2*-round Feistel, as described in Ex. 1. Let* a_q *denote an event where* q *inputs to* $\mathbf{R}^{(1)}$ *are distinct. Then* $\mathrm{Adv}^{\mathrm{prp}}_{\mathbf{G}}(q,\infty) \leq \mu(\mathbf{E}, \overline{a_q}) + \frac{q^2}{2^{n+1}}$*.*

Lemma 5. *(A corollary from Lemma* 4 (ii) *of [17]) Let* $\mathbf{F}$ *and* $\mathbf{G}$ *be two compatible RFs. If* $\mathbf{F}^{\mathcal{A}} \equiv \mathbf{G}^{\mathcal{B}}$ *for MESs* $\mathcal{A}$ *and* $\mathcal{B}$*, then* $(\mathbf{E}_1 \circ \mathbf{F} \circ \mathbf{E}_2)^{\mathcal{A}} \equiv (\mathbf{E}_1 \circ \mathbf{G} \circ \mathbf{E}_2)^{\mathcal{B}}$ *holds true, as long as* $(\mathbf{E}_1, \mathbf{E}_2)$ *is independent of* $\mathbf{F}$ *and* $\mathbf{G}$*. Here,* $\mathbf{E}_1$ *and* $\mathbf{E}_2$ *are not necessarily independent of each other.*

Lemma 6. *(Lemma* 1 (iv) *of [17]) Let MESs* $\mathcal{A}$ *and* $\mathcal{B}$ *be defined for* $\mathbf{F}$ *and* $\mathbf{G}$*. Moreover, let* X_i *and* Y_i *denote the* i*-th input and output of* $\mathbf{F}$ *(or* $\mathbf{G}$*), respectively. Assume* $\mathbf{F}|\mathcal{A} \equiv \mathbf{G}|\mathcal{B}$*. If* $P^{\mathbf{F}}_{a_i|X^iY^{i-1}a_{i-1}} \leq P^{\mathbf{G}}_{b_i|X^iY^{i-1}b_{i-1}}$ *for* $i \geq 1$*, which means* $P^{\mathbf{F}}_{a_i|X^iY^{i-1}a_{i-1}}(x^i, y^{i-1}) \leq P^{\mathbf{G}}_{b_i|X^iY^{i-1}b_{i-1}}(x^i, y^{i-1})$ *holds for all* (x^i, y^{i-1}) *such that* $P^{\mathbf{F}}_{a_{i-1}|X^{i-1}Y^{i-1}}(x^{i-1}, y^{i-1})$ *and* $P^{\mathbf{G}}_{b_{i-1}|X^{i-1}Y^{i-1}}(x^{i-1}, y^{i-1})$ *are positive, then there exists an MES* $\mathcal{C}$ *defined for* $\mathbf{G}$ *such that* $\mathbf{F}^{\mathcal{A}} \equiv \mathbf{G}^{\mathcal{B}\wedge\mathcal{C}}$*.*

Lemma 7. *(Lemma* 6 (ii) *of [17]) If* $\mathbf{F}^{\mathcal{A}} \equiv \mathbf{G}^{\mathcal{B}}$*, then* $\nu(\mathbf{F}, \overline{a_q}) = \nu(\mathbf{G}, \overline{b_q})$*.*

Lemma 8. *(Lemma* 6 (iii) *of [17])* $\nu(\mathbf{F}, \overline{a_q} \vee \overline{b_q}) \leq \nu(\mathbf{F}, \overline{a_q}) + \nu(\mathbf{F}, \overline{b_q})$ *if* $\mathcal{A}$ *and* $\mathcal{B}$ *are defined for* $\mathbf{F}$*.*

Lemma 9. *(Lemma* 10 (iii) *of [17]) For any two compatible RPs,* $\mathbf{F}$ *and* $\mathbf{G}$*,* $\mathbf{F}^{\mathcal{A}} \equiv \mathbf{G}^{\mathcal{B}}$ *implies* $\langle \mathbf{F} \rangle^{\mathcal{A}} \equiv \langle \mathbf{G} \rangle^{\mathcal{B}}$*.*

Lemma 10. *(Corollary* 1 (v) *of [17]) If* a_i *(*b_i*) is defined on the inputs (outputs) of* $\mathbf{F}$ *and* $\mathbf{F}^{\mathcal{B}}|\mathcal{A} \equiv \mathbf{U}^{\mathcal{B}}$ *for a source* $\mathbf{U}$ *compatible to* $\mathbf{F}$*, then* $\nu(\mathbf{E} \circ \mathbf{F}, \overline{a_q} \vee \overline{b_q}) \leq \mu(\mathbf{E}, \overline{a_q}) + \mu(\mathbf{U}, \overline{b_q})$ *for any* $\mathbf{E}$*. Here, a source is a random system that generates outputs that are independent of corresponding inputs.*

B Proof of Theorem 2

The proof of Theorem 2 is basically the same as the tight security proof of the 3-round Feistel demonstrated by Maurer (Theorem 7 of [17]). The only difference is in the definitions of MESs. Let $\tilde{\mathbf{R}}_{n,n}$ be the RF:$\mathcal{X} \to \mathcal{X}$ defined as $\tilde{\mathbf{R}}_{n,n}(x) = \mathbf{R}_{n-1,n}(\tilde{x})$, where $\tilde{x}$ denotes the first $n-1$ bits of x. First, we prove

$$\mathrm{Adv}^{\mathrm{prp}}_{\mathbf{E}\circ\mathbf{M}[\mathbf{R}_{n,n},\tilde{\mathbf{R}}_{n,n}]}(q,\infty) \leq \mu(\mathbf{E}, \overline{a_q}) + \frac{q^2}{2^n}, \tag{12}$$

where a_q denotes an event where q inputs to $\mathbf{R}_{n,n}$ (i.e., the left halves of $\mathbf{E}$'s outputs) are distinct. As well as Theorem 3, $\mu(\mathbf{E}, \overline{a_q})$ corresponds to the maximal probability of a collision occurring in the left halves of $\mathbf{E}$'s outputs, for all non-adaptive attackers.

For any RF:$\mathcal{X}^2 \to \mathcal{X}^2$, let (S_i, T_i) be the i-th input and let (U_i, V_i) be the i-th output, where $S_i, T_i, U_i, V_i \in \mathcal{X}$. Let a_i and b_i denote $\mathrm{dist}(S^i)$ and $\mathrm{dist}(\widetilde{V}^i)$, where $\widetilde{V}^i$ denotes $(\widetilde{V}_1, \ldots, \widetilde{V}_i)$ and $\widetilde{V}_i$ is the first $n-1$ bits of V_i. The first step is to show that

$$\mathbf{M}[\mathbf{R}_{n,n}, \widetilde{\mathbf{R}}_{n,n}]^{\mathcal{A}\wedge\mathcal{B}} \equiv \widehat{\mathbf{R}}_{2n,2n}^{\mathcal{A}\wedge\mathcal{B}} \equiv \mathbf{R}_{2n,2n}^{\mathcal{A}\wedge\mathcal{B}} \equiv \mathbf{P}_{2n}^{\mathcal{A}\wedge\mathcal{B}\wedge\mathcal{C}} \tag{13}$$

holds for some MES $\mathcal{C}$ defined for $\mathbf{P}_{2n}$. Here, $\widehat{\mathbf{R}}_{2n,2n}$ behaves just like $\mathbf{R}_{2n,2n}$ taking some distinct inputs, independent of actual inputs (i.e., it always outputs uniformly random and independent values). Recall that $\widetilde{\mathbf{R}}_{n,n}$ behaves just like $\mathbf{R}_{n,n}$, as long as the first $n-1$ bits of the inputs do not include collisions. From this, we observe that

$$P^{\mathbf{M}[\mathbf{R}_{n,n}, \widetilde{\mathbf{R}}_{n,n}]}_{U_i V_i | S^i T^i U^{i-1} V^{i-1} a_i b_i} \tag{14}$$

is a uniform distribution on $\mathcal{X} \times \tilde{\mathcal{X}}$, where $\tilde{\mathcal{X}}$ is a set of $v_i \in \mathcal{X}$ satisfying $\mathrm{dist}(\tilde{v}^i)$ (i.e., the first $n-1$ bits of v^i are unique). We thus have

$$\mathbf{M}[\mathbf{R}_{n,n}, \widetilde{\mathbf{R}}_{n,n}] | \mathcal{A} \wedge \mathcal{B} \equiv \widehat{\mathbf{R}}_{2n,2n} | \mathcal{A} \wedge \mathcal{B} \equiv \mathbf{R}_{2n,2n} | \mathcal{A} \wedge \mathcal{B}, \tag{15}$$

which immediately means

$$\mathbf{M}[\mathbf{R}_{n,n}, \widetilde{\mathbf{R}}_{n,n}]^{\mathcal{B}} | \mathcal{A} \equiv \widehat{\mathbf{R}}_{2n,2n}^{\mathcal{B}} | \mathcal{A} \equiv \mathbf{R}_{2n,2n}^{\mathcal{B}} | \mathcal{A}. \tag{16}$$

Using Eq. (16) and the fact that a_i is defined on the inputs, we obtain Eq. (13) except for the last equivalence. For the last equivalence, we observe that

$$P^{\mathbf{R}_{2n,2n}}_{a_i b_i | X^i Y^{i-1} a_{i-1} b_{i-1}} \leq P^{\mathbf{P}_{2n}}_{a_i b_i | X^i Y^{i-1} a_{i-1} b_{i-1}} \tag{17}$$

holds. The inequality above can easily be derived from the definitions of URF and URP. Applying Lemma 6 to Eq. (17), we obtain the last equivalence of Eq. (13).

Next, we apply Lemma 5 to Eq. (13). We then have

$$(\mathbf{E} \circ \mathbf{M}[\mathbf{R}_{n,n}, \widetilde{\mathbf{R}}_{n,n}])^{\mathcal{A}\wedge\mathcal{B}} \equiv (\mathbf{E} \circ \widehat{\mathbf{R}}_{2n,2n})^{\mathcal{A}\wedge\mathcal{B}} \equiv (\mathbf{E} \circ \mathbf{P}_{2n})^{\mathcal{A}\wedge\mathcal{B}\wedge\mathcal{C}} \tag{18}$$

for some MES $\mathcal{C}$. Let $\mathbf{G}$ denote $\mathbf{E} \circ \mathbf{M}[\mathbf{R}_{n,n}, \widetilde{\mathbf{R}}_{n,n}]$. We now have

$$\mathrm{Adv}^{\mathrm{prp}}_{\mathbf{G}}(q, \infty) = \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{G}, \mathbf{E}\circ\mathbf{P}_{2n}}(q, \infty) \leq \nu(\mathbf{G}, \overline{a_q} \vee \overline{b_q}) = \nu(\mathbf{E} \circ \widehat{\mathbf{R}}_{2n,2n}, \overline{a_q} \vee \overline{b_q}), \tag{19}$$

where the inequality follows from Eq. (18) and Theorem 6, and the last equality follows from Eq. (18) and Lemma 7. From Corollary 10, $\nu(\mathbf{E} \circ \widehat{\mathbf{R}}_{2n,2n}, \overline{a_q} \vee \overline{b_q}) \leq \mu(\mathbf{E}, \overline{a_q}) + \mu(\widehat{\mathbf{R}}_{2n,2n}, \overline{b_q})$ is obtained. Note that $\mu(\widehat{\mathbf{R}}_{2n,2n}, \overline{b_q})$ corresponds to the probability of a collision among q uniform random variables of length $n-1$ bits. Thus, $\mu(\mathbf{B}_{2n,2n}, \overline{b_q}) \leq \binom{q}{2} \frac{1}{2^{n-1}} < \frac{q^2}{2^n}$ holds, proving Eq. (12). Theorem 2 is proved using Eq. (12) and triangle inequality,

C Proof of Lemma 4

Let us abbreviate $\mathbf{N}_2[\mathbf{P}_n, \mathbf{R}_{n,n}]$ to $\mathbf{N}_2^*$. Recall that for any RF:$\mathcal{X}^2 \to \mathcal{X}^2$, (S_i, T_i) denotes the i-th input, where $S_i, T_i \in \mathcal{X}$. Similarly, (U_i, V_i) denotes the i-th output, where $U_i, V_i \in \mathcal{X}$. For example, if $\mathbf{N}_2[\mathbf{P}_n, \mathbf{R}_{n,n}]$ is considered, then $U_i = \mathbf{P}_n(S_i)$ and $V_i = \mathbf{R}_{n,n}(\mathbf{P}_n(S_i)) \oplus T_i$. We observe that a_i (i.e., $\mathrm{dist}(S^i)$) and b_i (i.e., $\mathrm{dist}(U^i)$) are equivalent events if $\mathbf{N}_2^*$ is considered, since $U_i = \mathbf{P}_n(S_i)$. Therefore, we have to prove that $\mathbf{N}_2^{*\mathcal{A}\wedge\mathcal{C}} \equiv \mathbf{P}_{2n}^{\mathcal{A}\wedge\mathcal{B}}$ holds for some MES $\mathcal{C}$. We first prove $\mathbf{N}_2^*|\mathcal{A} \equiv \mathbf{P}_{2n}|\mathcal{A} \wedge \mathcal{B}$. To prove this, we have to verify that

$$P^{\mathbf{N}_2^*}_{U_iV_i|U^{i-1}V^{i-1}S^iT^ia_i} = P^{\mathbf{P}_{2n}}_{U_iV_i|U^{i-1}V^{i-1}S^iT^ia_ib_i} \tag{20}$$

holds, where Eq. (20) means the both sides are equal as functions of (s^i, t^i, u^i, v^i) (see Def. 6 for example). Note that both sides of Eq. (20) are defined for all $S^i = s^i$ and $U^{i-1} = u^{i-1}$ such that $\mathrm{dist}(s^i)$ and $\mathrm{dist}(u^{i-1})$ hold. If u_i collides with u_j for some $1 \le j \le i-1$, then b_i does not hold. Therefore, both sides are 0 for such u_i (note that a_i is equivalent to b_i for $\mathbf{N}_2^*$). Otherwise b_i holds and so, the lhs of Eq. (20) is $1/((2^n - i + 1) \cdot 2^n)$ since U_i is uniformly distributed on $\mathcal{X} \setminus \{u_1, \ldots, u_{i-1}\}$ and V_i is uniformly random on $\mathcal{X}$. Therefore, we have to verify if the rhs of Eq. (20) is $1/((2^n - i + 1) \cdot 2^n)$ in this case. Using simple decomposition, we have

$$P^{\mathbf{P}_{2n}}_{U_iV_i|U^{i-1}V^{i-1}S^iT^ia_ib_i} = \frac{P^{\mathbf{P}_{2n}}_{U^iV^i|S^iT^ia_ib_i}}{P^{\mathbf{P}_{2n}}_{U^{i-1}V^{i-1}|S^{i-1}T^{i-1}a_{i-1}b_{i-1}}}. \tag{21}$$

The numerator of the rhs of Eq. (21) is a uniform distribution on a set of all (u^i, v^i) that satisfies $\mathrm{dist}(u^i)$. The number of such (u^i, v^i) is $(2^n \cdot (2^n - 1) \cdot \cdots \cdot (2^n - i + 1)) \cdot (2^n)^i$. Similarly, the denominator is a uniform distribution on the set of size $(2^n \cdot (2^n - 1) \cdot \cdots \cdot (2^n - i + 2)) \cdot (2^n)^{i-1}$. Thus the rhs of Eq. (21) equals

$$\left(\frac{(2^n \cdot (2^n - 1) \cdot \cdots \cdot (2^n - i + 1)) \cdot (2^n)^i}{(2^n \cdot (2^n - 1) \cdot \cdots \cdot (2^n - i + 2) \cdot (2^n)^{i-1}} \right)^{-1} = \frac{1}{(2^n - (i-1)) \cdot 2^n}. \tag{22}$$

Therefore, Eq. (20) holds true and hence we have $\mathbf{N}_2^*|\mathcal{A} \equiv \mathbf{P}_{2n}|\mathcal{A} \wedge \mathcal{B}$. To apply Lemma 6, we need to check if

$$P^{\mathbf{P}_{2n}}_{a_ib_i|U^{i-1}V^{i-1}S^iT^ia_{i-1}b_{i-1}} \le P^{\mathbf{N}_2^*}_{a_i|U^{i-1}V^{i-1}S^iT^ia_{i-1}} \tag{23}$$

holds true for all possible arguments $(u^{i-1}, v^{i-1}, s^i, t^i)$. When s^i does not satisfy a_i, clearly Eq. (23) holds, since both sides are 0. When s^i satisfies a_i, the rhs of Eq. (23) is 1. Thus, Eq. (23) is proved. Combining Eq. (23) and Lemma 6, we have $\mathbf{N}_2^{*\mathcal{A}\wedge\mathcal{C}} \equiv \mathbf{P}_{2n}^{\mathcal{A}\wedge\mathcal{B}}$ for some MES $\mathcal{C}$ defined for $\mathbf{N}_2^*$. Therefore, $\mathbf{N}_2^{*\mathcal{A}\wedge\mathcal{B}\wedge\mathcal{C}} \equiv \mathbf{P}_{2n}^{\mathcal{A}\wedge\mathcal{B}}$ is proved.

D Proof of Theorem 5

Let $\mathbf{Q}^*$ be $\mathbf{E}_1 \circ \mathbf{P}_{2n} \circ \mathbf{E}_2^{-1}$ and let $\mathbf{Q}$ be $\mathbf{E}_1 \circ \mathbf{N}_2[\mathbf{P}_n, \mathbf{R}_{n,n}] \circ \mathbf{E}_2^{-1}$. From Lemmas 4 and 5,

$$\mathbf{Q}^{\mathcal{A}\wedge\mathcal{B}\wedge\mathcal{C}} \equiv \mathbf{Q}^{*\mathcal{A}\wedge\mathcal{B}} \tag{24}$$

holds, where $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ are the MESs appearing in Lemma 4. From Eq. (24) and Lemma 9, we obtain $\langle \mathbf{Q} \rangle^{\mathcal{A}\wedge\mathcal{B}\wedge\mathcal{C}} \equiv \langle \mathbf{Q}^* \rangle^{\mathcal{A}\wedge\mathcal{B}}$. This conditional equivalence and Theorem 6 indicate $\mathrm{Adv}^{\mathrm{cca}}_{\mathbf{Q},\mathbf{Q}^*}(q,\infty) \le \nu(\langle \mathbf{Q}^* \rangle, \overline{a_q} \vee \overline{b_q})$, which corresponds to the maximal probability of $\overline{a_q}$ (i.e., a collision in the left halves of inputs to $\mathbf{P}_{2n}$) or $\overline{b_q}$ (i.e., a collision in the left halves of $\mathbf{P}_{2n}$'s outputs) for all (q,∞)-CCAs . Therefore, we have

$$\mathrm{Adv}^{\mathrm{sprp}}_{\mathbf{Q}}(q,\infty) = \mathrm{Adv}^{\mathrm{cca}}_{\mathbf{Q},\mathbf{Q}^*}(q,\infty) \le \nu(\langle \mathbf{Q}^* \rangle, \overline{a_q} \vee \overline{b_q}) \le \nu(\langle \mathbf{Q}^* \rangle, \overline{a_q}) + \nu(\langle \mathbf{Q}^* \rangle, \overline{b_q}). \tag{25}$$

The first equality holds since $\mathbf{Q}^* \equiv \mathbf{P}_{2n}$, and the last inequality follows from Lemma 8.

We next analyze $\nu(\langle \mathbf{Q}^* \rangle, \overline{a_q})$. Let us use the following notations. The i-th input and output of $\mathbf{Q}^*$ are X_i and Y_i, respectively. In addition, let $\hat{X}_i$ denote (S_i, T_i) and $\hat{Y}_i$ denote (U_i, V_i). Note that $\hat{X}_i$ and $\hat{Y}_i$ correspond to the i-th input and output of $\mathbf{P}_{2n}$ in $\mathbf{Q}^*$. Observe that

$$\begin{aligned} \nu(\langle \mathbf{Q}^* \rangle, \overline{a_q}) &= \max_{\mathbf{D}:(q,\infty)\text{-CCA}} \sum_{x^q, y^q} P^{\mathbf{Q}^*}_{\overline{a_q}|X^qY^q}(x^q, y^q) \cdot P^{\mathbf{D}\diamond\langle \mathbf{Q}^* \rangle}_{X^qY^q}(x^q, y^q) \\ &\le \max_{x^q, y^q} P^{\mathbf{Q}^*}_{\overline{a_q}|X^qY^q}(x^q, y^q) \end{aligned} \tag{26}$$

holds, where $\mathbf{D} \diamond \langle \mathbf{Q}^* \rangle$ denotes an environment where $\mathbf{D}$ attacks $\mathbf{Q}^*$ by means of CCA, and the second maximum is taken over all x^q and y^q satisfying $\mathrm{dist}(x^q)$ and $\mathrm{dist}(y^q)$. Let $\beta_q \subset (\mathcal{X}^2)^q$ be the set of $\hat{x}^q = (s^q, t^q)$ such that a_q (i.e., $\mathrm{dist}(s^q)$) does not hold but $\mathrm{dist}(\hat{x}^q)$ holds. Now we have

$$P^{\mathbf{Q}^*}_{\overline{a_q}|X^qY^q}(x^q, y^q) = \frac{\sum_{\hat{x}^q \in \beta_q} P^{\mathbf{E}_1}_{\hat{X}^q|X^q}(\hat{x}^q, x^q) \cdot P^{\mathbf{Q}^*}_{Y^q|\hat{X}^qX^q}(y^q, \hat{x}^q, x^q)}{(\prod_{i=0}^{q-1} 2^{2n} - i)^{-1}}. \tag{27}$$

It is not difficult to verify that $P^{\mathbf{Q}^*}_{Y^q|\hat{X}^qX^q}(y^q, \hat{x}^q, x^q)$ equals to

$$\sum_{\hat{y}^q \in (\mathcal{X}^2)^q} P^{\mathbf{P}_{2n}}_{\hat{Y}^q|\hat{X}^qX^q}(\hat{y}^q, \hat{x}^q, x^q) \cdot P^{\mathbf{E}_2^{-1}}_{Y^q|\hat{Y}^q\hat{X}^qX^q}(y^q, \hat{x}^q, x^q) = \frac{1}{\prod_{i=0}^{q-1} 2^{2n} - i}. \tag{28}$$

The last equality results from the fact that $\mathbf{P}_{2n}$ is a URP and $\mathbf{E}_2$ is invertible. From Eqs. (27) and (28), we have $P^{\mathbf{Q}^*}_{\overline{a_q}|X^qY^q}(x^q, y^q) = \sum_{\hat{x}^q \in \beta_q} P^{\mathbf{E}_1}_{\hat{X}^q|X^q}(\hat{x}^q, x^q)$, which is at most $\mu(\mathbf{E}_1, \overline{a_q})$ (see Def. 8). Similarly, $\nu(\langle \mathbf{Q} \rangle, \overline{b_q})$ is no more than $\mu(\mathbf{E}_2, \overline{b_q})$. Thus, the first claim is proved. Note that the above proof is valid even if $\mathbf{E}_1$ and $\mathbf{E}_2$ are dependent. Combining this observation and the fact that one Feistel round, $\mathbf{E}$, is an involution (i.e., $\mathbf{E}^{-1} \equiv \mathbf{E}$), the second claim is proved.

E Figure of Hybrid Large Block PRP and SPRP

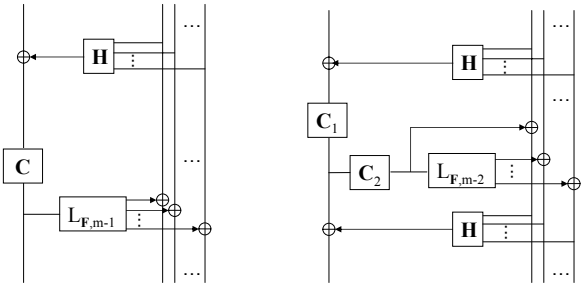


Fig. 2. Hybrid large block PRP (left) and SPRP (right)

Cryptanalysis of Sfinks*

Nicolas T. Courtois

Axalto Smart Cards Crypto Research, 36-38 rue de la Princesse,
BP 45, F-78430 Louveciennes Cedex, France
courtois@minrank.org

Abstract. Sfinks is an LFSR-based stream cipher submitted to ECRYPT call for stream ciphers by Braeken, Lano, Preneel *et al.* The designers of Sfinks do not include any real protection against algebraic attacks other than the so called "Algebraic Immunity", that relates to the complexity of a simple algebraic attack, and ignores more elaborate attacks. As a result, Sfinks is insecure.

Keywords: algebraic cryptanalysis, stream ciphers, nonlinear filters, Boolean functions, solving systems of multivariate equations, fast algebraic attacks on stream ciphers.

1 Introduction

Sfinks is a new stream cipher that has been submitted in April 2005 to ECRYPT call for stream cipher proposals, by Braeken, Lano, Mentens, Preneel and Varbauwhede [6]. It is a hardware-oriented stream cipher with associated authentication method (Profile 2A in ECRYPT project).

Sfinks is a very simple and elegant stream cipher, built following a very classical formula: a single maximum-period LFSR filtered by a Boolean function. Several large families of ciphers of this type (and even much more complex ones) have been in the recent years, quite badly broken by algebraic attacks, see for exemple [12, 13, 1, 14, 15, 2, 18]. Nevertheless the specialists of these ciphers counter-attacked by defining and applying the concept of Algebraic Immunity [7] to claim that some designs are "secure". Unfortunately, as we will see later, the notion of Algebraic Immunity protects against only one simple algebraic attack and ignores other algebraic attacks. More realistic (but also more complex to apply) security criteria for stream ciphers have been proposed in [13, 16].

We note that it is possible to design stream ciphers that would be in some sense "protected" against algebraic attacks (and also in a similar way against other known attacks, such as correlation and fast correlation attacks). It is even a common practice to add to the stream cipher some components that would make all these attacks less efficient (cf. [26, 13]), or even clearly impractical (e.g. [8]) to apply. This can be done, for example, with irregular clocking and/or a

* This work was partially supported by the French Ministry of Research RNRT X-CRYPT project and by the European Commission via ECRYPT network of excellence IST-2002-507932.

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 261–269, 2006.
© Springer-Verlag Berlin Heidelberg 2006

final compression component that would combine several consecutive outputs of the Boolean function in a complex way. The drawback of this is (possibly) some loss in speed and an important loss in hardware footprint (for example, irregular clocking would require a large buffer). It would make the design less elegant, and also much more vulnerable to timing and side-channel attacks.

Finally, the designers of Sfinks choose to stick to this simple and elegant design that is easy to study (and for which we would like to thank the authors as it helps the cryptanalysts too !). We have a simple filtered LFSR on which most of the known attacks can be applied directly and the only thing that prevents these attacks so far, is the parameters of Sfinks [6] were chosen so that the attack complexity is close to 2^{80}, without even any margin for (frequent) algorithmic improvements. Consequently, it is possible to say that Sfinks has been designed with no "protection" whatsoever against known attacks.

2 Short Description of Sfinks

A regularly clocked 256-bit binary LFSR provides, at each clock, 17 out of 256 its state bits, that are supplied to a Boolean function. The keystream is composed of successive output bits of this Boolean function.
The LFSR used in Sfinks is described by the following recursion formula:

$$s_{t+256} = s_{t+212} \oplus s_{t+194} \oplus s_{t+192} \oplus s_{t+187} \oplus s_{t+163} \oplus s_{t+151} \oplus s_{t+125} \oplus s_{t+115} \oplus s_{t+107} \oplus s_{t+85} \oplus s_{t+66} \oplus s_{t+64} \oplus s_{t+52} \oplus s_{t+48} \oplus s_{t+14} \oplus s_t$$

The Boolean Function Used in Sfinks
We call $f(x_t^{16}, \ldots x_t^0)$ the output filtering function of Sfinks [6]. The 17 variables used are selected as follows among the state bits of the LFSR:

$$\left(x_t^{16}, \ldots, x_t^0\right) \stackrel{def}{=} (s_{t+255}, s_{t+244}, s_{t+227}, s_{t+193}, s_{t+161}, s_{t+134}, s_{t+105}, s_{t+98}, s_{t+74}, s_{t+58}, s_{t+44}, s_{t+21}, s_{t+19}, s_{t+9}, s_{t+6}, s_{t+1}, s_t).$$

We define as P the corresponding projection mapping $GF(2^{256}) \rightarrow GF(2^{17})$. P is a multivariate linear transformation.
The function f is a highly non-linear Boolean function of degree 15, with 17 variables. It is defined as follows:

$$z_t = f(\bar{x}_t) = f(x_t^{16}, \ldots, x_t^0) = (INV(x_t^{16}, \ldots, x_t^1)\&1) \oplus x_t^0.$$

with INV being the inverse in $GF(2^{16})$ defined as follows. Let $GF(2^{16})$ be defined as $GF(2)[Z]/Z^{16} + Z^5 + Z^3 + Z^2 + 1$. We define INV as inverse in $GF(2^{16})$ complemented by $0 \mapsto 0$ as in Rijndael, implemented as a table operating on 16-bit words, in such a way that the least significative bit of the input is x_t^1, and it corresponds to the coefficient of Z^1 in polynomial arithmetic modulo $Z^{16} + Z^5 + Z^3 + Z^2 + 1$. At the output, f is defined by the least significative bit of the output word of INV (which again corresponds to Z^1).

3 Algebraic Attacks on Sfinks

Algebraic attacks on stream ciphers are based on the following observation. Let $L : GF(2^n) \rightarrow : GF(2^n)$ be a multivariate linear transformation that corresponds to clocking the LFSR. For each i, L^i is a known multivariate linear transformation. At any moment t in the cipher history, all bits of the internal state are **known** linear combinations of the bits of the initial state $s_0, \ldots, s_n$ (for Sfinks $n = 256$).

Let $z_t, t = 0, 1, 2, \ldots$ be the keystream generated by Sfinks and let f be its output Boolean function. We recall that $f \circ P$ is the version of f defined from $GF(2^n) \rightarrow GF(2)$ and takes all n bits as inputs, 17 of which are used and the other are ignored. Then we can write the problem of key recovery in Sfinks as follows.

$$\begin{cases} z_0 = f(\ P \quad\ (s_0, \ldots, s_{n-1})\ \ ) \\ z_1 = f(\ P\,(L\ \ (s_0, \ldots, s_{n-1}))\ ) \\ \qquad \vdots \\ z_t = f(\ P\,(L^t\ (s_0, \ldots, s_{n-1}))\ ) \\ \qquad \vdots \end{cases} \qquad (\#)$$

Algebraic attacks on stream cipher work by solving, (by more or less sophisticated methods) the above system equations (or a part of it). They use extensively the fact that the degree of these and other derived algebraic equations is preserved by the linear operation $P \circ L^t$, at any moment t.

In this paper we use the terminology of [13] to classify algebraic attacks on stream ciphers as S1, S2, S3, S4, S5 and S6. In addition we will give a complete description of all proposed attacks on Sfinks.

3.1 First Basic Algebraic Attacks on Sfinks (S1 and S2)

A Simplistic Algebraic Attack. The simplest attack scenario we can think of is known as direct linearization attack or S1, see [12, 13, 3]. It works as follows: the equations are of degree 15, and if we dispose of about $T = \binom{n}{15} + \binom{n}{14} + \ldots + \binom{n}{0} \approx 2^{79.2}$ keystream bits, than we can rewrite the system (#) as a system of T linear equations with T variables - all monomials are treated as new variables. The system is the solved with the complexity of T^ω, with ω being the exponent of the Gaussian reduction. In theory it is at most $\omega \leq 2.376$, see [9]. However the (neglected) constant factor in this algorithm is expected to be very big. Thus, in this paper we will systematically estimate the complexity of solving linear systems as about $T^{log_2 7}$ operations, which is believed to be achievable in practice with the Strassen's algorithm [29].

With this S1 attack we get a very large complexity: $T^{log_2 7} \approx 2^{222}$.

Probabilistic Variant. In scenario S2, introduced in [12], the Boolean function is approximated by a function of a lower degree to get a lower attack complexity. We do not develop tools to find good low-degree approximations of f. Nevertheless we believe that it is very unlikely that f used in Sfinks has very good

approximations that would lead to efficient algebraic attacks. The approximation to be interesting must hold with probability very close to 1, and in [12] such approximations existed because there were very few monomials of high degree. In Sfinks, f has many monomials of very high degree.

3.2 The S3 Algebraic Attack on Sfinks

In the scenario S3 introduced by Courtois and Meier in [13] and also studied by Carlet *et al* in [7], the degree of the equations (#) is substantially reduced.

This is possible due to the existence of a low-degree algebraic relation that relates input and output bits of f. For example, we assume that there exists an equation of the type:

$$zX^d + X^d.$$

This notation is very compact and convenient and is taken from [10]. It means that there is (at least one) equation of type

$$z \cdot g(x^{16}, \ldots, x^0) + h(x^{16}, \ldots, x^0) = 0 \qquad \text{with } \; z = f(x_t^{16}, \ldots, x_t^0).$$

with g and h being some multivariate polynomials of degree up to d. The equation has to be true with probability 1, i.e. for every choice of $(x^{16}, \ldots, x^0)$. In [7] it is shown[1] that equations of such type exist if and only if either f or $f+1$ have an annihilator of degree $\leq d$, i.e. $\exists g'$ of degree $\leq d$ s.t. $fg' = 0$ or $(f+1)g' = 0$.

Given the existence of one equation of type $zX^d + X^d$ for f (or of an annihilator of degree d), the cipher can be broken with complexity of $\binom{n}{d}^\omega$ as follows. Each equation of the system (#) as follows:

$$z_t = f(\ P\left(L^t\ (s_0, \ldots, s_{n-1})\right)\ )$$

is multiplied by the polynomial $g(\ P\left(L^t\ (s_0, \ldots, s_{n-1})\right)$ and since $fg = h$ it gives an equation of degree d:

$$z_t = h(\ P\left(L^t\ (s_0, \ldots, s_{n-1})\right)\ )$$

Then the system is solved by linearization exactly as described in Section 3.1.

For Sfinks, as we will se later (and as already remarked by the designers of Sfinks [6]) we can have $d = 6$ and the complexity of the attack is about 2^{108}. The keystream required in this attack is $T/4 \approx 2^{36.5}$ bits, as 4 linearly independent equations of type $zX^6 + X^6$ do exist.

[1] In fact this equivalent formulation of algebraic attacks were already introduced one year earlier in the appendix of the extended version of the original paper [13], under a different name of scenarios $S3_0$ and $S3_1$. There are many other equivalent formulations of the S3 attack, for example instead of annihilators we can talk about absorbing elements: g is the annihilator for f if and only if it is an absorbing element for $f+1$. We can also speak (as we do a lot in this paper) about algebraic I/O relations, see [11, 10, 16], that generalise the notion of Affine Multiples, known since 1992 [5, 27].

Theory vs. practice. It should be noted that in this attack some equations might become linearly dependent, however unlike in [11], this is not a problem at all in algebraic attacks on stream ciphers. This is because the attack method allows to produce as many new equations as may be necessary. In practice, it has been tested for LILI [26] by the authors of [13] in the extended version of this paper. The simulations show clearly that, the number of equations that are not linearly independent in this attack (that could be tolerated with little impact on the complexity) turns out to be really negligible. (See also [2].) Thus the complexity evaluations on algebraic attacks on stream ciphers are expected to be very tight and even rather conservative. It is in fact likely that applying the F5/2 Gröbner bases algorithm [17] to a subset of the resulting equations of degree d, combined with fast linear algebra, should allow to solve the systems with even lower complexity and/or with even less known keystream.

3.3 Fast Algebraic Attacks on Sfinks

Let $D = \binom{n}{d} + \binom{n}{d-1} + \ldots + 1$. Fast algebraic attacks on stream ciphers [14] are based on equations of type $zX^e + X^d$ with $e < d$. They instantiate the scenario S5 according to the terminology of [13]. The principle is as follows: we combine some D consecutive equations from (#) for some D consecutive positions t in such a way that the parts of degree d are eliminated. The equation obtained will be of degree "only" e and will be as follows:

$$\sum_{i=t}^{t+D} \alpha_{t+i} \cdot z_i \cdot g(\ P\left(L^i\ (s_0, \ldots, s_{n-1})\right)\ ) \qquad (*)$$

for some linear combination $(\alpha_0, \ldots, \alpha_{D-1}) \in GF(2)^D$. The same equation applies to each window of D consecutive steps and we will write (and use) it E times, for E overlapping intervals, with $E = \binom{n}{e} + \binom{n}{e-1} + \ldots + 1$. This is because we need to get the final system of degree e that is solvable by linearization (with complexity E^ω).

How to Compute α. The equation above $(*)$ will be true for a proper choice of α, that by definition is such that all the monomials of degree d are eliminated, which can be written as:

$$\forall_t \quad 0 = \sum_{i=t}^{t+D} \alpha_{t+i} \cdot h(\ P\left(L^i\ (s_0, \ldots, s_{n-1})\right)\ ) \qquad (**)$$

In other words, α is a linear combination of D consecutive bits, such that if in our cipher the output Boolean function were h, for any consecutive D steps applying α to bits output by the cipher would give always the sum equal to 0, (i.e. with h instead of f we would have $\sum \alpha_{t+i} z_{t+i} = 0$). Following this observation, in [14] Courtois proposed to use the Berlekamp-Massey algorithm to find this α. This idea has been validated by Armknecht in [2] but later it turned out that there is a much simpler, faster, and more powerful method. Hawkes and Rose,

inspired by the invited talk by Massey at FSE 2003 and by an old paper by Key [19], have shown in [18] that it is possible to compute in time of $D \log^2 D$ one single linear combination α that is "universal" for degree up to d, in the sense that it eliminates **any** Boolean function of degree $\leq d$ (and in particular for our function h), see [18]. This is the method we will adopt.

Substitution Step. In [14] it was claimed that the substitution step in the fast algebraic attacks should take about $E \cdot D$ steps. In [18] Hawkes and Rose explain that the simple substitution takes in fact DE^2 operations, and propose an improved FFT-based method that manages (after all) to do it in about $2ED \log D$ operations.

3.4 Summary - Complexity of Fast Algebraic Attacks

Following the work of Courtois, Hawkes and Rose [14, 18], and for any given $(e, d), e < d$ such that there is an equation of type $zX^e + X^d$ for f, we need to perform the following steps in order to perform an algebraic attack on any filter generator such as Sfinks:

0. **Relation Search Step.** Computing the equation[s] of type $zX^e + X^d$ for f. Since we have already handled this step (in few days on a PC) for all interesting cases, we will neglect the complexity of it in this paper (it is at most $\binom{17}{d}^\omega$ and can be improved).
1. **Pre-computation Step.** Given the characteristic polynomial of the LFSR, compute the ("universal") α for the degree d. This step requires according to [18] $D \log^2 D$ operations.
2. **Substitution Step.** Write the equations $(*)$ for E consecutive values of t, for example $t = 1, \ldots, E$. This step requires according to [18] about $2ED \log D$ operations.
3. **Solving Step.** Solve these equations by linearization. It requires E^ω operations.

Keystream complexity. The keystream required in the whole attack is $D + E - 1$ which is in general very close to D as if $e < d$ we have $E << D$. We note that in general the dimension of the space of I/O relations of type $zX^e + X^d$ for f is $A > 0$, but unlike in [13] it seems that when $A > 1$ it does not help to improve the attack. In S3 attack scenario [13] if the dimension of the space of I/O relations of type $zX^e + X^e$ for f is $A > 0$, then the keystream required in attack will be E/A. Here, the keystream complexity is dominated by D and there is little we can do to improve it. In fact, it can degrade the attack complexity, if we wanted to use several equations out of A, we need to repeat the substitution step A times, which in our best attack on Sfinks is the dominating step (in different cases and for other cryptosystems another step may be dominating, see Table 1 and [18]). Therefore for all our attacks we will use only one out of these A equations.

3.5 Our Results

In order to apply the fast algebraic attacks we need to search for suitable equations by computer simulations on the Boolean function of Sfinks. Finding such equations allows to execute the fast algebraic attack and there is very little theory[2] that would help to predict whether they do exist for a particular function. One has to check by running a series of simulations. Here are our results:

Table 1. Simulations on the dimension of the space of equations of type $zX^e + X^d$ for the Boolean function of Sfinks, i.e. the number of linearly independent functions g of degree $\leq e$ such that $h = fg$ is of degree $\leq d$. The resulting complexity of the fast algebraic attack (following [18]) is computed.

degree e of g	0	1	2	**2**	3	**3**	3	4	4	4	5	5
degree d of $h = fg$	15	14	7	**8**	6	**7**	8	5	6	7	5	6
dimension A of these g	1	0	0	**6**	0	**32**	136	0	4	204	0	4
out of which we used	–	–	–	**1**	–	**1**	1	–	1	1	–	1
complexity of the fast algebraic attack [14, 18]:												
pre-computation step	–	–	–	$\mathbf{2^{59.8}}$	–	$\mathbf{2^{54.5}}$	$2^{59.8}$	–	$2^{49.0}$	$2^{54.5}$	–	$2^{49.0}$
substitution step	–	–	–	$\mathbf{2^{69.7}}$	–	$\mathbf{2^{71}}$	2^{76}	–	$2^{71.6}$	$2^{76.9}$	–	$2^{77.3}$
solving step	–	–	–	$\mathbf{2^{42.1}}$	–	$\mathbf{2^{60.1}}$	$2^{60.1}$	–	$\mathbf{2^{76.9}}$	$2^{76.9}$	–	$2^{92.8}$
keystream required	–	–	–	$\mathbf{2^{48.6}}$	–	$\mathbf{2^{43.6}}$	$2^{48.5}$	–	$2^{38.5}$	$2^{43.6}$	–	$2^{38.5}$

The designers of Sfinks do mention fast algebraic attacks but did not analyse them in due details. Our fastest attack is in the column 4. We need 2^{70} computations and 2^{49} keystream bits. In column 6 we need 2^{71} computations and 2^{43} keystream bits. In both these cases, the substitution step dominates all the other steps of the attack.

We note that the attack in column 9 should also break Sfinks slightly faster than the claimed security level of 2^{80}. It requires only about $2^{38.5}$ keystream bits, (about 50 Giga-bytes), which can be seen as a realistic attack on Sfinks.

4 Summary and Conclusion

Sfinks is not equipped with neither a protection against, nor a sufficient security margin against algebraic attacks on stream ciphers [12, 13, 14, 18]. The result is an insecure cipher that can be broken with complexity of about 2^{71} computations and with 2^{43} keystream bits. We also present another attack slightly faster than 2^{80} that requires only 50 Giga-bytes of keystream which is realistic.

[2] There is a theory of worse-case attacks that show that some equations always exist for any component of a given size, see [15], however for a specific fixed Boolean function, better equations frequently do exist (e.g. for LILI-128, see [13]). This is maybe because as already suggested by the authors of [13] and [1], one should expect that there are trade-offs between (classical) non-linearity notions, and the resistance against (more recent) algebraic attacks.

Remark. In the design of block ciphers, Lars Knudsen (and others) have for a long time promoted the following: see how many rounds we need to make it secure, then double it (or even multiply by 4). We believe that stream ciphers should also be designed with such a comfortable security margin. Sfinks demonstrates that it is indeed very hard to design a fast and secure cipher with a very small hardware footprint. Finally, maybe stream ciphers cannot really be much faster than, e.g. AES in the counter mode.

Warning. It is hard to see if a cipher is already broken, or not, by already known algebraic attacks. The so called "Algebraic Immunity" of a Boolean function [7] does not solve the problem and it is unclear if it can give any guaranteed lower-bound on the complexity of algebraic attacks on stream ciphers (it basically only pertains to the simple attack scenario S3). Equations that lead to better attacks can be found at any moment. According to [14] it seems very hard to expect to explore systematically all the possibilities offered by the scenario S5 from [13]. The probabilistic versions S4 and S6 proposed in [13] are even more difficult to explore. Thus, even better algebraic attacks on Sfinks may remain uncovered.

References

1. Frederik Armknecht, Matthias Krause: *Algebraic Atacks on Combiners with Memory,* Crypto 2003, LNCS 2729, pp. 162-176, Springer.
2. Frederik Armknecht: *Improving Fast Algebraic Attacks,* FSE 2004, LNCS, Springer.
3. Steve Babbage: *Cryptanalysis of LILI-128,* Nessie project internal report, available at `https://www.cosic.esat.kuleuven.ac.be/nessie/reports/`, 22 January 2001.
4. Elad Barkan, Eli Biham, and Nathan Keller: *Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication,* In Crypto 2003, LNCS 2729, pp: 600-616, Springer, 2003.
5. I. Blake, X. Gao, R. Mullin, S. Vanstone and T. Yaghoobian, *Applications of Finite Fields,* Kluwer Academic Publishers, 1992.
6. An Braeken, Joseph Lano, Nele Mentens, Bart Preneel and Ingrid Verbauwhede: Sfinks specification and source code, April 2005, Available on ECRYPT Stream Cipher Project page, `http://www.ecrypt.eu.org/stream/sfinks.html`
7. Claude Carlet, Will Meier and Enes Pasalic: *Algebraic Attacks and Decomposition of Boolean Functions,* Eurocrypt 2004, pp. 474-491, LNCS 3027, Springer, 2004.
8. A. Clark, E. Dawson, J. Fuller, J. Golic, H-J. Lee, W. Millan, S-J. Moon, and L. Simpson: *The LILI-II Keystream Generator,* Presented at ACISP-2002, the 7th Australasian Conference on Information Security and Privacy. 3 - 5 July 2002. Deakin University, Melbourne, Australia.
9. Don Coppersmith, Shmuel Winograd: *Matrix multiplication via arithmetic progressions,* J. Symbolic Computation (1990), 9, pp. 251-280.
10. Nicolas Courtois: *The security of Hidden Field Equations (HFE)*, Cryptographers' Track Rsa Conference 2001, LNCS 2020, Springer, pp. 266-281.
11. Nicolas Courtois and Josef Pieprzyk: *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations,* Asiacrypt 2002, LNCS 2501, pp.267-287, Springer, A preprint with a different version of the attack is available at `http://eprint.iacr.org/2002/044/`.

12. Nicolas Courtois: *Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt,* ICISC 2002, November 2002, Seoul, Korea, LNCS 2587, pp. 182-199, Springer. An updated version is available at `http://eprint.iacr.org/2002/087/`.
13. Nicolas Courtois and Willi Meier: *Algebraic Attacks on Stream Ciphers with Linear Feedback,* Eurocrypt 2003, Warsaw, Poland, LNCS, Springer. A long, extended version of this paper is available from `www.nicolascourtois.net`.
14. Nicolas Courtois: *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback,* Crypto 2003, LNCS 2729, pp: 177-194, Springer.
15. Nicolas Courtois: *Algebraic Attacks on Combiners with Memory and Several Outputs,* ICISC 2004, LNCS, to appear in Springer in early 2005. Extended version available on `http://eprint.iacr.org/2003/125/`.
16. Nicolas Courtois: *The Inverse S-box, Non-linear Polynomial Relations and Cryptanalysis of Block Ciphers,* in AES 4 Conference, Bonn May 10-12 2004, LNCS 3373, pp. 170-188, Springer.
17. Jean-Charles Faugère: *A new efficient algorithm for computing Gröbner bases without reduction to zero (F5),* Workshop on Applications of Commutative Algebra, Catania, Italy, 3-6 April 2002, ACM Press.
18. Philip Hawkes, Gregory Rose: *Rewriting Variables: the Complexity of Fast Algebraic Attacks on Stream Ciphers,* in Crypto 2004, LNCS 3152, pp. 390-406, Springer, 2004. Available from `eprint.iacr.org/2004/081/`.
19. Edwin L. Key: *An Analysis of the Structure and Complexity of Nonlinear Binary Sequence Generators,* IEEE Transactions on Information Theory, Vol IT-22, No. 6, November 1976.
20. J. N. Massey and S. Serconek: *A Fourier Transform Approach to the Linear Complexity of Nonlinearly Filtered Sequences,* In Crypto '94, LNCS 839, Springer-Verlag, pp. 332–340, 1994.
21. Willi Meier and Othmar Staffelbach: *Fast correlation attacks on certain stream ciphers,* Journal of Cryptology, 1(3):159-176, 1989.
22. Willi Meier and Othmar Staffelbach: *Nonlinearity Criteria for Cryptographic Functions,* Eurocrypt'89, LNCS 434, Springer, pp.549-562, 1990.
23. Jacques Patarin: *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88,* In Crypto'95, Springer, LNCS 963, pp. 248-261, 1995.
24. Alex Biryukov, Adi Shamir: *Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers,* Asiacrypt 2000, LNCS 2248, Springer, pp. 1-13.
25. Adi Shamir, Jacques Patarin, Nicolas Courtois and Alexander Klimov: *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, Eurocrypt'2000, LNCS 1807, Springer, pp. 392-407.
26. L. Simpson, E. Dawson, J. Golic and W. Millan: *LILI Keystream Generator,* SAC'2000, LNCS 2012, Springer, pp. 248-261, Cf. `www.isrc.qut.edu.au/lili/`.
27. Jacques Patarin: *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of Asymmetric Algorithms*; Eurocrypt'96, Springer, pp. 33-48.
28. Claude Elwood Shannon: *Communication theory of secrecy systems,* Bell System Technical Journal 28 (1949), see in patricular page 704.
29. Volker Strassen: *Gaussian Elimination is Not Optimal,* Numerische Mathematik, vol 13, pp 354-356, 1969.

Weaknesses of COSvd (2,128) Stream Cipher

Bin Zhang[1,⋆], Hongjun Wu[2], Dengguo Feng[1,*], and Hong Wang[3,⋆⋆]

[1] State Key Laboratory of Information Security,
Institute of Software, the Chinese Academy of Sciences,
Beijing 100080, P.R. China
martin_zhangbin@yahoo.com.cn
[2] Katholieke Universiteit Leuven, Dept. ESAT/COSIC, Belgium
wu.hongjun@esat.kuleuven.be
[3] State Key Laboratory of Information Security,
Graduate School of the Chinese Academy of Sciences,
Beijing 100039, P.R. China

Abstract. The COSvd (2,128) cipher was proposed at the ECRYPT SASC'2004 workshop by Filiol et. al to strengthen the past COS (2,128) stream cipher. It uses clock-controlled non-linear feedback registers filtered by a highly non-linear output function and was claimed to prevent any existing attacks. However, as we will show in this paper, there are some serious security weaknesses in COSvd (2,128). The poorly designed S-box generates biased keystream and the message could be restored by a ciphertext-only attack in some broadcast applications . Besides, we launch a divide-and-conquer attack to recover the secret keys from $O(2^{26})$-byte known plaintext with high success rate and complexity $O(2^{113})$, which is much lower than 2^{512}, the complexity of exhaustive search.

Keywords: Stream cipher, COS cipher, Divide-and-Conquer, Non-linear feedback shift register.

1 Introduction

With the progress of the ECRYPT project, a lot of new stream ciphers emerge recently. The COSvd (2,128) stream cipher was proposed by Filiol et. al at the ECRYPT SASC'2004 workshop [3] and was claimed to resist against any existing attacks. It uses two non-linear feedback shift registers and a highly non-linear output function to generate keystream block-by-block. Compared to the original COS (2,128) stream cipher [2], there are two major improvements in the new design. First, at each step both the NLFSRs will update according to a clocking

⋆ Supported by the National Natural Science Foundation of China (Grant No. 60273027, 60373047) and the National Grand Fundamental Research 973 program of China(Grant No. 2004CB318004).

⋆⋆ Supported by the National Natural Science Foundation of China (Grant No. 60403005).

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 270–283, 2006.
© Springer-Verlag Berlin Heidelberg 2006

manner. Second, the 128-bit blocks generated via the crossing over mechanism from the NLFSRs will go through a highly non-linear filter (consisting of a chaotic layer and a S-box) to generate a 128-bit keystream block. Hence, the previous efficient attacks on the weaker version COS (2,128) no longer work [8, 9, 10]. It seems that the COSvd (2,128) stream cipher has been adopted for at least one commercial standard. The designers claimed that this new version can prevent any existing attacks.

However, as we will show below, there are still some serious flaws in COSvd (2,128) stream cipher. The poorly designed S-box generates a biased keystream, resulting in the retrieval of the plaintext messages in some broadcast applications. Moreover, there are strong correlations between the non-linear filter function and the output keystream, which facilitates a divide-and-conquer attack on the COSvd (2,128) stream cipher. Assume the chaotic layer uses double precision floating-point real numbers (which is an important case in practice), a divide-and-conquer attack is developed to restore the secret keys from $O(2^{26})$-byte keystream with success rate 93.4597% and complexity $O(2^{113})$, which is much lower than 2^{512}, the complexity of exhaustive search. Further, our attack can be easily modified to work for any precision implementation of the chaotic layer. The corresponding increase of complexity is very limited.

The rest of this paper is organized as follows. A description of the COSvd (2,128) stream cipher is given in Section 2. Section 3 deals with the security weaknesses in the design of the COSvd (2,128) stream cipher and presents a divide-and-conquer attack based on these flaws. Finally, some conclusions are given in Section 4.

2 Description of the COSvd (2,128) Stream Cipher

Since the key setup of the COSvd (2,128) has nothing to do with our analysis, we ignore it here and only focus on the keystream generation process. The COSvd (2,128) stream cipher uses two 128-bit non-linear feedback shift registers L_1 and L_2 as basic building blocks. The feedback functions of L_1 and L_2 are available from [3], they use bits $2, 5, 8, 15, 26, 38, 44, 47, 57$ of L_1 and L_2 as inputs. Let $L_1 = L_{10} \parallel L_{11} \parallel L_{12} \parallel L_{13}$ and $L_2 = L_{20} \parallel L_{21} \parallel L_{22} \parallel L_{23}$, where $\parallel$ denotes concatenation and L_{ij} is a 32-bit word for $i = 1, 2$, $j = 0, 1, 2, 3$. At ith step, the output keystream is generated as follows:

1. Compute clocking value d and d'.
   (a) Compute $clk = 2 * lsb(L_2) + lsb(L_1)$, where $lsb(\cdot)$ denotes the least significant bit.
   (b) $d = C[clk]$, where $C[0, \ldots, 3] = \{64, 65, 66, 64\}$.
   (c) Compute $clk = 2 * msb(L_1) + msb(L_2)$, where $msb(\cdot)$ denotes the most significant bit.
   (d) $d' = C'[clk]$, where $C'[0, \ldots, 3] = \{41, 43, 47, 51\}$.
2. If i is even, clock L_1 d times and L_2 d' times. If i is odd, clock L_1 d' times and L_2 d times.

3. Produce a 128-bit block B_i as $B_i = B_{i0} \parallel B_{i1} \parallel B_{i2} \parallel B_{i3}$, where $B_{i0} = L_{20} \oplus L_{12}$, $B_{i1} = L_{21} \oplus L_{13}$, $B_{i2} = L_{22} \oplus L_{10}$ and $B_{i3} = L_{23} \oplus L_{11}$, $\oplus$ is bitwise exclusive or.
4. Divide B_i into 16 bytes B_i^j, $0 \leq j < 16$, then pass B_i byte-by-byte to the following HNLL (Highly Non-Linear Layer) to generate the output keystream.

Highly Non-Linear Layer (HNLL). This module aims at hiding the internal states of the NLFSRs and forbids the previous attacks against the COS (2,128) cipher [8, 9, 10]. First, a 10-bit segment of a chaotic sequence is applied to B_i^j. Then feed the outcome to a S-box to generate one byte keystream. More precisely, The henon map [4, 5] is used to construct the chaotic sequence, i.e. for an initial point (x_{-2}, x_{-1}), generate a binary sequence z_i as follows. Let $x_n = 1 + 0.3x_{n-2} - 1.4x_{n-1}^2$, then $z_n = 1$ if $x_n > 0.39912$. Otherwise $z_n = 0$. Generate another binary sequence z_i' from (x'_{-2}, x'_{-1}) in the same way, Xor z_i and z_i' to get a binary sequence $h_0, h_1, \ldots$. For each $B_i^j = b_{i7}^j, b_{i6}^j, \ldots, b_{i0}^j$, truncate two chaotic segments $H_i^j = h_{10(16i+j)} \parallel h_{10(16i+j)+1} \parallel \ldots \parallel h_{10(16i+j)+7}$ and $K_i^j = h_{10(16i+j)+8} \parallel h_{10(16i+j)+9}$, generate one byte keystream as Tabkey2$[B_i^j \oplus H_i^j][K_i^j]$, where Tabkey2 is a $\mathbb{F}_2^8 \times \mathbb{F}_2^2$ to $\mathbb{F}_2^8$ S-box given in Appendix A.

The key of the COSvd (2,128) consists of the initial states of the two non-linear feedback shift registers and the initial values $(x_{-2}, x_{-1}), (x'_{-2}, x'_{-1})$. In [3], the designers do not specify the accuracy of the latter four real numbers. Although our attack works for any precision implementation of the chaotic layer, we focus on the case that these four real numbers are all in double precision floating-point form in our analysis, which is an important case in practical applications. Following the IEEE 754 floating-point standard [6], a double precision floating-point number is represented as a 64-bit word, thus the key space of the COSvd (2,128) stream cipher in our analysis is 2^{512}.

3 Our Analysis

In this section, we will show that there are some serious security flaws in COSvd (2,128) stream cipher. First, the output keystream of this cipher is biased due to the poorly designed S-box. It is easy to distinguish the keystream of COSvd (2,128) from a truly random binary sequence. In some broadcast applications where a message is encrypted many times with different segments of keystream, we can restore the message easily from the ciphertexts. Second, there are strong correlations between the keystream and the non-linear filter function. It is a typical case to launch a divide-and-conquer attack.

3.1 Some Weaknesses

According to the specification of the S-box Tabkey2 in Appendix A, the distribution of each integer in $[0, 256)$ can be computed easily. The results are listed in Table 1. We can see from Table 1 that though 238 elements appear 4 times as expected, 18 elements appear more or less than 4 times, which can be used

Table 1. Distribution of integers $[0, 256)$ in Tabkey2

Times	Number	Element
2	1	$\{0\}$
3	8	$\{2, 11, 50, 52, 65, 104, 160, 208\}$
5	8	$\{4, 5, 19, 32, 37, 75, 174, 151\}$
6	1	$\{17\}$
4	238	Others

to distinguish the keystream. For example, the byte 17 appears with probability $6/(256 \cdot 4) = 3/2^9$. In the keystream, byte 17 appears about 1.5 times more than it appears in a random binary sequence. We verified the above statement with experiments. We generated 2^{18}-byte keystream and found that the probability that 17 appears in it is $2^{-8} + 1.05 \cdot 2^{-9}$, which is very close to $3/2^9$.

Due to the large biases in the keystream, the plaintext message could be recovered in some broadcast applications when one message is encrypted many times with different segments of keystream [7]. Assume the message $m = m_1 \dots m_l$ (consisting of l bytes) is encrypted for N times. For each m_i, the corresponding ciphertext byte sequence is $m_i \oplus c_j$ $(1 \le j \le N)$. Byte $17 \oplus m_i$ appears in the ciphertext byte sequence with probability $p = 2^{-8} + 2^{-9}$ and is expected to be the most frequently appearing element. In N samples, byte 17 has binomial distribution (N, p) rather than $(N, p' = 2^{-8})$ in the random case. We approximate the binomial distribution with the normal distribution. Let $u = Np$, $\sigma = \sqrt{Np(1-p)}$ be mean and the standard deviation, respectively. Let u' and σ' be the corresponding parameters in random case. If $|u - u'| > 2(\sigma + \sigma')$, i.e. $N > 2^{16.31}$, the byte sequence $m_i \oplus c_j$ can be distinguished from random byte sequence $m'_i \oplus c_j$ with probability 0.9772. Hence, we can recover the message m byte-by-byte in this way. In our experiments, we encrypted the message by COSvd(2,128) for 2^{17} times and successfully recovered the message byte by xoring the most frequently appearing byte in the ciphertext with 17.

Except for the above flaws, there are still some other weaknesses in the S-box. For example, if byte 50 appears in the output keystream, then from the array in Appendix A (it is in C language-like notation), we know that there are three candidate positions generating 50, i.e. $(68, 0)$, $(108, 2)$ and $(115, 3)$. Transform the coordinates into binary form, we have:

$$\begin{aligned} 68 &\to 01000100, \quad 0 \to 00 \\ 108 &\to 01101100, \quad 2 \to 10 \\ 115 &\to 01110011, \quad 3 \to 11. \end{aligned} \tag{1}$$

Let our eyes go down each column, we can see From (1) that

$$\begin{aligned} b^j_{i7} \oplus h_{10(16i+j)+7} &= b^j_{i6} \oplus h_{10(16i+j)+6} \oplus 1 = 0 \\ b^j_{i0} \oplus h_{10(16i+j)} &= b^j_{i1} \oplus h_{10(16i+j)+1} = h_{10(16i+j)+9} \\ b^j_{i0} \oplus h_{10(16i+j)} &= b^j_{i4} \oplus h_{10(16i+j)+4} \end{aligned}$$

$$
\begin{aligned}
&b^j_{i2} \oplus h_{10(16i+j)+2} = b^j_{i4} \oplus h_{10(16i+j)+4} \oplus 1\\
&b^j_{i5} \oplus h_{10(16i+j)+5} = h_{10(16i+j)+8}\\
&b^j_{i4} \oplus h_{10(16i+j)+4} \oplus b^j_{i5} \oplus h_{10(16i+j)+5} = b^j_{i3} \oplus h_{10(16i+j)+3}\\
&b^j_{i2} \oplus h_{10(16i+j)+2} \oplus b^j_{i3} \oplus h_{10(16i+j)+3} = 1 \oplus b^j_{i5} \oplus h_{10(16i+j)+5}, \qquad (2)
\end{aligned}
$$

and

$$
\begin{aligned}
&P(h_{10(16i+j)+8} = 1) = 2/3\\
&P(h_{10(16i+j)+9} = 0) = 2/3. \qquad (3)
\end{aligned}
$$

From (2) and (3), we can see that there are strong correlations between the inputs and the outputs of the S-box. In fact, we tested all the bytes in $[0, 256)$ and our experimental results show that the linear equations in (2) exist for every byte (for the sake of limited space, we omit the results here), the only difference is that in some cases we get less equations than in the above example. Hence, from the keystream, we can deduce a lot of linear equations in terms of the bits in $B^j_i \oplus H^j_i$ and K^j_i. One may expect to launch a known plaintext attack based on these linear equations to break this cipher. However, the existence of the chaotic sequence makes such efforts fail.

3.2 A Divide-and-Conquer Attack

Instead, we develop a divide-and-conquer attack to remove the HNLL from the non-linear feedback shift registers first, after which the internal states of the NLFSRs could be easily recovered.

First note that although the single bit distribution of the henon map is balanced due to the careful choice of the threshold value 0.39912, the two-bit distribution of the henon map is heavily biased. By generating 2^{30} output bits with different starting points within the convergence quadrilateral Q defined by $(-1.33, 0.42)$, $(1.32, 0.133)$, $(1.245, -0.14)$ and $(-1.06, -0.5)$, we obtain that (z_i, z_{i+1}) would be $(0, 0)$, $(0, 1)$, $(1, 0)$ and $(1, 1)$ with probability $0.25 - 2^{-3.2811}$, $0.25 + 2^{-3.36745}$, $0.25 + 2^{-3.36745}$ and $0.25 - 2^{-3.45931}$, respectively. These large biases form the basis of our divide-and-conquer attack. Besides, the henon map has another security weakness under the double precision floating-point assumption. According to the IEEE 754 floating-point standard [6], a double precision floating-point real number is represented as a 64-bit word: 1 bit for sign, 11 bits for the exponent and a 52-bit mantissa. There is an implied high-order 1-bit which is always set to 1. Let $e = e_{10}e_9, \ldots, e_0$ be the 11-bit exponent and $d_1 d_2 \ldots d_{52}$ denote the 52-bit mantissa, where $d_i (1 \leq i \leq 52)$ are binary numbers. Then the absolute value of a double precision floating-point number is $(1 + \sum_{i=1}^{52} d_i \cdot 2^{-i}) \cdot 2^{e-1023}$. Here comes our observation, see Figure 1. Let $e = e_{10}e_9, \ldots, e_0$, if we only search $e_2e_1e_0$ and let $(e_{10}, e_9 \ldots, e_3) = (0, 1, 1, \ldots, 1)$, then the points that we do not cover by this method are those with the absolute value of at least one coordinate less than $2^{-7} + 2^{-7} \cdot (1 - 2^{-52}) < 2^{-6} = 0.015625$. Thus the probability that we guess

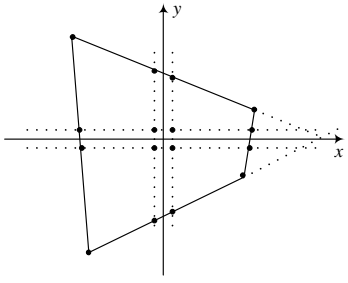


Fig. 1. Convergence quadrilateral Q and the guessed area

correctly is very high, i.e. the area covered by the internal points with the absolute value of either y-coordinate or x-coordinate less than 0.015625 is quite small compared to the whole area Q. We use Helen formula to compute the area of triangles involved in Figure 1. The proportion that the dashed area vs the whole area Q is $(0.00953758 + 0.0389093 - 0.000244141)/1.48139 = 3.25389\%$. Hence, if we exhaustively search the initial point (x_{-2}, x_{-1}) of one henon map sequence in this way, the complexity is only $O(2^{112})$, while with success rate 96.7461%.

Now we are ready to present our divide-and-conquer attack. The basic idea of our attack is as follows. If we guess the initial point (x_{-2}, x_{-1}) of one henon map sequence and deduce partial information of K_i^j from the keystream and z_i generated from (x_{-2}, x_{-1}) by the specified correlation, then we can determine the two-bit distribution of the xored sequence of K_i^j and z_i. If the resulted sequence behaves like a henon map sequence under the circumstance, we accept the guess. Otherwise, discard it.

More precisely, let $f : GF(2)^2 \to \mathbb{R}$ be a distribution for $x \in GF(2)^2$, it is well known that [1] the distribution of the xor of 2 i.i.d. (independent and identical distribution) random vectors with distribution f is $f^{\otimes 2}$, where $f^{\otimes 2} = (f \otimes f)(x) = \sum_{y \in GF(2)^2} f(y) \cdot f(x \oplus y)$. For a henon map sequence z_i, its two-bit distribution is $f(0) = 1/4 - 2^{-3.2811}$, $f(1) = 1/4 + 2^{-3.36745}$, $f(2) = 1/4 + 2^{-3.36745}$, $f(3) = 1/4 - 2^{-3.45931}$ (here we use quaternary representation). Thus the two-bit distribution of the xor of two henon map sequences is:

$$f^{\otimes 2}(0) = \sum_{i=0}^{3} f(i) \cdot f(i) = 0.287625, \qquad f^{\otimes 2}(1) = \sum_{i=0}^{3} f(i \oplus 1) \cdot f(i) = 0.212447$$

$$f^{\otimes 2}(2) = \sum_{i=0}^{3} f(i \oplus 2) \cdot f(i) = 0.212447, \quad f^{\otimes 2}(3) = \sum_{i=0}^{3} f(i \oplus 3) \cdot f(i) = 0.287482.$$

Now the mountain in front of us is that from the output keystream we can not uniquely determine the value of K_i^j, which seems to frustrate the consequent procedures. However, due to the poorly designed S-box Tabkey2, this difficulty can be overcome by the following method. Note that byte 0 appears in the keystream only when $K_i^j = 01$ or 10, which implies that if we guess (x_{-2}, x_{-1}),

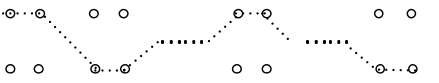


Fig. 2. Guess process

generate z_i and xor the corresponding $z_{10(16i+j)+8} \parallel z_{10(16i+j)+9}$ whenever the output byte is 0 with 01 and 10 respectively, we actually xor $z_{10(16i+j)+8} \parallel z_{10(16i+j)+9}$ by K_i^j and $K_i^j \oplus 11$ respectively, since $01 \oplus 10 = 11$ is commutative. Figure 2 shows the process. In Figure 2, the dashed route represents the actual value of K_i^j. Then we can count the two-bit probability of the resulted sequence to see if the guess is correct.

If the byte 0 in the keystream corresponds to $K_i^j = 10$, the two-bit distribution of the corresponding sequence z_i' is as follows.

$$
\begin{aligned}
&P(z'_{10(16i+j)+8} \parallel z'_{10(16i+j)+9} = 00 \mid K_i^j = 10) \\
&= \frac{P(z'_{10(16i+j)+8} \parallel z'_{10(16i+j)+9} = 00, z_{10(16i+j)+8} \parallel z_{10(16i+j)+9} = 10)}{P(K_i^j = 10)} \\
&= \frac{(1/4 - 2^{-3.2811})(1/4 + 2^{-3.36745})}{0.212447} = 0.24024 \\
&P(z'_{10(16i+j)+8} \parallel z'_{10(16i+j)+9} = 01 \mid K_i^j = 10) \\
&= \frac{P(z'_{10(16i+j)+8} \parallel z'_{10(16i+j)+9} = 01, z_{10(16i+j)+8} \parallel z_{10(16i+j)+9} = 11)}{P(K_i^j = 10)} \\
&= \frac{(1/4 - 2^{-3.45931})(1/4 + 2^{-3.36745})}{0.212447} = 0.259759 \\
&P(z'_{10(16i+j)+8} \parallel z'_{10(16i+j)+9} = 10 \mid K_i^j = 10) = 0.24024 \\
&P(z'_{10(16i+j)+8} \parallel z'_{10(16i+j)+9} = 11 \mid K_i^j = 10) = 0.259759.
\end{aligned}
$$

If the byte 0 in the keystream corresponds to $K_i^j = 01$, the two-bit distribution of the corresponding z_i' can be computed similarly and we just list the results here.

$$
\begin{aligned}
&P(z'_{10(16i+j)+8} \parallel z'_{10(16i+j)+9} = 00 \mid K_i^j = 01) = 0.24024 \\
&P(z'_{10(16i+j)+8} \parallel z'_{10(16i+j)+9} = 01 \mid K_i^j = 01) = 0.24024 \\
&P(z'_{10(16i+j)+8} \parallel z'_{10(16i+j)+9} = 10 \mid K_i^j = 01) = 0.259759 \\
&P(z'_{10(16i+j)+8} \parallel z'_{10(16i+j)+9} = 11 \mid K_i^j = 01) = 0.259759.
\end{aligned}
$$

Hence, if (x_{-2}, x_{-1}) is correctly guessed, then each two-bit pattern appears in the resulted sequence with probability:

$$
\begin{aligned}
&(0.24024 \cdot \frac{l}{2} + 0.24024 \cdot \frac{l}{2} + 0.259759 \cdot \frac{l}{2} + 0.259759 \cdot \frac{l}{2})/2l \\
&= (0.24024 + 0.259759)/2 = 0.25, \qquad (4)
\end{aligned}
$$

i.e. uniform distribution, where l is the number of 0 byte in the keystream. If the initial point is wrongly guessed and sequence z_i'' is generated, then

$$\begin{aligned}&(z''_{10(16i+j)+8}, z''_{10(16i+j)+9}) \oplus (z_{10(16i+j)+8}, z_{10(16i+j)+9})\\ &\oplus(z'_{10(16i+j)+8}, z'_{10(16i+j)+9})\\ &= (z''_{10(16i+j)+8}, z''_{10(16i+j)+9}) \oplus (10) \quad \text{or}\\ &= (z''_{10(16i+j)+8}, z''_{10(16i+j)+9}) \oplus (01),\end{aligned}$$

i.e. pair (00) of z'' maps to (10), (01) to (11), (10) to (00) and (11) to (01) if $K_i^j = 10$; pair (00) maps to 01, (01) to (00), (10) to (11) and (11) to (10) if $K_i^j = 01$. Hence, the two-bit distribution of the resulted sequence is:

$$\begin{aligned}&00:\\ &\frac{\frac{l}{2}(\frac{1}{4}+2^{-3.36745})+\frac{l}{2}(\frac{1}{4}+2^{-3.36745})+\frac{l}{2}(\frac{1}{4}+2^{-3.36745})+\frac{l}{2}(\frac{1}{4}+2^{-3.36745})}{2l}\\ &=\frac{1}{4}+2^{-3.36745}\\ &01:\\ &\frac{\frac{l}{2}(\frac{1}{4}-2^{-3.45931})+\frac{l}{2}(\frac{1}{4}-2^{-3.2811})+\frac{l}{2}(\frac{1}{4}-2^{-3.2811})+\frac{l}{2}(\frac{1}{4}-2^{-3.45931})}{2l}\\ &=\frac{1}{4}-2^{-3.36746}\\ &10:\ \frac{1}{4}-2^{-3.36746}, \qquad 11:\frac{1}{4}+2^{-3.36745}.\end{aligned} \tag{5}$$

It is obvious that the distribution in (5) is different from that in (4), which can be used to filter out the wrong guesses. We made experiments to verify the above analysis. We generated 2^{30}-bit chaotic sequence of $z_i \oplus z'_i$. If (x_{-2}, x_{-1}) is correctly guessed, the distribution of 00 in $(z_i, z_{i+1}) \oplus 01$ and $(z_i, z_{i+1}) \oplus 10$ whenever $(z_i, z_{i+1}) \oplus (z'_i, z'_{i+1}) = 01$ or 10 is 0.25016676 as expected. Otherwise, this value is $1/4 + 2^{-3.36799}$, which is very close to the theoretical result.

Assume $(2^8 \cdot n)$-byte keystream are available, then $n/2$ non-consecutive 0-byte keystream are known to a cryptanalyst. Randomly guess the initial point (x_{-2}, x_{-1}) and generate sequence z_i. For each byte 0 in the keystream, xor the corresponding pair $(z_{10(16i+j)+8}, z_{10(16i+j)+9})$ with (01) and (10), respectively. Let $G = \{(z_{10(16i+j)+8}, z_{10(16i+j)+9}) \oplus (01), (z_{10(16i+j)+8}, z_{10(16i+j)+9}) \oplus (10)\}_{i,j}$ be the resulting sequence. Let a_0, a_1, a_2 and a_3 be the number of (00), (01), (10) and (11) in the resulting sequence G, respectively. If (x_{-2}, x_{-1}) is correctly guessed, (00), (01), (10) and (11) would appear with uniform distribution in the resulting sequence. Let

$$\begin{aligned}A = \ &\{(a_0, a_1, a_2, a_3) \mid n \cdot (\frac{1}{4} - x) \le a_0, a_1, a_2, a_3 \le n \cdot (\frac{1}{4} + x)\\ &\text{and } a_0 + a_1 + a_2 + a_3 = n\},\end{aligned} \tag{6}$$

where x is a parameter to be determined later so that if the correct point is guessed, $(a_0, a_1, a_2, a_3) \in A$ with the following probability P_1 very close to 1.

$$P_1 = \sum_{(a_0,a_1,a_2,a_3)\in A} \frac{n!}{a_0!a_1!a_2!a_3!} \cdot (\frac{1}{4})^n$$

$$P_2 = \sum_{(a_0,a_1,a_2,a_3)\in A} \frac{n!}{a_0!a_1!a_2!a_3!} \cdot (\frac{1}{4} + 2^{-3.36745})^{a_0+a_3} \cdot (\frac{1}{4} - 2^{-3.36746})^{a_1+a_2},$$

where P_2 is the probability that a wrong guess will result in $(a_0, a_1, a_2, a_3) \in A$. Instead of computing P_1 and P_2 directly, we approximate P_1 and P_2 with the multivariate normal distribution:

$$P_1 \rightharpoonup \int_{n(\frac{1}{4}-x)-0.5}^{n(\frac{1}{4}+x)+0.5} \int_{n(\frac{1}{4}-x)-0.5}^{n(\frac{1}{4}+x)+0.5} \int_{n(\frac{1}{4}-x)-0.5}^{n(\frac{1}{4}+x)+0.5} f(y_1, y_2, y_3)dy_1dy_2dy_3 \quad (7)$$

$$P_2 \rightharpoonup \int_{n(\frac{1}{4}-x)-0.5}^{n(\frac{1}{4}+x)+0.5} \int_{n(\frac{1}{4}-x)-0.5}^{n(\frac{1}{4}+x)+0.5} \int_{n(\frac{1}{4}-x)-0.5}^{n(\frac{1}{4}+x)+0.5} g(y_1, y_2, y_3)dy_1dy_2dy_3, \quad (8)$$

where $f(y_1, y_2, y_3) = \frac{1}{\sqrt{(2\pi)^3(|B|)}} e^{-(\mathbf{y}-\mathbf{u})^T B^{-1}(\mathbf{y}-\mathbf{u})/2}$, $\mathbf{y} = (y_1, y_2, y_3)^T$ and $\mathbf{u} = (n/4, n/4, n/4)^T$, matrix B is given as follows:

$$\begin{pmatrix} \frac{3}{16}n & -\frac{1}{16}n & -\frac{1}{16}n \\ -\frac{1}{16}n & \frac{3}{16}n & -\frac{1}{16}n \\ -\frac{1}{16}n & -\frac{1}{16}n & \frac{3}{16}n \end{pmatrix}.$$

Similarly, in (8), $g(y_1, y_2, y_3) = \frac{1}{\sqrt{(2\pi)^3(|B'|)}} e^{-(\mathbf{y}-\mathbf{u}')^T B'^{-1}(\mathbf{y}-\mathbf{u}')/2}$, $\mathbf{u}' = ((1/4 - 2^{-3.36746})n, (1/4 + 2^{-3.36745})n, (1/4 + 2^{-3.36745})n)^T$ and B' is:

$$\begin{pmatrix} 0.129665n & -0.0531118n & -0.0531118n \\ -0.0531118n & 0.226559n & -0.120335n \\ -0.0531118n & -0.120335n & 0.226559n \end{pmatrix}.$$

We use Mathematica to compute P_1 and P_2 for different values of n and x in our experiments. We found that when $n = 2^{12}$ and $x = 96/2^{12}$, $P_1 = 0.998521$ and $P_2 = 2^{-193.725} < 2^{-112}$. Since byte 0 appears in the keystream with probability 2^{-9}, we can restore (x_{-2}, x_{-1}) with success rate $0.967461 \cdot 0.998521 = 96.603\%$ and complexity $O(2^{112})$ from $O(2^{20})$-byte keystream.

So far, we have recovered the initial point (x_{-2}, x_{-1}) of one henon map sequence. The remaining problems are to get the initial point (x'_{-2}, x'_{-1}) of the other henon map sequence and the initial states of the NLFSRs. As can be seen below, it is much easier to fulfill the remaining tasks.

When targeting (x'_{-2}, x'_{-1}), it suffices to note that if we exhaustively search (x'_{-2}, x'_{-1}) and generate z'_i, then $h_{10(16i+j)+8} = 1 \oplus h_{10(16i+j)+9}$ hold with probability 1 whenever the corresponding output byte is 0. Therefore, given 2^{16}-byte keystream, the wrong guesses of (x'_{-2}, x'_{-1}) would pass the filter process with

probability 2^{-128}. The overall complexity till now is $O(2^{112} + 2^{112}) = O(2^{113})$ and the success rate is $0.967461^2 \cdot 0.998521 = 93.4597\%$.

After getting (x'_{-2}, x'_{-1}), we know exactly the values of K_i^j, which implies that we can remove the HNLL from the output of two NLFSRs in almost all cases (except for those 9 bytes appear ≥ 5 times in S-box: in each case only 1 or 2 K_i^j values cause little uncertainty because of large correlations specified above and this will cause the increase of complexity only by a constant factor), i.e. from the output keystream and K_i^j, we know the corresponding inputs $B_i^j \oplus H_i^j$ to the S-box. Since the two henon map sequences have been recovered already, we know exactly B_i^j. Consider any seven consecutive steps starting from an odd

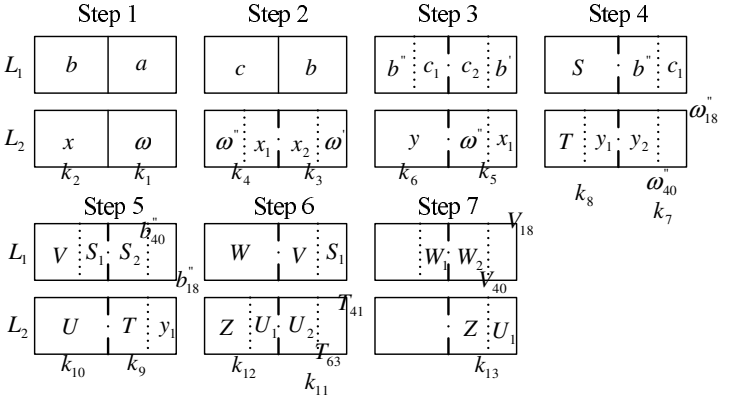


Fig. 3. 7 consecutive steps

step, as shown in Figure 3. L_1 is clocked d times at the second, fourth and sixth steps and d' times at the other steps. L_2 is the opposite. In Figure 3, a, b, c, x, ω, y, S, U, k_i $(1 \leq i \leq 13)$ and W are 64-bit words. Here we use the notations k_i $(1 \leq i \leq 13)$ to represent the outcomes of the two NLFSRs through the cross over mechanism. Without loss of generality, we assume in these seven steps, $d = 64$ and $d' = 41$. This case occurs with probability $(2^{-3})^6 = 2^{-18}$ and other cases can be treated similarly. Thus the lengths of strings ω', ω'', x_1, x_2, b', b'', c_1, c_2, y_1, y_2, T, V, S_1, S_2, Z, U_1, U_2, W_1 and W_2 can be determined accordingly. According to the clocking rule of COSvd(2,128) and the cross over mechanism, we have:

$$\begin{cases} b_{63} \oplus \omega_{63} = k_{1,63} \\ \quad \cdots \\ b_0 \oplus \omega_0 = k_{1,0} \\ a_{63} \oplus x_{63} = k_{2,63} \\ \quad \cdots \\ a_0 \oplus x_0 = k_{2,0} \\ b_{63} = 0 \\ x_{63} = 0 \\ a_0 \oplus \omega_0 = 0, \end{cases}$$

where $b = b_{63} \dots b_0$, $\omega = \omega_{63} \dots \omega_0$, $a = a_{63} \dots a_0$, $x = x_{63} \dots x_0$ and $k_i = k_{i,63} \dots k_{i,0}$. We have 256 unknown bits (involved in a, b, x and ω), while we already get 131 linear equations. This is the flaw of the cross over mechanism. If we forward to step 2, we get:

$$\begin{cases} c_{63} \oplus x_{40} = k_{3,63} \\ \quad \cdots \\ c_{23} \oplus x_0 = k_{3,23} \\ c_{22} \oplus \omega_{63} = k_{3,22} \\ \quad \cdots \\ c_0 \oplus \omega_{41} = k_{3,0} \\ b_{63} \oplus \omega''_{40} = k_{4,63} \\ \quad \cdots \\ b_{23} \oplus \omega''_0 = k_{4,23} \\ c_{63} = 0 \\ \omega''_{40} = 0 \\ b_0 \oplus \omega_{41} = 0, \end{cases}$$

especially,

$$\begin{cases} b_{22} \oplus x_{63} = k_{4,22} \\ \quad \cdots \\ b_0 \oplus x_{41} = k_{4,0}, \end{cases}$$

i.e. we get 23 more linear equations on a, b, x and ω. Besides, the new added bits in c and ω'' can be represented as linear combinations of the keystream bits and the bits involved in b, x and ω. With the steps forwarding, the new added variables can all be represented as linear combinations of keystream bits and the bits involved in b, x and ω and we can get 23 more linear equations on the initial states every one more step. For example, in step 3, we get:

$$\begin{cases} c_{63} \oplus x_{63} = k_{3,63} \oplus x_{40} \oplus x_{63} = k_{5,22} \\ \quad \cdots \\ c_{41} \oplus x_{41} = k_{3,41} \oplus x_{18} \oplus x_{41} = k_{5,0}, \end{cases}$$

and so on. In step 7, we get:

$$\begin{cases} W_{41} \oplus U_{41} = k_{11,41} \oplus k_{10,18} \oplus k_{5,59} \oplus k_{4,59} \oplus b_{59} \oplus k_{12,0} \oplus k_{9,0} \oplus k_{8,0} \oplus c_{41} \\ \qquad\qquad = k_{13,0} \\ \quad \cdots \\ W_{63} \oplus U_{63} = k_{11,63} \oplus k_{10,40} \oplus k_{7,17} \oplus k_{4,58} \oplus b_{58} \oplus k_{12,22} \oplus k_{9,22} \oplus k_{8,22} \oplus c_{63} \\ \qquad\qquad = k_{13,22}. \end{cases}$$

At the end of the seventh step, we can get at least $128 + 23 \cdot 6 = 266$ linear equations on the bits of a, b, x and ω. Hence, we can verify the consistency of the linear system to filter out the wrong positions and solve the linear system whenever the consistency is satisfied to get the values of a, b, x and ω. The required keystream length is $7 \cdot 2^{18} \cdot 2 \cdot 2^4 \rightarrow O(2^{26})$ bytes, while the complexity is $O(2^{24} + (2^{25} - 1) \cdot 2^{24}) = O(2^{49})$.

So far, we have completely recovered the secret keys of COSvd(2,128) with complexity $O(2^{113})$ and success rate 93.4597% from $O(2^{26})$-byte keystream. Our analysis shows that it is hard to say the design of COSvd(2,128) is successful. There are quite a number of security flaws in it, although several methods have been used to strengthen the security of this cipher. The main strength of COSvd(2,128) lies in the chaotic layer, other parts of this cipher is quite weak in cryptographic sense. Although the attack parameters are determined under the double precision floating-point assumption, our attack actually works for any precision implementation of the chaotic layer. The only difference is the slightly increased complexity contributing to search the initial point (x_{-2}, x_{-1}) of one henon map. The vital weakness of this cipher is the poorly designed S-box, which is the basis of our divide-and-conquer attack.

4 Conclusion

In this paper, we showed that although several methods have been used to strengthen the security of COS(2,128), the stream cipher COSvd(2,128) is still insecure. There are some serious security weaknesses in this cipher. We developed a divide-and-conquer attack to recover the secret keys much faster than the exhaustive search. We suggest that this cipher should not be used in practice.

References

1. H. Cramer, "Mathematical Methods of Statistics", Princeton University Press, 1946.
2. E. Filiol, C. Fontaine, "A new Ultrafast stream ciphers design: COS Ciphers", *the 8th IMA Conference on Cryptography and Coding*, LNCS vol. 2260, Springer-Verlag,(2001), pp. 85-98.
3. E. Filiol, C. Fontaine and S. Josse, "The COSvd Ciphers", *The State of the Art of Stream Ciphers: Workshop Record*, Belgium, October 2004, pp.45-59.
4. D. Erdmann, S. Murphy, "Henon Stream Cipher", *Electronic Letters*, vol. 28, No.9, 1992, pp. 893-895.
5. M. Henon, "A two-dimensional mapping with a strange attactor", *Communications in Mathematical Physics*, vol. 50, 1976, pp. 69-77.
6. D. Goldberg, D. Priest, "What Every computer scientist should know about floating-point arithmetic", *ACM Comp. Surv.*, vol. 23, No.1, 1991, pp.5-48.
7. I. Mantin, A. Shamir, "A Practical Attack on Broadcast RC4", *Fast Software Encryption-FSE'2001*, LNCS vol. 2355, Springer-Verlag,(2002), pp. 152-164.
8. H. Wu, F. Bao, "Cryptanalysis of stream cipher COS (2,128) mode I", *Australian Conference on Information Security and Privacy-ACISP'2002*, LNCS vol. 2384, Springer-Verlag,(2002), pp. 154-158.
9. S. Babbage, "The COS Stream Ciphers are Extremely Weak", http://eprint.iacr.org/2001/078/
10. S. Babbage, "Cryptanalysis of the COS (2,128) stream ciphers", http://eprint.iacr.org/2001/106/

A Function Tabkey2 [3]

```
unsigned char Tabkey2[256][4] = {
{17,198,0,37},{210,240,183,153},{228,85,222,214},{130,147,36,201},
{217,169,187,173},{80,238,162,101},{135,12,178,219},{6,13,48,93},
{132,8,152,196},{227,86,199,249},{161,220,69,202},{147,53,185,244},
{241,226,101,158},{151,25,145,8},{13,27,26,186},{174,10,133,55},
{200,26,139,241},{12,17,24,180},{138,22,170,226},{213,209,166,178},
{186,247,22,193},{2,5,241,17},{167,213,104,236},{187,245,72,61},
{221,105,236,91},{29,49,33,16},{75,17,34,17},{126,156,76,74},
{26,54,52,28},{88,249,193,246},{115,187,214,110},{232,218,119,146},
{156,42,59,154},{16,33,51,24},{25,35,49,30},{207,76,217,167},
{168,103,83,212},{175,91,126,253},{211,254,77,59},{74,202,229,40},
{199,211,45,184},{134,196,23,4},{5,5,227,5},{1,3,118,159},
{155,36,15,224},{76,214,100,230},{106,136,254,122},{94,246,54,68},
{165,192,74,183},{202,31,150,123},{58,99,67,33},{154,38,97,216},
{35,100,224,34},{64,193,28,116},{252,57,153,149},{216,24,146,240},
{52,108,105,57},{85,177,252,232},{177,243,130,223},{51,102,155,44},
{230,119,175,221},{21,43,17,137},{142,183,30,25},{40,114,80,121},
{180,124,109,166},{54,106,110,43},{33,66,102,49},{60,120,198,39},
{50,70,98,60},{128,1,60,211},{55,90,8,22},{198,140,159,189},
{39,74,78,38},{42,89,84,53},{49,116,186,46},{38,73,3,56},
{41,112,82,119},{215,175,71,203},{149,148,161,81},{113,172,107,251},
{45,95,91,41},{235,64,212,78},{9,9,19,9},{10,28,255,82},
{11,23,201,10},{247,185,85,237},{3,7,237,42},{87,227,249,213},
{15,30,31,168},{159,231,163,215},{153,173,200,252,},{231,94,203,139},
{212,16,253,245},{239,215,32,58},{188,237,108,136},{14,20,29,85},
{61,122,125,54,},{253,166,225,160},{218,34,148,247},{117,210,61,66},
{116,199,134,67},{124,184,223,64},{89,163,195,248},{169,82,68,197},
{70,200,192,69},{102,207,138,199},{129,131,86,233},{190,127,238,90},
{95,252,50,6},{183,123,189,47},{246,109,124,152},{179,101,209,135},
{104,217,210,114},{32,32,65,140},{170,98,248,195},{91,165,14,50},
{146,133,38,235},{152,32,129,148},{103,205,42,89},{18,18,37,18},
{254,60,177,208},{242,96,115,32},{43,87,35,62},{140,135,47,228},
{56,83,116,51},{143,21,220,72},{81,228,160,243},{201,152,44,182},
{244,232,245,217},{114,170,239,107},{108,212,221,86},{67,134,89,109},
{66,132,204,98},{192,206,142,128},{120,241,140,79},{150,62,13,225},
{100,141,197,120},{69,139,213,113},{93,251,11,96},{83,234,180,118},
{110,180,246,45},{144,48,234,157},{203,44,63,14},{57,93,235,162},
{78,149,157,77},{160,157,79,15},{84,179,169,106},{105,255,81,29},
{99,233,117,92},{189,110,251,187},{77,146,7,112},{53,68,127,142},
{82,224,165,238},{219,46,55,144},{214,51,112,177},{173,250,111,192},
{136,189,16,163},{166,72,158,111},{226,88,215,210},{71,219,143,12},
{90,190,182,83},{176,115,99,227},{145,186,53,156},{27,45,132,11},
{19,19,39,19},{24,58,93,23},{20,56,41,164},{225,195,208,155},
{22,47,88,20},{4,4,9,133},{193,2,228,161},{250,203,66,21},
{7,15,219,84},{248,253,10,126},{174,133,242,198},{240,225,90,220},
{30,61,62,27},{109,145,202,206},{243,204,87,175},{44,81,173,124},
{195,55,144,188},{223,239,43,194},{47,126,181,3},{123,182,190,76},
{151,8,46,151},{185,181,240,97},{73,194,147,117},{194,19,149,172},
```

```
{127,158,216,104},{204,151,106,31},{28,41,58,170},{157,40,167,234},
{122,244,250,108},{148,59,40,222},{184,229,113,190},{237,6,137,130},
{131,4,6,254},{46,125,5,48},{234,164,123,132},{101,150,179,7},
{233,143,12,134},{171,137,95,191},{249,113,191,129},{220,171,25,185},
{178,71,135,200},{107,153,56,88},{68,222,136,231},{209,176,4,105},
{141,144,27,138},{72,221,154,242},{205,159,21,141},{181,107,20,255},
{251,80,172,147},{96,129,114,80},{197,121,232,181},{255,117,194,99},
{191,248,120,13},{121,230,171,87},{97,155,205,94},{23,63,218,1},
{75,0,151,75},{36,97,73,165},{63,79,92,52},{163,69,247,204},
{208,178,164,229},{92,242,184,95},{65,130,131,127},{164,235,122,239},
{236,216,231,131},{229,128,207,174},{182,75,196,100},{34,111,103,115},
{172,92,141,250},{196,14,128,70},{125,168,2,73},{98,188,244,209},
{206,154,188,179},{48,77,230,176},{37,37,75,37},{31,39,174,26},
{224,78,211,205},{139,223,1,63},{118,236,243,65},{245,191,226,150},
{86,174,70,125},{62,84,168,36},{158,138,94,207},{137,162,18,169},
{112,167,233,102},{59,118,121,171},{238,67,96,145},{79,197,57,103},
{162,201,64,143},{119,161,176,218},{222,29,156,71},{111,142,206,35}};
```

Expanding Weak PRF with Small Key Size

Kazuhiko Minematsu and Yukiyasu Tsunoo

NEC Corporation, 1753 Shimonumabe, Nakahara-Ku,
Kawasaki 211-8666, Japan
k-minematsu@ah.jp.nec.com

Abstract. We propose modes for weakly-secure block ciphers that take one block input to provide output of arbitrary length. Damgård and Nielsen proposed such a mode called the Pseudorandom Tree (PRT) mode, and demonstrated that PRT could be used to establish a communication channel that is secure against Chosen-Plaintext Attacks, if the underlying block cipher is secure against any Known-Plaintext Attacks. We present a mode that reduces the key size of PRT to about 60% without any additional computation. We call this the Extended PRT (ERT) mode and prove its security. One drawback of PRT and ERT is that their key sizes are not much small under small expansion, since functions with small expansion are important from practical point of view. We also present a mode that greatly reduces the key size under small expansion.

1 Introduction

Known-Plaintext Attack (KPA) is a weaker cryptographic attack than Chosen-Plaintext Attack (CPA), as KPA requires the attacker to use uniformly random and independent plaintexts, whereas CPA allows adaptive strategies for choosing plaintexts. KPA-secure functions and their applications have been studied by many researchers [1, 12, 13]. Modes of KPA-secure function can offer ways of using such weakly-secure functions or provide some extra security if they are used with strongly-secure functions, e.g., AES.

One excellent work on this topic was done by Damgård and Nielsen [4]. They proposed the Pseudorandom Tree (PRT) mode, which turns a KPA-secure n-bit block cipher into a KPA-secure function with n-bit input and $n\theta$-bit output for some positive integer θ where θ is the expansion rate. The PRT mode was shown to be KPA-secure if the underlying block cipher was KPA-secure. Damgård and Nielsen demonstrated that the PRT could be securely used as an additive stream cipher using a random initial vector (IV). Their proposal was simple: generate uniformly random input to the PRT and use it as a header. The ciphertext consisted of the header and the XOR of the plaintext and the PRT's output. This protocol was apparently secure against CPA, since a CPA cannot manipulate the header. Moreover, they showed that standard modes such as CTR and CBC could not be used as a mode for KPA-secure block cipher to provide communication secure against CPAs. Therefore, these standard modes cannot be a substitute for PRT. While a naive approach requires θ keys of a KPA-secure

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 284–298, 2006.
© Springer-Verlag Berlin Heidelberg 2006

block cipher if we want to expand the output θ times, PRT uses only a constant (1 or 2) number of block cipher keys, which we call master keys, and expands them to obtain $2\log_2\theta$ expanded keys. These expanded keys, that hereafter we will simply call keys, are used to form a binary tree.

In this paper, we propose two provably-secure approaches to reducing the (expanded) key size of PRT. Specifically, our focus is not on the order, but on the constant of key size, since it seems extremely difficult to improve the key size of PRT to $O(1)$ (which implies a mode with a constant-time keyscheduling), as pointed out by Damgård and Nielsen. Nonetheless, we demonstrate that the constant of key size can be greatly reduced for practical expansion rate without any additional computation. To our knowledge, this is the first attempt to improve the PRT.

The first approach, which we will call the Extended PRT (ERT) mode, is a natural extension of PRT. ERT mode inherits all the beneficial characteristics of PRT, such as optimal throughput (c block cipher invocations produce c blocks of output) and log-time random access to an output block in an arbitrary position. The key observation with ERT is that the composition operator used by PRT is not fully optimized. A simple change to the composition operator yields a mode using $2\log_3\theta$ keys, where θ is the expansion rate. This is about 60% of the PRT's key size. Since ERT merely changes the composition operator, no additional computation is needed. Although the ERT's key size is still $O(\log\theta)$, it is a very simple and practical solution to reduce the key size.

One drawback of PRT and ERT is poor performance (with respect to key size) under small expansion. This originates from the overhead involved in PRT and ERT. To solve this problem, we employed quite a different approach from them. In our second proposal, which was called the Factorial Tree (FCT) mode, we utilized all possible cascades of d distinct elements among r independent block ciphers ($d \leq r$). Here, r indicates the key size and d is a parameter called depth. FCT's throughput is optimal and random access requires at most d block cipher calls. The expansion rate of FCT is $O(r^d)$ and hence FCT's key size is asymptotically larger than that of PRT or ERT if d is a constant. However, FCT provides a smaller key size than those of PRT and ERT under a small expansion rate even if the depth is a very small constant, since FCT has no overhead. For example, if the target expansion rate is 4, PRT requires 4 keys, i.e., it achieves no improvement, while FCT with depth 2 requires only 2 keys. Moreover, FCT with depth 3 is better than PRT or ERT for expansion rate up to about 2500, which might be large enough for some practical applications. We emphasize that the expansion rate need not be much large in practice and hence the key size for small expansion is an important performance measure. For example, in the encrypted communication described before, the expansion rate need not correspond to the length of a plaintext: if the length of a plaintext is longer than the output length of a KPA-secure function, we simply issue another random IV and concatenate two outputs to get a longer keystream. The header of the ciphertext is a concatenation of two IVs. Obviously, there is a trade-off between the expansion rate and the maximum length of a header.

This paper is organized as follows. First, we give some definitions on provable security against CPA and KPA in Sect. 2. Then, we describe basic compositions for building a KPA-secure function in Sections 3.1 and 3.2. We also describe the PRT mode and its security in Sect. 3.3. In Sect. 4, we define the ERT mode and demonstrate that it effectively reduces the key size. Then we show the FCT mode, which works better than PRT or ERT for small expansion. We summarize our results and conclude in Sect. 5.

2 Preliminaries

2.1 Random Functions and Their Compositions

Definition 1. *A set of functions indexed with a random key is called a random function (RF). If* $\mathbf{F} : \mathcal{X} \to \mathcal{Y}$ *is an RF, it is distributed over* $\{f : \mathcal{X} \to \mathcal{Y}\}$*. If we want to emphasize* $\mathbf{F}$*'s key is* K*,* $\mathbf{F}_K$ *is written. Moreover, an RF:* $\{0,1\}^n \to \{0,1\}^m$ *with uniform distribution is called a uniform random function (URF) and denoted by* $\mathbf{R}_{n,m}$*. An RF with uniform distribution on all* n*-bit permutations is called a uniform random permutation (URP) and denoted by* $\mathbf{P}_n$*.*

Note that the word "random" does not imply uniformity in our definition. A random function only means that it is probabilistic. As they were in Def. 1, RFs will be written in bold symbols. If $\mathbf{F}$ and $\mathbf{G}$ have common input/output spaces, they are called compatible. Moreover, if their input/output distributions are identical, we say they are equivalent (even if their key spaces are incompatible).

Definition 2. *Let* $\mathbf{F} : \mathcal{X} \to \mathcal{Y}$*, and* $\mathbf{G} : \mathcal{Y} \to \mathcal{Z}$*, and* $\mathbf{H} : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$*. We define two operators,* $\mathbf{F} \triangleleft \mathbf{G} : \mathcal{X} \to \mathcal{Y} \times \mathcal{Z}$ *and* $\mathbf{F} \trianglelefteq \mathbf{H} : \mathcal{X} \to \mathcal{Y} \times \mathcal{Z}$*, as follows.*

$$\mathbf{F} \triangleleft \mathbf{G}(x) \stackrel{\text{def}}{=} (\mathbf{F}(x), \mathbf{G}(\mathbf{F}(x))), \text{ and } \quad \mathbf{F} \trianglelefteq \mathbf{H}(x) \stackrel{\text{def}}{=} (\mathbf{F}(x), \mathbf{H}(x, \mathbf{F}(x))).$$

2.2 Notions of Security and Their Properties

We will now describe the notions of security against CPA and KPA. All our security definitions are the same as the standard ones [2, 4]. Consider an attacker, $\mathbf{D}$, that can access the encryption oracle (EO). Here, EO has implemented $\mathbf{H}$, which is equivalent to either $\mathbf{F}$ or $\mathbf{G}$. $\mathbf{D}$ determines whether $\mathbf{H}$ is $\mathbf{F}$ or $\mathbf{G}$ after the interaction with EO. The advantage of $\mathbf{D}$ is defined as

$$\mathrm{V}(\mathbf{F}, \mathbf{G}|\mathbf{D}) \stackrel{\text{def}}{=} |\Pr[\mathbf{D}\text{'s guess is } \mathbf{F}|\mathbf{H}=\mathbf{F}] - \Pr[\mathbf{D}\text{'s guess is } \mathbf{F}|\mathbf{H}=\mathbf{G}]|. \quad (1)$$

Definition 3. *The CPA-advantage (KPA-advantage) is defined as the maximal advantage of all attackers using CPA (KPA). That is,*

$$\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{F},\mathbf{G}}(q,t) \stackrel{\text{def}}{=} \max_{\mathbf{D}:(q,t)\text{-}CPA} \mathrm{V}(\mathbf{F}, \mathbf{G}|\mathbf{D}), \mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{F},\mathbf{G}}(q,t) \stackrel{\text{def}}{=} \max_{\mathbf{D}:(q,t)\text{-}KPA} \mathrm{V}(\mathbf{F}, \mathbf{G}|\mathbf{D}).$$

Here, (q,t)-CPA denotes a CPA that uses q queries with time complexity t[1]*. Similarly, (q,t)-KPA denotes a KPA that uses q independent and uniformly random queries with time complexity t. Especially, let*

$$\mathrm{Adv}^{\mathrm{prf}}_{\mathbf{F}}(q,t) \stackrel{\mathrm{def}}{=} \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{F},\mathbf{R}}(q,t), \qquad \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,t) \stackrel{\mathrm{def}}{=} \mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{F},\mathbf{R}}(q,t), \tag{2}$$

where $\mathbf{R}$ is a URF compatible to $\mathbf{F}$.

If $\mathrm{Adv}^{\mathrm{prf}}_{\mathbf{F}}(q,t)$ ($\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,t)$) is negligible for any practical (q,t), we say $\mathbf{F}$ is a pseudorandom function (PRF) (a weak pseudorandom function, WPRF).

Let $\mathbf{F},\mathbf{G}$ be compatible RFs with n-bit input, and let $\mathbf{R}$ be the URF with n-bit output. The following equation will play an important role in our analysis.

$$\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{R}\triangleleft\mathbf{F},\mathbf{R}\triangleleft\mathbf{G}}(q,t) = \mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{F},\mathbf{G}}(q,t+O(nq)). \tag{3}$$

This is natural, since all adaptive strategies are useless in distinguishing $\mathbf{R}\triangleleft\mathbf{F}$ from $\mathbf{R}\triangleleft\mathbf{G}$. The only difference is in the way they generate uniformly random inputs for $\mathbf{F}$ and $\mathbf{G}$. We assume the time for generating q uniformly random inputs needs $O(nq)$ computation time. Hereafter, $t+O(nq)$ is denoted by t'.

It is well known that triangle inequality holds for the CPA-advantage. That is, for any $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{H}$, $\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{F},\mathbf{H}}(q,t)$ is at most $\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{F},\mathbf{G}}(q,t)+\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{G},\mathbf{H}}(q,t)$. It is easy to see that triangle inequality also holds for the KPA-advantage. Here, we will present a lemma that is useful for the analysis of the KPA-advantage.

Lemma 1. *For any RFs with n-bit input, $\mathbf{F}$ and $\mathbf{G}$, we have $\mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{F},\mathbf{G}}(q,t) \le \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{F},\mathbf{G}}(q,t)$. Moreover, let $\mathbf{E}$ be a random function with n-bit output, and $\mathbf{R}$ be the URF compatible with $\mathbf{E}$. Then,*

$$\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{E}\triangleleft\mathbf{F},\mathbf{E}\triangleleft\mathbf{G}}(q,t) \le 2\mathrm{Adv}^{\mathrm{prf}}_{\mathbf{E}}(q,t) + \mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{F},\mathbf{G}}(q,t'). \tag{4}$$

Proof. The first claim is obvious. For the second, we have

$$\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{E}\triangleleft\mathbf{F},\mathbf{E}\triangleleft\mathbf{G}}(q,t) \le \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{E}\triangleleft\mathbf{F},\mathbf{R}\triangleleft\mathbf{F}}(q,t) + \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{R}\triangleleft\mathbf{F},\mathbf{R}\triangleleft\mathbf{G}}(q,t) + \mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{R}\triangleleft\mathbf{G},\mathbf{E}\triangleleft\mathbf{G}}(q,t).$$

Combining the above inequality with Eq. (3) proves the second claim.

2.3 Applications of WPRF

From the definition of the KPA-advantage, it is clear that any WPRF can be used for secure encrypted communications. That is, to encrypt an n-bit message, m, we first generate a uniformly random input to $\mathbf{F}$, denoted by X, then $(X, \mathbf{F}(X)_{[1,\ldots,n]} \oplus m)$ is sent to the receiver. Here, $\mathbf{F}$ is a WPRF with an output length that is longer than n-bit and $\mathbf{F}(X)_{[1,\ldots,n]}$ is the first n-bit of $\mathbf{F}(X)$. $\mathbf{F}$'s key is assumed to be shared in advance of the communication. Therefore, the receiver can compute $\mathbf{F}(X)$ and then recover m. This was proposed by Damgård and Nielsen [4]. Intuitively, this encrypted communication is secure against any CPA against the communication channel (note that CPAs are not allowed to

[1] Here, the time complexity includes the worst case execution time and the program size, in some fixed RAM computation model.

manipulate the header) as long as $\mathbf{F}$ is a WPRF. Note that we can use a WPRF that has a shorter output length than the length of a plaintext by using multiple headers, as explained in Introduction.

3 PRT

3.1 Basic Expansion Operations

We describe the Damgård and Nielsen's PRT mode [4]. Its keyscheduling will not be presented since we will not discuss the keyscheduling issue. The PRT mode turns $\mathbf{F} : \{0,1\}^n \to \{0,1\}^n$ into an RF: $\{0,1\}^n \to \{0,1\}^{n\theta}$ for some positive integer θ. To describe the PRT, we need some definitions of basic expansion operations and lemmas on KPA-security of these operations.

Definition 4. *Let $\mathbf{F} : \mathcal{X} \to \mathcal{Y}$. We assume $\mathbf{F}$ has a key, K. Then, $\mathbf{F}^{\to d} : \mathcal{X} \to \mathcal{Y}^d$ and $\mathbf{F}^{d\to d} : \mathcal{X}^d \to \mathcal{Y}^d$ are defined as follows.*

$$\begin{aligned}\mathbf{F}^{\to d}(x) &= (\mathbf{F}_{K_1}(x), \mathbf{F}_{K_2}(x), \ldots, \mathbf{F}_{K_d}(x)),\\ \mathbf{F}^{d\to d}(x_1,\ldots,x_d) &= (\mathbf{F}_K(x_1), \mathbf{F}_K(x_2),\ldots,\mathbf{F}_K(x_d)),\end{aligned} \tag{5}$$

where $K_1, \ldots, K_d$ are independent and uniform keys for $\mathbf{F}$. The key for $\mathbf{F}^{\to d}$ is a set of d keys of $\mathbf{F}$.

3.2 Lemmas on KPA-Security of Basic Operations

The following two lemmas were shown by Damgård and Nielsen.

Lemma 2. *Let $\mathbf{F} : \{0,1\}^n \to \{0,1\}^n$. Then,* $\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}^{\to d}}(q,t) \le d \cdot \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,t)$.

Lemma 3. *Let $\mathbf{F} : \{0,1\}^m \to \{0,1\}^n$. Then,* $\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}^{d\to d}}(q,t) \le \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(dq,t) + (d^2q^2)/2^{m+1}$.

We will now prove the security for $\triangleleft$ and $\trianglelefteq$ composition operators. The security for the $\trianglelefteq$ operator is entirely original.

Lemma 4. *Let $\mathbf{F} : \{0,1\}^m \to \{0,1\}^n$, and $\mathbf{G} : \{0,1\}^n \to \{0,1\}^{n'}$, and $\mathbf{H} : \{0,1\}^{n+m} \to \{0,1\}^{n'}$. Let us assume $m = O(n)$. Then,*

$$\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}\triangleleft\mathbf{G}}(q,t) \le \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,t) + \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{G}}(q,t') + \frac{q^2}{2^{n+1}}, \tag{6}$$

$$\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}\trianglelefteq\mathbf{H}}(q,t) \le \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,t) + \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{H}}(q,t') + \frac{q^2}{2^m}. \tag{7}$$

Proof. The first claim was proved in Lemma 2 of Damgård and Nielsen [4]. For the second, We have

$$\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}\trianglelefteq\mathbf{H}}(q,t) \le \mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{F}\trianglelefteq\mathbf{H},\mathbf{R}_{m,n}\trianglelefteq\mathbf{H}}(q,t) + \mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{R}_{m,n}\trianglelefteq\mathbf{H},\mathbf{R}_{m,n}\trianglelefteq\mathbf{R}_{n+m,n'}}(q,t), \tag{8}$$

since $\mathbf{R}_{m,n} \trianglelefteq \mathbf{R}_{n+m,n'}$ is equivalent to $\mathbf{R}_{m,n+n'}$. Clearly, the first term at the rhs of Eq. (8) equals $\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,t)$. Let us analyze the second term. Since a collision among inputs is useless for an attacker, $\mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{R}_{m,n} \trianglelefteq \mathbf{H}, \mathbf{R}_{m,n} \trianglelefteq \mathbf{R}_{n+m,n'}}(q,t)$ is at most the maximum advantage in distinguishing $\mathbf{R}_{m,n} \trianglelefteq \mathbf{H}$ from $\mathbf{R}_{m,n} \trianglelefteq \mathbf{R}_{n+m,n'}$ under an input distribution that is uniform on all elements in $(\{0,1\}^m)^q$ with no colliding inputs (i.e., the uniform distribution on the set of $(x_1, \ldots, x_q)$, where $x_i \in \{0,1\}^m$ and $x_i \neq x_j$ if $i \neq j$). This input distribution is denoted by Pd^q_m. Similarly, the uniform distribution on $(\{0,1\}^m)^q$ is denoted by Pu^q_m. For both $\mathbf{R}_{m,n} \trianglelefteq \mathbf{H}$ and $\mathbf{R}_{m,n} \trianglelefteq \mathbf{R}_{n+m,n'}$, if inputs have no collisions, then the leftmost m-bit of outputs are independent and uniform. This implies that we only have to evaluate the maximum advantage in distinguishing $\mathbf{H}$ and $\mathbf{R}_{n+m,n'}$ under the input distribution such that the leftmost n-bits (the rightmost m-bits) are distributed according to Pu^q_n (Pd^q_m). Note that this input distribution can be simulated with $\mathbf{R}_{m,n} \| \mathbf{P}_m$, which denotes RF: $\{0,1\}^m \to \{0,1\}^{n+m}$ such that $\mathbf{R}_{m,n} \| \mathbf{P}_m(x) = (\mathbf{R}_{m,n}(x), \mathbf{P}_m(x))$. From these observations, we obtain

$$\begin{aligned}\mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{R}_{m,n} \trianglelefteq \mathbf{H}, \mathbf{R}_{m,n} \trianglelefteq \mathbf{R}_{n+m,n'}}(q,t) &\le \mathrm{Adv}^{\mathrm{kpa}}_{(\mathbf{R}_{m,n} \| \mathbf{P}_m) \triangleleft \mathbf{H}, (\mathbf{R}_{m,n} \| \mathbf{P}_m) \triangleleft \mathbf{R}_{n+m,n'}}(q,t) \\ &\le 2 \cdot \mathrm{Adv}^{\mathrm{prf}}_{\mathbf{R}_{m,n} \| \mathbf{P}_m}(q,t) + \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{H}}(q,t'). \quad (9)\end{aligned}$$

where the second inequality comes from Eq. (4) and the fact that $t' = t + O(nq)$. Then, $\mathrm{Adv}^{\mathrm{prf}}_{\mathbf{R}_{m,n} \| \mathbf{P}_m}(q,t)$ is clearly no more than $\binom{q}{2} \frac{1}{2^m} \le \frac{q^2}{2^m}$. This concludes the proof.

3.3 Description of PRT and Its KPA-Security

Definition 5. *Let* $\mathbf{G}$ *be* $\mathbf{F}^{\to 2}$*, where* $\mathbf{F} : \{0,1\}^n \to \{0,1\}^n$ *has an* n*-bit key. Therefore* $\mathbf{G}$ *has a* $2n$*-bit key. The PRT mode of* $\mathbf{G}$ *with depth* d *is defined as* $\mathrm{PRT}[\mathbf{G}]_d \stackrel{\mathrm{def}}{=} \mathbf{G}_1 \triangleleft \mathbf{G}_2^{2\to 2} \triangleleft \mathbf{G}_3^{4\to 4} \triangleleft \ldots \triangleleft \mathbf{G}_d^{2^{d-1}\to 2^{d-1}}$*. Here,* $\mathbf{G}_1, \ldots \mathbf{G}_d$ *are* d *independent versions of* $\mathbf{G} = \mathbf{F}^{\to 2}$*. The key of* $\mathrm{PRT}[\mathbf{G}]_d$ *is* $2nd$*-bit (i.e.* $2d$ *keys of* $\mathbf{F}$*). In PRT, the composition operators are treated in descending order. For example,* $\mathbf{G}_1 \triangleleft \mathbf{G}_2^{2\to 2} \triangleleft \mathbf{G}_3^{4\to 4}$ *denotes* $(\mathbf{G}_1 \triangleleft (\mathbf{G}_2^{2\to 2} \triangleleft \mathbf{G}_3^{4\to 4}))$ *(see left of Fig.1).*

The proof of security for PRT is as follows.

Theorem 1. *(Theorem 2 of Damgård and Nielsen [4])*

$$\mathrm{Adv}^{\mathrm{wprf}}_{\mathrm{PRT}[\mathbf{G}]_d}(q,t) \le 2d \cdot \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(2^{d-1}q, t') + \frac{q^2 2^{2d}}{2^{n+2}} + \frac{dq^2}{2^{2n+1}}. \quad (10)$$

The PRT mode has various attractive properties. Its throughput is optimal, as it requires only one invocation of $\mathbf{F}$ to generate one block output. In addition, random access to an arbitrary block of an output sequence can be done with at most $\log_2 \theta$ invocations of $\mathbf{F}$, where θ is the expansion rate. Finally, PRT requires $2d = 2(\log_2(\theta + 2) - 1)$ keys of $\mathbf{F}$.

4 Our Proposal

4.1 On the Difficulty of Achieving Constant Expanded Key Size

As mentioned earlier, our purpose is to reduce the (expanded) key size. Since PRT's key size is $O(\log \theta)$, it is natural to ask whether this can be efficiently reduced to $O(1)$, i.e., a constant. However, we found that this was very hard to achieve. For example, if a mode uses feedback (i.e. if there are always two output blocks such as $(\mathbf{F}_K(x), \mathbf{F}_K(\mathbf{F}_K(x)))$, we can provide an example that makes the mode completely vulnerable to KPA. That is, the uniformly random involution (URI), which is distributed uniformly over all involutions[2]. URI is secure against non-adaptive CPAs and therefore WPRF. However, URI can easily be distinguished from URF when feedback has occurred[3].

If we could efficiently convert a WPRF into a compatible PRF using constant keys of the WPRF, then the OFB mode of the resulting PRF would provide a desirable solution (it is easy to verify that the OFB mode of a PRF is a WPRF with a large output). For some special WPRFs, such as two-round random Feistel[4], we have an efficient WPRF-to-PRF conversion: a cascade of these special WPRFs actually yields a PRF [8, 14]. However, cascading does not work for all WPRFs. For instance, a random function that has one fixed I/O point and behaves as URF for other inputs. Although this random function is clearly a WPRF, a cascade of it can never be a PRF. In principle, WPRF-to-PRF conversion is generally possible by combining the keyscheduling of Damgård and Nielsen, which turns a WPRF into a pseudorandom generator (PRG), and a PRG-to-PRF conversion (e.g., Goldreich-Goldwasser-Micali [6]). One can also use the Naor-Reingolds' approach [15]. However, they will take much computation and hence is impractical.

Consequently, the order of key size seems difficult to be improved efficiently. However, in the next section we demonstrate that the *constant* of key size can be greatly reduced without any additional computation.

4.2 ERT: Reduced Key Size for Free

In the tree defining PRT, the output made by the i-th layer is delivered to the $(i+1)$-th layer as input. This composition is not fully optimized and there is a room for improvement to reduce the key size (or, equivalently, expand the output length with the same key size). We changed the composition operator so that *the concatenation of input and output* for the i-th layer was delivered to the $(i+1)$-th layer. That is, we used $\trianglelefteq$ instead of $\triangleleft$. We will now describe the Extended Pseudorandom Tree (ERT) mode.

Definition 6. *(ERT) Let* $\mathbf{G}$ *be* $\mathbf{F}^{\rightarrow 2}$*, where* $\mathbf{F} : \{0,1\}^n \rightarrow \{0,1\}^n$*. The ERT of* $\mathbf{G}$ *with depth* d *is denoted by* $\mathrm{ERT}[\mathbf{G}]_d$*. It is defined as follows.*

[2] f is a involution if $f(f(x)) = x$ for all input x.

[3] We also tried similar modes such as $(\mathbf{F}_K(x), \mathbf{F}_K(\mathbf{F}_K(x) \oplus x))$ and found some variants of URI to be counterexamples.

[4] Two-round random Feistel has two round functions that are independent PRFs.

$$\mathrm{ERT}[\mathbf{G}]_d \stackrel{\text{def}}{=} \mathbf{G}_1 \trianglelefteq \mathbf{G}_2^{3\to 3} \trianglelefteq \mathbf{G}_3^{9\to 9} \trianglelefteq \ldots \trianglelefteq \mathbf{G}_d^{3^{d-1}\to 3^{d-1}}, \tag{11}$$

where the operators have been processed in ascending order. For example, $\mathbf{G}_1 \trianglelefteq \mathbf{G}_2^{3\to 3} \trianglelefteq \mathbf{G}_3^{9\to 9}$ *denotes* $((\mathbf{G}_1 \trianglelefteq \mathbf{G}_2^{3\to 3}) \trianglelefteq \mathbf{G}_3^{9\to 9})$ *(see right of Fig.1). The key for* $\mathrm{ERT}[\mathbf{G}]_d$ *is the* $2d$ *keys of* $\mathbf{F}$.

To achieve expansion rate θ, PRT needs $2(\log_2(\theta+2)-1)$ keys of $\mathbf{F}$, while ERT needs $2\log_3(\theta+1)$ keys. Thus, ERT successfully reduces the key size to about $\log_3 2 \sim 63\%$. The following theorem proves the KPA-advantage of ERT.

Theorem 2. *Let* $\mathbf{G}$ *be* $\mathbf{F}^{\to 2}$*, where* $\mathbf{F} : \{0,1\}^n \to \{0,1\}^n$ *with an* n*-bit key.*

$$\mathrm{Adv}^{\mathrm{wprf}}_{\mathrm{ERT}[\mathbf{G}]_d}(q,t) \le 2d\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(3^{d-1}q,t') + \frac{q^2 3^{2d}}{2^{n+4}} + \frac{dq^2}{2^n}.$$

Proof.

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{wprf}}_{\mathrm{ERT}[\mathbf{G}]_d}(q,t) &\le \sum_{i=0}^{d-1} \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{G}_{i+1}^{3^i\to 3^i}}(q,t') + \frac{q^2}{2^{3^i n}} \\
&\le \sum_{i=0}^{d-1} \left(\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{G}_{i+1}}(3^i q,t') + \frac{q^2 3^{2i}}{2^{n+1}} \right) + \frac{dq^2}{2^n}, && (12)\\
&\le d \cdot \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{G}}(3^{d-1}q,t') + \frac{q^2}{2^{n+1}} \left(\sum_{i=0}^{d-1} 9^i \right) + \frac{dq^2}{2^n}, && (13)\\
&\le 2d \cdot \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(3^{d-1}q,t') + \frac{3^{2d}q^2}{2^{n+4}} + \frac{dq^2}{2^n}, && (14)
\end{aligned}$$

where the first inequality follows from the second claim of Lemma 4, the second from Lemma 3, and the last from 2.

ERT inherits all the beneficial properties of PRT, such as optimal throughput and log-time random access to outputs. In addition, the KPA-security of ERT is comparable to that of PRT, as demonstrated by Theorems 1 and 2. Although the ERT's key size is still $O(\log\theta)$, ERT is a very simple and practical solution to reduce the key size for an arbitrarily large expansion rate.

4.3 FCT: A Mode Without Overhead

Although ERT provides a simple and effective improvement, it does not provide a great improvement for small expansion (say, less than 20). For example, PRT and ERT requires 4 keys to achieve the expansion rate 4. This is because both PRT and ERT are based on $\mathbf{F}^{\to 2}$ operation. As mentioned, a WPRF with a small expansion is often sufficient for practical applications. This is particularly true for a communication system where most plaintexts are very short (e.g., for typical IP networks, roughly 40% of the packets are 40 bytes long [7]). Therefore, it would be better if any improvement is possible for a small expansion rate. In this section, we propose a different concept from PRT and ERT as a solution to this problem. Let r and d be positive integers such that $d \le r$. Let $\sigma(r,d) \stackrel{\text{def}}{=}$

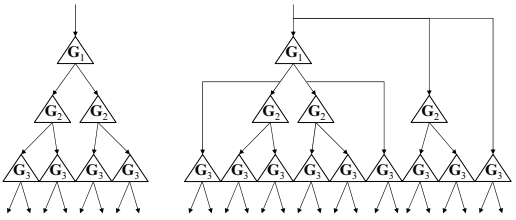


Fig. 1. PRT (left) and ERT (right) with depth 3. Note that every output of $\mathbf{G}_i$ for $i = 1, 2, 3$ is a part of the global output. The output length of PRT is 14-block, whereas that of ERT is 26-block.

$\sum_{i=1}^{d} r!/(r-i)!$. Here, 0! is defined as 1. We propose a mode that requires r keys of $\mathbf{F} : \{0,1\}^n \to \{0,1\}^n$ and achieves expansion rate $\sigma(r,d)$. Here d is a parameter and is called depth.

Definition 7. *Let* $\mathbf{F}_1, \ldots, \mathbf{F}_r$ *be* r *independent versions of* $\mathbf{F}$: $\{0,1\}^n \to \{0,1\}^n$. *The Factorial Tree (FCT) of* $\mathbf{F}$ *with* r *keys and depth* d *is defined as follows. Generate all possible cascades consisting of at most* d *elements of* $\{\mathbf{F}_1, \ldots, \mathbf{F}_r\}$ *(there are* $\sigma(r,d)$ *cascades). When an* n*-bit input,* x*, is given, the FCT sends* x *to all cascades as their input. All outputs of these cascades are concatenated to form the output for the FCT.*

For example, the FCT of $\mathbf{F}$ with setting $(r,d) = (3,3)$, is shown in Fig. 2. FCT's throughput is optimal, and the key of FCT can be generated by the keyscheduling of Damgård and Nielsen's with a slight degradation in KPA-advantage.

In principle, the FCT with the maximum depth (i.e., $d = r$) outperforms any of other modes and seems to be the best we can (with respect to the key size). However, if the depth is too much large, it might be difficult to implement it. Nevertheless, we found that the FCT with a small constant depth was useful if the target expansion rate was not so large. For example, FCT with depth two requires only two keys to achieve expansion rate 4 (remember that 4 keys are needed for PRT and ERT). Moreover, FCT with depth three works better than PRT and ERT for expansion rate up to 2500, which might be practically large enough for some applications.

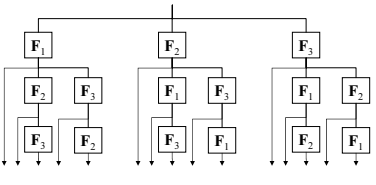


Fig. 2. FCT with three keys and depth three

4.4 KPA-Security of FCT

Analyzing the security of FCT becomes more complicated, as depth d increases. Therefore, we present a theorem in this section describing the KPA-advantage of FCT in recursive form with respect to d. With this theorem, proving the KPA-advantage for a small d is easy. In principle, however, we can derive the proof for any d up to r using the theorem.

Before stating the theorem, let us informally explain why FCT is a WPRF. The key idea behind FCT is to remove feedback caused by *one* RF (see Sect. 4.1). We therefore designed FCT so that such feedback could not occur with high probability with any KPA. In contrast, feedback involving two or more RFs with independent keys (e.g., $(\mathbf{F}_{K_1}(x), \mathbf{F}_{K_2}(\mathbf{F}_{K_1}(x)))$) is not a problem. FCT effectively uses feedback involving independent components. Note that a cascade of WPRFs can not generally be a PRF, as mentioned earlier. We can nevertheless prove that FCT is a WPRF.

The general theorem for KPA-security with FCT is as follows.

Theorem 3. *Let* $\mathrm{FCT}_{r,d}[\mathbf{F}]$ *denote the FCT of* $\mathbf{F} : \{0,1\}^n \to \{0,1\}^n$ *with depth* d *and* r *keys. Consider the substitution* $\mathbf{F}_1, \ldots, \mathbf{F}_{i-1}$ *in* $\mathrm{FCT}_{r,d}[\mathbf{F}]$ *with* $i-1$ *independent URFs, denoted by* $\mathbf{R}^{(1)}, \ldots, \mathbf{R}^{(i-1)}$*, compatible to* $\mathbf{F}$*. With this substitution, the resulting RF compatible to* $\mathrm{FCT}_{r,d}[\mathbf{F}]$ *is denoted by* $\mathrm{FCT}^{(i)}_{r,d}[\mathbf{F}]$*. Clearly,* $\mathrm{FCT}^{(1)}_{r,d}[\mathbf{F}]$ *corresponds to* $\mathrm{FCT}_{r,d}[\mathbf{F}]$ *and* $\mathrm{FCT}^{(r+1)}_{r,d}[\mathbf{F}]$ *corresponds to* $\mathrm{FCT}_{r,d}[\mathbf{R}_{n,n}]$*. Then,* $\mathrm{Adv}^{\mathrm{wprf}}_{\mathrm{FCT}_{r,d}[\mathbf{F}]}(q,t)$ *is at most*

$$2\sum_{i=1}^{r} \mathrm{Adv}^{\mathrm{wprf}}_{\mathrm{FCT}^{(i)}_{r-1,d-1}[\mathbf{F}]}(q,t') + r\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q(\sigma'+1),t') + \frac{rq^2(3(\sigma')^2+1)}{2^n}, \quad (15)$$

where σ *and* σ' *denote* $\sigma(r,d)$ *and* $\sigma(r-1,d-1)$*, respectively.*

Proof. Let us abbreviate $\mathrm{FCT}^{(i)}_{r,d}[\mathbf{F}]$ to $\mathbf{L}^{(i)}_{r,d}$. From triangle inequality, we have

$$\mathrm{Adv}^{\mathrm{wprf}}_{\mathrm{FCT}_{r,d}[\mathbf{F}]}(q,t) \le \sum_{i=1}^{r} \mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{L}^{(i)}_{r,d},\mathbf{L}^{(i+1)}_{r,d}}(q,t) + \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{L}^{(r+1)}_{r,d}}(q,t). \quad (16)$$

Note that the difference between $\mathbf{L}^{(i)}_{r,d}$ and $\mathbf{L}^{(i+1)}_{r,d}$ is only in their i-th component, and when x is a global input issued by the attacker, the inputs to their i-th components are $(x, \mathbf{L}^{(i)}_{r-1,d-1}(x))$ for both $\mathbf{L}^{(i)}_{r,d}$ and $\mathbf{L}^{(i+1)}_{r,d}$ (note that $(x, \mathbf{L}^{(i)}_{r-1,d-1}(x))$ is $\sigma'+1$ blocks). Therefore, $\mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{L}^{(i)}_{r,d},\mathbf{L}^{(i+1)}_{r,d}}(q,t)$ corresponds to the maximum advantage in distinguishing $\mathbf{F}_i$ and $\mathbf{R}^{(i)}$ under the condition that the inputs are generated by $\mathbf{R}_{n,n} \triangleleft \mathbf{L}^{(i)}_{r-1,d-1}$. The outputs of $\mathbf{R}_{n,n} \triangleleft \mathbf{L}^{(i)}_{r-1,d-1}$, which consists of $\sigma'+1$ blocks, may not be uniform. However, Eq. (4) shows us how to evaluate the maximum advantage under non-uniform inputs. Combining Eq. (4) and the above observation, we obtain

$$\mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{L}^{(i)}_{r,d},\mathbf{L}^{(i+1)}_{r,d}}(q,t) \le 2\mathrm{Adv}^{\mathrm{prf}}_{\mathbf{R}_{n,n} \triangleleft \mathbf{L}^{(i)}_{r-1,d-1}}(q,t) + \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}_i}(q(\sigma'+1),t'). \quad (17)$$

Then, we have

$$\begin{aligned}\mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{L}^{(i)}_{r,d},\mathbf{L}^{(i+1)}_{r,d}}(q,t) &\le 2\mathrm{Adv}^{\mathrm{cpa}}_{\mathbf{R}_{n,n}\triangleleft\mathbf{L}^{(i)}_{r-1,d-1},\mathbf{R}_{n,n}\triangleleft\mathbf{R}_{n,\sigma' n}}(q,t)\\ &\quad + 2\mathrm{Adv}^{\mathrm{prf}}_{\mathbf{R}_{n,n}\triangleleft\mathbf{R}_{n,\sigma' n}}(q,t) + \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}_i}(q(\sigma'+1),t'), \qquad (18)\\ &\le 2\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{L}^{(i)}_{r-1,d-1}}(q,t') + \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q(\sigma'+1),t') + q^2/2^n, \qquad (19)\end{aligned}$$

where the first inequality results from triangle inequality and Eq. (17), and the second inequality results from Eq. (3) and trivial collision analysis of $\mathbf{R}_{n,n} \triangleleft \mathbf{R}_{n,\sigma' n}$ and $\mathbf{R}_{n,n(\sigma'+1)}$. Then, the remaining task is to analyze $\mathbf{L}^{(r+1)}_{r,d}$. We have the following lemma, which is proved in Appendix A.

Lemma 5

$$\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{L}^{(r+1)}_{r,d}}(q,t) \le \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{L}^{(r+1)}_{r,d}}(q,\infty) \le \frac{3rq^2(\sigma')^2}{2^n}. \qquad (20)$$

Combining Eqs. (16) and (19) and lemma 5 proves the theorem.

Theorem 3 may not be intuitive. Roughly speaking, it tells us that we can prove the KPA-advantage of $\mathrm{FCT}_{r,d}[\mathbf{F}]$ based on that of $\mathrm{FCT}_{r-1,d-1}[\mathbf{F}]$. Since d is at most r, an iterated use of Theorem 3 will yield term $\mathrm{FCT}_{r-d,1}[\mathbf{F}]$, which is equivalent to $\mathbf{F}^{\to(r-d)}$ and hence we can easily prove its KPA-advantage using Lemma 2. From Theorem 3, we derive corollaries to prove the security of FCT with depths of 2 and 3. Although it is theoretically possible to demonstrate proofs with depths greater than 3, they would be too long to describe here.

Corollary 1

$$\mathrm{Adv}^{\mathrm{wprf}}_{\mathrm{FCT}_{r,2}[\mathbf{F}]}(q,t) \le r^2\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,t') + r\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(rq,t') + \frac{3r^3q^2}{2^n}.$$

Proof. Let us abbreviate $\mathrm{FCT}^{(i)}_{r,d}[\mathbf{F}]$ to $\mathbf{L}^{(i)}_{r,d}$. Note that $\mathbf{L}^{(i)}_{r-1,1}$ is RF: $\{0,1\}^n \to \{0,1\}^{(r-1)n}$ that consists of $r-i$ independent $\mathbf{F}$s and $i-1$ independent URFs. This indicates that $\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{L}^{(i)}_{r-1,1}}(q,t)$ is at most $(r-i)\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,t)$. Thus, Theorem 3 tells us that $\mathrm{Adv}^{\mathrm{wprf}}_{\mathrm{FCT}_{r,2}[\mathbf{F}]}(q,t)$ is at most

$$\begin{aligned}&2\sum_{i=1}^{r}(r-i)\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,t') + r\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(rq,t') + \frac{rq^2(3(r-1)^2+1)}{2^n}\\ &\le r^2\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,t') + r\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(rq,t') + \frac{3r^3q^2}{2^n}. \qquad (21)\end{aligned}$$

This proves Corollary 1.

A proof of the security for $\mathrm{FCT}_{r,3}[\mathbf{F}]$ is similarly obtained. The proof of Corollary 2 is in Appendix B.

Corollary 2

$$\mathrm{Adv}^{\mathrm{wprf}}_{\mathrm{FCT}_{r,3}[\mathbf{F}]}(q,t) \leq r^3\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,t') + r^2\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(rq,t') + r\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(r^2q,t') + \frac{3r^5q^2}{2^n}.$$

5 Conclusion

We presented two proposals to reduce the key size of the PRT. All our proposed modes inherited the beneficial features of PRT, such as optimal throughput and fast random access with no additional computation. Although they do not improve the order of key size, they perform much better than PRT for practical expansion rate. Table 1 shows key sizes and expansion rates of PRT and our proposals.

Table 1. Expansion rate of PRT and our proposed modes

key size	PRT	ERT	$\mathrm{FCT}_{r,2}$	$\mathrm{FCT}_{r,3}$
2	2	2	4	4
4	6	8	16	40
6	14	26	36	156
8	30	80	64	400
10	62	242	100	820
12	126	728	144	1464
14	254	2186	196	2380

As shown by Table 1, FCT with a small depth works better than PRT or ERT for small expansion. If a larger expansion rate is required, ERT is the best among these four modes. We can further reduce the key size by combining FCT with ERT (e.g., $\mathrm{ERT}[\mathrm{FCT}_{r,2}[\mathbf{F}]]_d$), though the key size will be still $O(\log\theta)$.

If we fix expansion rate θ, the KPA-advantages of PRT and ERT are roughly the same (they are about $2\log_2\theta \cdot \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(\theta q,t') + \frac{q^2\theta^2}{2^n}$). In contrast, a strict security comparison between PRT and FCT is difficult even if the FCT's depth is limited. We only confirmed that all the KPA-advantages of PRT, ERT, and FCT with a depth 2 or 3 were close to 1 when q was about $2^{(n/2)} - \log_2\theta$. That is, all modes can be vulnerable to the birthday attack.

PRT and ERT seem to be a PRF if the underlying block cipher is also a PRF. In contrast, FCT is totally vulnerable to CPA even if independent URFs are used as components. This difference may originate from the FCT's optimized design as a mode of WPRF. It would be interesting to prove the CPA-advantages of PRT and ERT when they use PRFs. Some CPA-secure modes of PRF that provides an arbitrary large expansion rate using only one key are already known (e.g., Gilbert's modified counter mode [5]), so it is interesting to investigate whether PRT and ERT can offer significantly higher CPA-security than the previous modes. Also, it would be interesting to investigate if we could build a KPA-secure mode with the optimal throughput that requires fewer keys than that required for the FCT with the maximal depth.

References

1. W. Aiello, S. Rajagopalan, and R. Venkatesan. "High-speed Pseudorandom Number Generation with Small Memory." *Fast Software Encryption*, FSE'99, LNCS 1636, pp. 290-304, 1999.
2. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. " A Concrete Security Treatment of Symmetric Encryption." *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, FOCS '97, pp. 394-403, 1997.
3. M. Bellare, J. Kilian and P. Rogaway. "The Security of Cipher Block Chaining." *Advances in Cryptology*- CRYPTO'94, LNCS 839, 1994.
4. I. Damgård and J. Nielsen. "Expanding Pseudorandom Functions; or: From Known-Plaintext Security to Chosen-Plaintext Security." *Advances in Cryptology*- CRYPTO'02, LNCS 2442, pp. 449-464, 2002.
5. H. Gilbert. "The Security of "One-Block-to-Many" Modes of Operation." *Fast Software Encryption*, FSE'03, LNCS 2887, pp. 377-395, 2003.
6. O. Goldreich, S. Goldwasser, and S. Micali. "How to Construct Random Functions." *Journal of the ACM*, vol.33, no.4, pp. 792-807, 1986.
7. A. Klemm, C. Lindemann, and M. Lohmann. "Traffic Modeling of IP Networks Using the Batch Markovian Arrival Process." *Computer Performance Evaluation / TOOLS 2002*, pp. 92-110, 2002.
8. M. Luby and C. Rackoff. "How to Construct Pseudo-random Permutations from Pseudo-random functions." *SIAM J. Computing*, vol.17, no.2, pp. 373-386, 1988.
9. U. Maurer and J. L. Massey. "Cascade Ciphers: The Importance of Being First." *J. Cryptology* vol.6, no.1, pp. 55-61, 1993.
10. U. Maurer. "Indistinguishability of Random Systems." *Advances in Cryptology*- EUROCRYPT'02, LNCS 2332, pp. 110-132, 2002.
11. U. Maurer and K. Pietrzak. "Composition of Random Systems: When Two Weak Make One Strong." *Theory of Cryptography* - TCC'04, LNCS 2951, pp. 410-427, 2004.
12. M. Naor, O. Reingold. "Number-theoretic Constructions of Efficient Pseudorandom Functions." 38 *th Annual Symposium on Foundations of Computer Science*, FOCS'97, pp. 458-467, 1997.
13. M. Naor and O. Reingold. "From Unpredictability to Indistinguishability: A Simple Construction of Pseudo-Random Functions from MACs (Extended Abstract)." *Advances in Cryptology* - CRYPTO'98, LNCS 1462, pp. 267-282, 1998.
14. M. Naor and O. Reingold. "On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited." *Journal of Cryptology*, vol. 12(1), pp. 29-66, 1999.
15. M. Naor and O. Reingold. "Synthesizers and their application to the parallel construction of pseudo-random functions." *J. of Computer and Systems Sciences*, Vol. 58 (2), pp. 336-375, 1999.
16. 3rd Generation Partnership Project. `http://www.3gpp.org`
17. S. Vaudenay. "Feistel Ciphers with L_2-Decorrelation." *Selected Areas in Cryptography* - SAC'98, LNCS 1556, pp. 1-14, 1998.
18. `http://www.nlanr.net/NA/Learn/packetsizes.html`

A Proof of Lemma 5

Let us evaluate $\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{L}^{(r+1)}_{r,d}}(q, \infty)$. For $1 \leq i \leq r$, let J_i denote the event that all inputs to $\mathbf{F}_i$ are distinct throughout the (q,t)-KPA. Note that J_i is defined on

the q input/output pairs of $\mathbf{L}_{r,d}^{(r+1)}$ and hence, we can define J_i for any random function $\{0,1\}^n \to \{0,1\}^{\sigma n}$. It is not difficult to see that $\mathbf{L}_{r,d}^{(r+1)}$ and $\mathbf{R}_{n,\sigma(r,d)n}$ are equivalent given $J \stackrel{\text{def}}{=} J_1 \wedge J_2 \wedge \cdots \wedge J_r$. More precisely, note that

$$\Pr[\mathbf{L}_{r,d}^{(r+1)}(x_j) = y_j, j = 1, \ldots, q \,|\, J] = \Pr[\mathbf{R}_{n,\sigma n}(x_j) = y_j, j = 1, \ldots, q \,|\, J] \quad (22)$$

holds for any $x^q \in (\{0,1\}^n)^q$ and $y^q \in (\{0,1\}^{\sigma n})^q$ satisfying J (i.e., both sides are 0 if (x^q, y^q) fails to satisfy J, and $|\{y^q \in (\{0,1\}^{\sigma n})^q : (x^q, y^q) \text{ satisfies } J\}|^{-1}$ otherwise). This equation enables us to analyze the *non-adaptive* CPA-advantage between $\mathbf{L}_{r,d}^{(r+1)}$ and $\mathbf{R}_{n,\sigma n}$. For events A and B defined for both $\mathbf{L}_{r,d}^{(r+1)}$ and $\mathbf{R}_{n,\sigma n}$, let $P_{A|B,x^q}$ and $Q_{A|B,x^q}$ be the conditional probabilities of A when B is given and q inputs, denoted by x^q, are fed into $\mathbf{L}_{r,d}^{(r+1)}$ and $\mathbf{R}_{n,\sigma n}$, respectively. Let *gue* be the event that the attacker's guess is $\mathbf{L}_{r,d}^{(r+1)}$. The maximum non-adaptive CPA-advantage without computational restriction is written as

$$\begin{aligned}
&\max_{x^q} \left(|P_{gue|J,x^q} \cdot P_{J|x^q} - P_{gue|J,x^q} \cdot Q_{J|x^q}| + |P_{gue|\overline{J},x^q} \cdot P_{\overline{J}|x^q} - P_{gue|\overline{J},x^q} \cdot Q_{\overline{J}|x^q}| \right) \\
&\leq \max_{x^q} \left(|P_{\overline{J}|x^q} - Q_{\overline{J}|x^q}| + \max\{P_{\overline{J}|x^q}, Q_{\overline{J}|x^q}\} \right) = 2 \max \left\{ \max_{x^q} P_{\overline{J}|x^q}, \max_{x^q} Q_{\overline{J}|x^q} \right\}, \quad (23)
\end{aligned}$$

where the maximums are taken over all x^qs that have no collisions. Since the optimal non-adaptive CPA is stronger than any KPA, Eq. (23) is also the upper bound for the KPA-advantage without the computational restriction. The remaining task is to evaluate Eq. (23).

Using an inductive analysis, we can verify that $P_{\overline{J}|x^q}$ consists of collision events that have a success probability of $\frac{1}{2^n}$. Therefore, what we have to do is to count the number of collision events defining $\overline{J}$. Let us consider the events defining $\overline{J_1}$. The events are classified into two types: (I) a collision between two output blocks and (II) a collision between an output block and an input. The number of type (I) collision events in $\overline{J_1}$ is $\binom{q\sigma'}{2}$ (recall that there are $q\sigma'$ output blocks that are fed into $\mathbf{R}^{(i)}$ for each i), and the number of type (II) collision events is $q^2\sigma'$. Using a symmetrical argument, it follows that the total number of collision events in $\overline{J}$ is at most $r(\binom{q\sigma'}{2} + q^2\sigma')$ and thus,

$$P_{\overline{J}|x^q} \leq \frac{r}{2^n} \left(\binom{q\sigma'}{2} + q^2\sigma' \right) \leq \frac{3rq^2\sigma'^2}{2^{n+1}}. \quad (24)$$

A similar analysis can be done on $Q_{\overline{J}|x^q}$, and $Q_{\overline{J}|x^q}$ is shown to have the same upper bound as the rhs of Eq. (24). Applying these upper bounds to Eq. (23), we have

$$\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{L}_{r,d}^{(r+1)}}(q,t) \leq \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{L}_{r,d}^{(r+1)}}(q,\infty) \leq \frac{3rq^2\sigma'^2}{2^n}. \quad (25)$$

This proves Lemma 5.

B Proof of Corollary 2

Let us abbreviate $\mathrm{FCT}^{(i)}_{r,d}[\mathbf{F}]$ to $\mathbf{L}^{(i)}_{r,d}$. Let $\sigma \stackrel{\mathrm{def}}{=} \sigma(r,3) = r^3 - 2r^2 + 2r$ and $\sigma' \stackrel{\mathrm{def}}{=} \sigma(r-1,2) = (r-1)^2$ and $\sigma'' \stackrel{\mathrm{def}}{=} \sigma(r-2,1) = r-2$ and $\sigma''' \stackrel{\mathrm{def}}{=} \sigma(r-3,1) = r-3$. To apply Theorem 3 to $\mathrm{FCT}_{r,3}[\mathbf{F}]$, we have to analyze $\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{L}^{(i)}_{r-1,2}}(q,t)$. It is done as

$$
\begin{aligned}
&\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{L}^{(i)}_{r-1,2}}(q,t) \\
&\le \sum_{j=i}^{r-2} \mathrm{Adv}^{\mathrm{kpa}}_{\mathbf{L}^{(j)}_{r-2,1},\mathbf{L}^{(j+1)}_{r-2,1}}(q,t) + \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{L}^{(r-1)}_{r-2,1}}(q,t) && (26)\\
&\le \sum_{j=i}^{r-2} \left(2\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{L}^{(j)}_{r-3,1}}(q,t') + \mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}((\sigma'''+1)q,t') \right) + \frac{q^2}{2^n} && (27)\\
&= \sum_{j=i}^{r-2} 2(r-j-2)\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,t') + (r-i-1)\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}((r-2)q,t') + \frac{q^2}{2^n}, && (28)
\end{aligned}
$$

where inequalities follows from the same analysis we used in derivating Eqs. (17), (18) and (19) and equality results from the trivial fact that $\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{L}^{(j)}_{r-3,1}}(q,t') = (r-j-2)\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,t')$. Combining Eq. (28) and Theorem 3, we obtain

$$
\begin{aligned}
&\mathrm{Adv}^{\mathrm{wprf}}_{\mathrm{FCT}_{r,3}[\mathbf{F}]}(q,t) \\
&\le 2\sum_{i=1}^{r} \Big[(i^2 + (3-2r)i + (r^2-3r+2))\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,t') + (r-i-1)\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(\sigma''q,t') \Big] \\
&+ r\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}((\sigma'+1)q,t') + \frac{rq^2}{2^n} + \frac{rq^2(3(\sigma')^2+1)}{2^n} && (29)\\
&\le \frac{2r}{3}(r^2-6r+11)\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(q,t') + r(r-3)\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(rq,t') \\
&+ r\mathrm{Adv}^{\mathrm{wprf}}_{\mathbf{F}}(r^2q,t') + \frac{rq^2}{2^n} + \frac{rq^2(3(r-1)^4+1)}{2^n}. && (30)
\end{aligned}
$$

Using the above inequality and the assumption that $r \ge 3$, we conclude the proof.

On Linear Systems of Equations with Distinct Variables and Small Block Size

Proof of a Combinatorial Conjecture with Applications to Random Feistel Schemes

Jacques Patarin

Université de Versailles, 45 Avenue des Etats-Unis,
78035 Versailles Cedex, France

Abstract. In this paper we will prove the Conjecture 8.1. of [7]. We call it "Conjecture $P_i \oplus P_j$". It is a purely combinatorial conjecture that has however some cryptographic consequence. For example, from this result we can improve the proven security bounds on random Feistel schemes with 5 rounds: we will prove that no adaptive chosen plaintext/chosen ciphertext attack can exist on 5 rounds Random Feistel Schemes when $m \ll 2^n$. This result reach the optimal bound of security against an adversary with unlimited computing power (but limited by m queries) with the minimum number of rounds. It solves the last case of a famous open problem (cf [8]).

An extended version of this paper is available from the author.

Part I - Introduction, first example

1 Introduction

A "Luby-Rackoff construction with k rounds", which is also known as a "random Feistel cipher" is a Feistel cipher in which the round functions $f_1, \cdots, f_k$ are independently chosen as truly random functions. Since the famous original paper [2] of M. Luby and C. Rackoff, these constructions have inspired a considerable amount of research. In [5] and [7] a summary of existing works on this topic is given. In [2] it was proved that the probability p to distinguish a 3-round random Feistel cipher from a random permutation with an adaptive chosen plaintext attack (CPA-2) is always such that $p \leq \frac{m(m-1)}{2^n}$, where m is the number of queries, and where the permutations are from $2n$ bits $\rightarrow 2n$ bits. So p is negligible when $m \ll 2^{n/2}$. Similarly the probability p to distinguish a 4-round random Feistel cipher from a random permutation with an adaptive chosen plaintext/ciphertext attack (CPCA-2) is always such that $p \leq \frac{m(m-1)}{2^n}$. One way of research is to study security of random Feistel schemes when $m \ll 2^n$ instead of $m \ll 2^{n/2}$. The bound $m \ll 2^{n/2}$ is called the "birthday bound", and it was proved (cf [1], [6]) that this bound is optimal for ≤ 4 rounds. The bound

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 299–321, 2006.
© Springer-Verlag Berlin Heidelberg 2006

$m \ll 2^n$ is the larger bound that we can get against an adversary with unlimited computing power but limited to m queries, since such an adversary can try all the possible round functions. In [1], W. Aiello and R. Venkatesan have found a construction of locally random functions ("Benes") where the optimal bound ($m \ll 2^n$) is obtained instead of the birthday bound. However here the functions are not permutations. In [4], U. Maurer and K. Pietrzak have proved that we can get as close as wanted to $m \ll 2^n$ when the number of rounds tends to infinity. In [7] and [8], J. Patarin has proved that for ≥ 6 rounds we have CPCA-2 security when $m \ll 2^n$. For 5 rounds the CPCA-2 security was still unclear.

In this paper, we will prove the conjecture 8.1 of [7] p. 525, and from this conjecture, we will solve this open problem of 5 rounds: for 5 rounds, as for more rounds, we will show that we have CPCA-2 security when $m \ll 2^n$. We get here the optimal bound of security against an adversary with unlimited computing power (but limited to m queries) with the minimum number of rounds. Moreover, conjecture 8.1 (called "Conjecture $P_i \oplus P_j$" in this paper) is very general and may have many other applications.

2 First Example to Illustrate the "Theorem $P_i \oplus P_j$"

(In section 3, we will define precisely what we call the "Theorem $P_i \oplus P_j$". We will here illustrate a very specific case of this conjecture, where the proof is already not so easy).

- Let $I_n = \{0,1\}^n$, where n is an integer.
- Let $P_1, P_2, \cdots, P_\alpha$ be α pairwise distinct values of I_n, randomly chosen.
- Let $\lambda_0 = P_1 \oplus P_2$, $\lambda_1 = P_3 \oplus P_4$, $\cdots$, $\lambda_{\alpha/2-1} = P_{\alpha-1} \oplus P_\alpha$.

Theorem 1. *If $\alpha \ll 2^n$, then the probability to distinguish $\lambda_0, \lambda_1, \cdots, \lambda_{\alpha/2-1}$ from $\alpha/2 - 1$ random values of I_n (or of $I_n - \{0\}$, it changes nothing essentially) is negligible (we are also interested in finding an explicit majoration of this probability).*

Despite its apparent simplicity, this theorem is not very easy to prove. In this paper, we will prove it, and more generally we will prove a stronger theorem about sets of equations like this, that we call "Theorem $P_i \oplus P_j$". In this "Theorem $P_i \oplus P_j$", we will study more generally some systems where:

1. The equations are more general than 2 by 2 equations on the P_i: we will study cases of at most ξ_{max} values P_i linked in the same block, when ξ_{max} is a constant, or when $\xi_{max}\alpha \ll 2^n$.
2. When the λ_i values are fixed to any values $\neq 0$, we will evaluate the minimum number of pairwise distinct P_i solution of the equations.

3 What Is the "Theorem $P_i \oplus P_j$"

Definition 1. *Let (A) be a set of equations $P_i \oplus P_j = \lambda_k$, with $P_i, P_j, \lambda_k \in I_n$. If by linearity from (A) we cannot generate an equation in only the λ_k, we will*

say that (A) *has "no circle in* P*", or that the equations of* (A) *are "linearly independent in* P*".*

Let a be the number of equations in (A), and α be the number of variables P_i in (A). So we have parameters $\lambda_1, \lambda_2, \cdots, \lambda_a$ and $a+1 \leq \alpha \leq 2a$.

Definition 2. *We will say that two indices* i *and* j *are "in the same block" if by linearity from the equations of* (A) *we can obtain* $P_i \oplus P_j =$ *an expression in* $\lambda_1, \lambda_2, \cdots, \lambda_a$.

Definition 3. *We will denote by* ξ_{max} *the maximum number of indices that are in the same block.*

Example 1. If $A = \{P_1 \oplus P_2 = \lambda_1, P_1 \oplus P_3 = \lambda_2, P_4 \oplus P_5 = \lambda_3\}$, here we have two blocks of indices $\{1,2,3\}$ and $\{4,5\}$, and $\xi_{max} = 3$.

Definition 4. *For such a system* (A)*, when* $\lambda_1, \lambda_2, \cdots, \lambda_a$ *are fixed, we will denote by* h_α *the number of* $P_1, P_2, \cdots, P_\alpha$ *solutions of* (A) *such that:* $\forall i, j$, $i \neq j \Rightarrow P_i \neq P_j$*. We will also denote* $H_\alpha = 2^{na} h_\alpha$.

Remark. h_α and H_α are a concise notations for $h_\alpha(A)$ and $H_\alpha(A)$. For a given value α, h_α and H_α can have different values for different systems A.

Definition 5. *We will denote by* J_α *the number of* $P_1, P_2, \cdots, P_\alpha$ *in* I_n *such that:* $\forall i, j$, $i \neq j \Rightarrow P_i \neq P_j$*. So* $J_\alpha = 2^n \cdot (2^n - 1) \cdots (2^n - \alpha + 1)$.

Theorem 2 ("Theorem $P_i \oplus P_j$" when ξ_{max} is fixed). *Let* ξ_{max} *be a fixed integer,* $\xi_{max} \geq 2$*. Let* (A) *be a set of equations* $P_i \oplus P_j = \lambda_k$ *with no circle in* P*, with* α *variables* P_i*, such that:*

1. *We have no more than* ξ_{max} *indices in the same block.*
2. *The* $\lambda_1, \lambda_2, \cdots, \lambda_a$ *have any fixed values such that: for all* i *and* j *in the same block,* $i \neq j$*, the equation of* $P_i \oplus P_j$ *in* $\lambda_1, \lambda_2, \cdots, \lambda_a$ *is* $\neq 0$ *(i.e. by linearity from* (A) *we cannot generate an equation* $P_i = P_j$ *with* $i \neq j$*).*

Then we have for sufficient large n*:* $H_\alpha \geq J_\alpha$*. (This means: for all fixed* ξ_{max}*,* $\exists n_0 \in \mathbb{N} / \forall n \geq n_0$*, for all system* A *that satisfies 1. and 2., we have:* $H_\alpha(A) \geq J_\alpha$*).*

Remark. This theorem was proved in [7] if we add the condition $\alpha^3 \ll 2^{2n}$ (and also $\xi_{max}\alpha^3 \ll 2^{2n}$ since ξ_{max} is here a fixed integer).

Theorem 3 ("Theorem $P_i \oplus P_j$" when $\xi_{max}\alpha \ll 2^n$). *With the same notations, we have the same result, with the hypothesis* $\xi_{max}\alpha \ll 2^n$ *(instead of* ξ_{max} *a fixed integer).*

Remark. For cryptographic use, or for the problem given in section 2, weaker version of this theorem will be enough. For example, instead of $H_\alpha \geq J_\alpha$ for sufficiently large n, $H_\alpha \geq J_\alpha \left(1 - f(\frac{\xi\alpha}{2^n})\right)$, where f is a function such that $f(x) \to 0$ when $x \to 0$, is enough.

Another variant of this Theorem $P_i \oplus P_j$ is:

Theorem 4 ("Theorem $P_i \oplus P_j$ when $\xi_{max} \leq O(n)$ and $\xi_{average} \leq 3$). *Let $\xi_{average}$ be the average value of ξ, where ξ is the number of variables P_j that are fixed from the equations (A) when we fix a variable P_i. If $\xi_{max} \leq O(n)$ and $\xi_{average} \leq 3$, then for sufficient large n, $H_\alpha \geq J_\alpha$.*

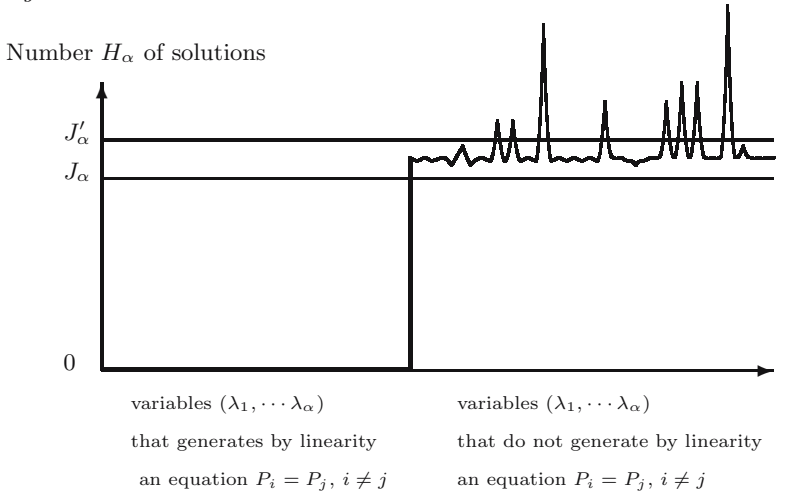


Fig. 1.

Generalizations of the "Theorem $P_i \oplus P_j$". This theorem may have many generalizations. For example:

- Generalization 1: the theorem is still true in any group G (instead of I_n).
- Generalization 2: we have a similar property for equations with 3, 4, $\cdots$, or k variables, i.e. each equation is $P_{i_1} \oplus P_{i_2} \cdots P_{i_k} = \lambda_l$ with pairwise distinct P_i variables.

However in this paper we will only study the original "Theorem $P_i \oplus P_j$" (i.e. theorems 2 and 3) since it is this one that is needed to study random Feistel schemes.

Part II - Analysis with $\xi_{max} = 2$

We will first study the cases where $\xi_{max} = 2$.

4 First Results When $\xi_{max} = 2$ and $\alpha^2 \ll 2^n$ or $\alpha^3 \ll 2^{2n}$

We start with this section since here we will illustrate the general strategy that we will follow to prove the "Theorem $P_i \oplus P_j$". This general strategy is to compute H_α and J_α by induction on α, by adding one more block of ($\leq \xi_{max}$) variables at each time.

We have: $J_\alpha = 2^n(2^n-1)\cdots(2^n-\alpha+1)$.
So $J_{\alpha+2} = (2^n-\alpha)(2^n-\alpha-1)J_\alpha$.
$J_{\alpha+2} = (2^{2n} - 2^n(2\alpha+1) + \alpha(\alpha-1))J_\alpha$. (1)
We also have: $H_{\alpha+2} \geq 2^n(2^n-2\alpha)H_\alpha$. (2)

Remark. Here $H_{\alpha+2}$ and H_α are concise notations for $H_{\alpha+2}(A')$ and $H_\alpha(A)$, where (A') is any system of equations obtained from (A) by addition of one more block: $P_{\alpha+1} \oplus P_{\alpha+2} = \lambda_{\alpha/2}$.

Proof of (2). When $P_1, \cdots, P_\alpha$ are fixed pairwise distinct, we look for solutions $P_{\alpha+1}, P_{\alpha+2}$ such that: $P_{\alpha+1} \oplus P_{\alpha+2} = \lambda_{\alpha/2}$, and such that $P_1, \cdots, P_\alpha$, $P_{\alpha+1}$, $P_{\alpha+2}$ are pairwise distinct. So $P_{\alpha+2}$ is fixed when $P_{\alpha+1}$ is fixed, and we want $P_{\alpha+1} \notin \{P_1, \cdots, P_\alpha, \lambda_{\alpha/2} \oplus P_1, \cdots, \lambda_{\alpha/2} \oplus P_\alpha\}$. So for $(P_{\alpha+1}, P_{\alpha+2})$ we have between $2^n - 2\alpha$ and $2^n - \alpha$ solutions when $P_1, \cdots, P_\alpha$ are fixed.

Now from (1) and (2) we have:

$$\frac{H_{\alpha+2}}{J_{\alpha+2}} \geq \frac{2^{2n} - 2\alpha \cdot 2^n}{2^{2n} - 2^n(2\alpha+1) + \alpha(\alpha+1)} \frac{H_\alpha}{J_\alpha} = \left(1 + \frac{2^n - \alpha(\alpha+1)}{2^{2n} - 2^n(2\alpha+1) + \alpha(\alpha+1)}\right) \frac{H_\alpha}{J_\alpha}$$

We have also $H_2 > J_2$ since $H_2 = 2^{2n} > J_2 = 2^n(2^n-1)$. So if $\alpha^2 \ll 2^n$, we have $H_\alpha \geq J_\alpha$ by induction on α. (3)
Moreover,

$$\frac{H_{\alpha+2}}{J_{\alpha+2}} \geq \left(1 + \frac{-\alpha(\alpha+1)}{2^{2n} - O(\alpha 2^n)}\right) \frac{H_\alpha}{J_\alpha}$$

So we have:

$$\frac{H_{\alpha+2}}{J_{\alpha+2}} \geq \left(1 - \frac{\alpha(\alpha+1)}{2^{2n} - O(\alpha 2^n)}\right)^{\alpha/2} \frac{H_2}{J_2}$$

So:

$$\frac{H_{\alpha+2}}{J_{\alpha+2}} \geq 1 - O\left(\frac{\alpha^3}{2^{2n}}\right) \text{ and so } H_\alpha \geq J_\alpha \left(1 - O\left(\frac{\alpha^3}{2^{2n}}\right)\right)$$

So if $\alpha^3 \ll 2^{2n}$, $H_\alpha \geq J_\alpha(1-\varepsilon)$, where ε is very small. (4)

Now to extend the result (3) with the condition $\alpha \ll 2^n$ instead of $\alpha^2 \ll 2^n$, or to extend the result (4) with the condition $\alpha \ll 2^n$ instead of $\alpha^3 \ll 2^{2n}$, we will improve the evaluation (2) of $H_{\alpha+2}$ from H_α.

5 General Properties When $\xi_{max} = 2$

Here, since $\xi_{max} = 2$, our set of equations is:

$$(A) \begin{cases} P_2 = P_1 \oplus \lambda_0 \\ P_4 = P_3 \oplus \lambda_1 \\ \vdots \\ P_\alpha = P_{\alpha-1} \oplus \lambda_{\alpha/2-1} \end{cases}$$

h_α is by definition the number of $P_1, \cdots, P_\alpha$ pairwise distinct, elements of I_n, and solution of (A). We want to evaluate h_α by induction on α, i.e. we want to evaluate $h_{\alpha+2}$ from h_α. We will say that $(P_1, \cdots, P_\alpha)$ are solution of h_α, when they are solution of (A). We will denote $\beta = \alpha/2 - 1$. For $h_{\alpha+2}$ we have (A) and one more equation: $P_{\alpha+1} \oplus P_{\alpha+2} = \lambda_{\beta+1}$ (see figure 2)

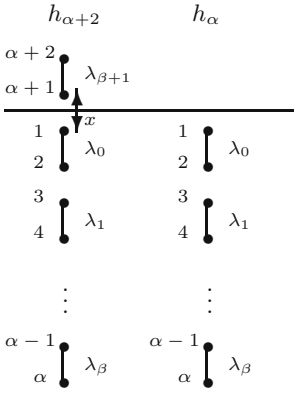


Fig. 2. We want to evaluate $h_{\alpha+2}$ from h_α

Remark. We start from a solution $P_1, \cdots, P_\alpha$ of h_α and we want to complete it to get the solutions of $h_{\alpha+2}$. For this we have to choose $x = P_{\alpha+1} \oplus P_1$ such that x will not create a collision $P_j = P_{\alpha+1}$ or $P_j = P_{\alpha+2}$, $1 \leq j \leq \alpha$. This means: $x \oplus P_1 \neq P_j$ and $\lambda_{\beta+1} \oplus x \oplus P_1 \neq P_j$, $\forall j$, $1 \leq j \leq \alpha$. So this means $x \notin V$ with $V = V_1 \cup V_2$, with $V_1 = \{P_1 \oplus P_j, 1 \leq j \leq \alpha\}$ and $V_2 = \{\lambda_{\beta+1} \oplus P_1 \oplus P_j, 1 \leq j \leq \alpha\}$. We have $|V| = |V_1 \cup V_2| = |V_1| + |V_2| - |V_1 \cap V_2|$, and we have $|V_1| = \alpha$ and $|V_2| = \alpha$ (since the P_j values, $1 \leq j \leq \alpha$, are pairwise distinct). So

$$h_{\alpha+2} = \sum_{(P_1,\cdots,P_\alpha) \text{ solution of } h_\alpha} (2^n - |V|)$$

$$h_{\alpha+2} = \sum_{(P_1,\cdots,P_\alpha) \text{ solution of } h_\alpha} (2^n - 2\alpha + |V_1 \cap V_2|)$$

$$h_{\alpha+2} = (2^n - 2\alpha) h_\alpha \sum_{1 \leq i \leq \alpha, 1 \leq j \leq \alpha} \text{Number of } P_1, \cdots, P_\alpha \text{ solution of } h_\alpha$$

$$\text{plus the equation } \lambda_{\beta+1} = P_i \oplus P_j$$

Now when we add the equality $\lambda_{\beta+1} = P_i \oplus P_j$ to the system of equation (A) of h_α, 3 cases can occur:

Case 1. $\lambda_{\beta+1} = P_i \oplus P_j$ was already an equation of (A). Here this means $\lambda_{\beta+1} = \lambda_i$ for a value i, $1 \leq i \leq \alpha$. Remark: $\lambda_{\beta+1} = \lambda_i$ creates 2 collisions in $V_1 \cap V_2$: it creates $\lambda_{\beta+1} \oplus P_1 \oplus P_i = P_1 \oplus P_j$ and $\lambda_{\beta+1} \oplus P_1 \oplus P_j = P_1 \oplus P_i$.

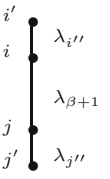


Fig. 3. Here $\lambda_{\beta+1} = \lambda_{i''}$, $\lambda_{\beta+1} = \lambda_{j''}$ and $\lambda_{\beta+1} = \lambda_{i''} \oplus \lambda_{j''}$ are impossible if the P_j are pairwise distinct

Case 2. $\lambda_{\beta+1} = P_i \oplus P_j$ is in contradiction with the equations of (A). This can come from the fact that $P_i \oplus P_j = \lambda_k$ is in (A) and $\lambda_{\beta+1} \neq \lambda_k$. Or this can come from the fact that $P_i \oplus P_{i'} = \lambda_{i''}$ is in (A) and $P_j \oplus P_{j'} = \lambda_{j''}$ is in (A), so from $P_i \oplus P_j = \lambda_{\beta+1}$ we get $P_{j'} = \lambda_{j''} \oplus \lambda_{\beta+1} \oplus P_i$, $P_{i'} = \lambda_{i''} \oplus \lambda_{\beta+1} \oplus P_j$, and $P_{i'} \oplus P_{j'} = \lambda_{i''} \oplus \lambda_{j''} \oplus \lambda_{\beta+1}$. This is impossible if $\lambda_{\beta+1} = \lambda_{j''}$, $\lambda_{\beta+1} = \lambda_{i''}$ or $\lambda_{\beta+1} = \lambda_{i''} \oplus \lambda_{j''}$, since the P_k values are pairwise distinct (see figure 3).

Case 3. The equation $\lambda_{\beta+1} = P_i \oplus P_j$ is not in contradiction with the equations of (A), and is not a consequence of the equations of (A). We will say that this case is the "generic" case, and we will denote by h'_α the number of $P_1, \cdots, P_\alpha$ solution of (A) and $\lambda_{\beta+1} = P_i \oplus P_j$ when we are in such "generic" case.

Remark. The value of h'_α is dependent on the λ_i values. However we will see in section 6 that all the values h'_α are very near.

From (1) and from the 3 cases above we get immediately:

Theorem 5.

$$\frac{h_{\alpha+2}}{2^n} = h_\alpha \left[1 - \frac{2\alpha}{2^n} + \frac{2 \text{ Number of equations } \lambda_{\beta+1} = \lambda_i}{2^n}\right] + \sum_{(i,j)\in M} (\text{Number of } P_1, \cdots, P_\alpha \text{ solution of } h_\alpha \text{ plus the equation } \lambda_{\beta+1} = P_i \oplus P_j)$$

where $M = \{(i,j), 1 \leq i < j \leq \alpha$, *such that if* i *is odd, then* $j \neq i+1$, *and such that* $\lambda_{\beta+1} \neq \lambda_i$, $\lambda_{\beta+1} \neq \lambda_j$ *and* $\lambda_{\beta+1} \neq \lambda_i \oplus \lambda_j\}$

We have $|M| = \alpha(\alpha-2)-$ Number of $(i,j), 1 \leq i \leq \alpha, 1 \leq j \leq \alpha / \lambda_{\beta+1} = \lambda_i$ or λ_j or $\lambda_i \oplus \lambda_j$. If $(i,j) \in M$, we are in case 3, the "generic" case, and the number of $P_1, \cdots, P_\alpha$ solution of h_α plus the equation $\lambda_{\beta+1} = P_i \oplus P_j$ is denoted by h'_α. We will now (in section 6) compare h'_α and h_α in order to get from theorem 5 a relation between $h_{\alpha+2}$ and h_α.

6 Relations Between h'_α and h_α When $\xi_{max} = 2$

The aim of this section is to prove that $h'_\alpha \geq h_\alpha(1 - O(\frac{\alpha}{2^{2n}}))$ (we are here in the generic case 3, as seen is section 5). Notice that here we look for an evaluation in $\frac{1}{2^{2n}}$, not only in $\frac{1}{2^n}$ (because $|M| = O(\alpha^2)$, where M is the set of theorem 5). For this, the idea is to evaluate h_α and h'_α from $h_{\alpha-4}$ (see figure 4).

Notations. We have $P_1 \oplus P_2 = \lambda_1$, $P_3 \oplus P_4 = \lambda_3$. For h'_α, we will denote $\lambda' = P_2 \oplus P_4$ and $x = P_4 \oplus P_5$ (see figure 4).

Theorem 6. *We have:*

$$h'_\alpha = \sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} (2^n - (|W_1| + |W_2|) + |W_1 \cap W_2|) \qquad (1)$$

and

$$h_\alpha \leq \sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} (2^n - |W_1|)(2^n - |W_2|) \qquad (2)$$

with $W_1 = \{\lambda_1 \oplus \lambda' \oplus P_j, \lambda' \oplus P_j, 5 \leq j \leq \alpha\}$
and $W_2 = \{\lambda_3 \oplus P_j, P_j, 5 \leq j \leq \alpha\}$.

Proof of (1). We start from a solution $P_5, \cdots, P_\alpha$ of $h_{\alpha-4}$ and we want to complete it to get the solutions of h'_α. For this we have to choose $x = P_4 \oplus P_5$ such that x will not create a collision $P_j = P_1$ or $P_j = P_2$, $5 \leq j \leq \alpha$ (#) and x will not create a collision $P_j = P_3$ or $P_j = P_4$, $5 \leq j \leq \alpha$ (##).

We have: $P_1 = \lambda_1 \oplus \lambda' \oplus x \oplus P_5$, and $P_2 = \lambda' \oplus x \oplus P_5$. So (#) means $x \notin W'_1$ where $W'_1 = \{P_5 \oplus \lambda_1 \oplus \lambda' \oplus P_j, P_5 \oplus \lambda' \oplus P_j, 5 \leq j \leq \alpha\}$. Similarly, we have: $P_3 = \lambda_3 \oplus x \oplus P_5$, and $P_4 = x \oplus P_5$. So (##) means $x \notin W'_2$ where $W'_2 = \{P_5 \oplus \lambda_3 \oplus P_j, P_5 \oplus P_j, 5 \leq j \leq \alpha\}$. So

$$h'_\alpha = \sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} (2^n - |W'_1 \cup W'_2|).$$

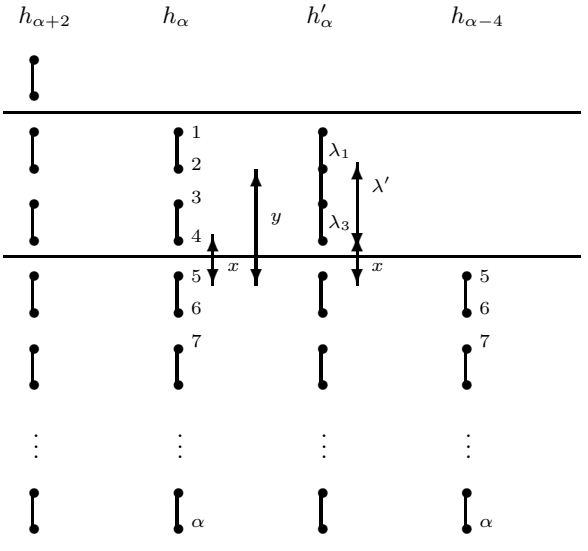


Fig. 4. We want to compare h_α and h'_α. This figure illustrates that we will do this by evaluating h_α from $h_{\alpha-4}$ and h'_α from $h_{\alpha-4}$.

Now W_1 and W_2 are just the translation of W_1' and W_2' by P_5, so $|W_1' \cup W_2'| = |W_1 \cup W_2| = |W_1| + |W_2| + |W_1 \cap W_2|$, so we have (1) as claimed.

Proof of (2). We start from a solution $P_5, \cdots, P_\alpha$ of $h_{\alpha-4}$ and we want to complete it to get the solutions of h_α. For this we have to choose $x = P_4 \oplus P_5$ and $y = P_2 \oplus P_5$ such that x will not create a collision $P_j = P_3$ or $P_j = P_4$, $5 \le j \le \alpha$ (3), y will not create a collision $P_j = P_1$ or $P_j = P_2$, $5 \le j \le \alpha$ (4), and x and y will not create a collision $P_1 = P_3$, $P_1 = P_4$, $P_2 = P_3$ or $P_2 = P_4$ (5). So

$$h_\alpha \le \sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} (2^n - \text{Number of } x \text{ that create (3)}) \cdot (2^n - \text{Number of } y \text{ that create (4)}).$$

(We have here $\le$ and not $=$ since we removed condition (5)). So

$$h_\alpha \le \sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} (2^n - |W_1|)(2^n - |W_2|)$$

as claimed.

In order to compare h'_α and h_α, we see from theorem 6 that we want to evaluate

$$\chi = \sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} \left(\frac{|W_1 \cap W_2|}{2^n} - \frac{|W_1||W_2|}{2^{2n}} \right).$$

Let $A_1 = \{P_j \oplus \lambda', 5 \le j \le \alpha\}$
$A_2 = \{P_j \oplus \lambda_1 \oplus \lambda', 5 \le j \le \alpha\}$
$A_1' = \{P_j, 5 \le j \le \alpha\}$
$A_2' = \{P_j \oplus \lambda_3, 5 \le j \le \alpha\}$.

We have $W_1 = A_1 \cup A_2$, and $W_2 = A_1' \cup A_2'$. We have $|A_1| = |A_2| = |A_1'| = |A_2'| = \alpha - 4$ (because the P_j solutions of $h_{\alpha-4}$ are pairwise distinct). We have $W_1 \cap W_2 = (A_1 \cup A_2) \cap (A_1' \cup A_2')$.

Theorem 7. *We have:*

$$\begin{aligned} |W_1 \cap W_2| &= |(A_1 \cup A_2) \cap (A_1' \cup A_2')| \\ &= |A_1 \cap A_1'| + |A_1 \cap A_2'| + |A_2 \cap A_1'| + |A_2 \cap A_2'| \\ &\quad -|A_1 \cap A_1' \cap A_2'| - |A_2 \cap A_1' \cap A_2'| - |A_1 \cap A_2 \cap A_1'| - |A_1 \cap A_2 \cap A_2'| \\ &\quad +|A_1 \cap A_2 \cap A_1' \cap A_2'|. \end{aligned}$$

Proof. This is a classical result on sets.

Theorem 8. *We have:*

$$\begin{aligned} |W_1||W_2| &= |A_1||A_1'| + |A_1||A_2'| + |A_2||A_1'| + |A_2||A_2'| \\ &\quad -|A_1 \cap A_2||A_1'| - |A_1 \cap A_2||A_2'| - |A_1||A_1' \cap A_2'| - |A_2||A_1' \cap A_2'| \\ &\quad +|A_1 \cap A_2||A_1' \cap A_2'|. \end{aligned}$$

Proof. $|W_1||W_2| = |A_1 \cup A_2||A'_1 \cup A'_2| = (|A_1| + |A_2| - |A_1 \cap A_2|)(|A'_1| + |A'_2| - |A'_1 \cap A'_2|)$ and by developing this expression, we get theorem 8.

Theorem 9. $\chi = O\left(\frac{\alpha}{2^{2n}}\right) h_{\alpha-4}$

Proof. We have:

$\sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} |A_1 \cap A_2| = \frac{\alpha^2+O(\alpha)}{2^n} h_{\alpha-4}$

$\sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} |A'_1 \cap A'_2| = \frac{\alpha^2+O(\alpha)}{2^n} h_{\alpha-4}$

$\sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} |A_1 \cap A'_1| = \frac{\alpha^2+O(\alpha)}{2^n} h_{\alpha-4}$

$\sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} |A_1 \cap A'_2| = \frac{\alpha^2+O(\alpha)}{2^n} h_{\alpha-4}$

$\sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} |A_2 \cap A'_1| = \frac{\alpha^2+O(\alpha)}{2^n} h_{\alpha-4}$

$\sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} |A_2 \cap A'_2| = \frac{\alpha^2+O(\alpha)}{2^n} h_{\alpha-4}$

$\sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} |A_1 \cap A'_1 \cap A'_2| = \frac{\alpha^3+O(\alpha^2)}{2^{2n}} h_{\alpha-4}$

$\sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} |A_2 \cap A'_1 \cap A'_2| = \frac{\alpha^3+O(\alpha^2)}{2^{2n}} h_{\alpha-4}$

$\sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} |A_1 \cap A_2 \cap A'_1| = \frac{\alpha^3+O(\alpha^2)}{2^{2n}} h_{\alpha-4}$

$\sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} |A_1 \cap A_2 \cap A'_2| = \frac{\alpha^3+O(\alpha^2)}{2^{2n}} h_{\alpha-4}$

$\sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} |A_1 \cap A_2 \cap A'_1 \cap A'_2| = \frac{\alpha^4+O(\alpha^3)}{2^{3n}} h_{\alpha-4}$

So from theorem 7 and theorem 8 and from the definition of χ, we get:

$$\chi = \frac{h_{\alpha-4}}{2^n}\left[\frac{4\alpha^2+O(\alpha)}{2^n} - \frac{4\alpha^3+O(\alpha^2)}{2^{2n}} + \frac{\alpha^4+O(\alpha^3)}{2^{3n}}\right]$$

$$-\frac{h_{\alpha-4}}{2^{2n}}\left[4(\alpha-4)^2 - 4\left(\frac{\alpha^2+O(\alpha)}{2^n}\right)(\alpha-4) + \left(\frac{\alpha^2+O(\alpha)}{2^n}\right)\left(\frac{\alpha^2+O(\alpha)}{2^n}\right)\right]$$

$$\chi = \left(\frac{O(\alpha)}{2^{2n}} + \frac{O(\alpha^2)}{2^{3n}} + \frac{O(\alpha^3)}{2^{4n}}\right) h_{\alpha-4}$$

So $\chi = O\left(\frac{\alpha}{2^{2n}}\right) h_{\alpha-4}$ as claimed.

Theorem 10. $h'_\alpha \geq \frac{h_\alpha}{2^n}\left[1 - O\left(\frac{\alpha}{2^{2n}}\right)\right]$.

Proof. We have:

$$h_\alpha \leq \sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} (2^n - |W_1|)(2^n - |W_2|)$$

and

$$h'_\alpha = 2^n \sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} \left(1 - \frac{|W_1| + |W_2|}{2^n} + \frac{|W_1 \cap W_2|}{2^n}\right).$$

So

$$h'_\alpha \geq 2^n \sum_{(P_5,\cdots,P_\alpha) \text{ solution of } h_{\alpha-4}} \left(1 - \frac{|W_1| + |W_2|}{2^n} + \frac{|W_1||W_2|}{2^n}\right) + 2^n\chi$$

So

$$h'_\alpha \geq \frac{h_\alpha}{2^n} + 2^n h_{\alpha-4} O\left(\frac{\alpha}{2^{2n}}\right).$$

Moreover, $h_{\alpha-4} = \frac{h_\alpha}{2^n}(1 + O(\alpha))$, so

$$h'_\alpha \geq \frac{h_\alpha}{2^n} + \frac{h_\alpha}{2^n} O\left(\frac{\alpha}{2^{2n}}\right)(1 + O(\alpha))$$

So $h'_\alpha \geq \frac{h_\alpha}{2^n}\left(1 - O\left(\frac{\alpha}{2^{2n}}\right)\right)$ as claimed.

7 Number of Indices $i_1, \cdots, i_k / \lambda_{i_1} \oplus \cdots \oplus \lambda_{i_k} = 0$

Let $\lambda_1, \cdots, \lambda_\alpha$ be α elements of I_n.
Let N_1 be the number of (i, j), $1 \leq i < j \leq \alpha$, such that $\lambda_i = \lambda_j$.
Let N_2 be the number of (i, j, k), $1 \leq i \leq \alpha$, $1 \leq j \leq \alpha$, $1 \leq k \leq \alpha$ such that $\lambda_i \oplus \lambda_j \oplus \lambda_k = 0$.

More generally, for all integer $l \geq 2$, let N_l be the number of $(i_1, \cdots, i_{l+1})$, $1 \leq i_1 \leq \alpha$, $1 \leq i_2 \leq \alpha$, $\cdots$, $1 \leq i_{l+1} \leq \alpha$ such that $\lambda_{i_1} \oplus \cdots \oplus \lambda_{i_{l+1}} = 0$.

In this section we will prove some results on the N_k values. (These results will be useful when we will compare H with special values λ_i from H with random values λ_i).

Theorem 11. *We always have:* $N_2 \leq 2\alpha N_1 + \alpha^2$.

Remark

1. If the λ_i values are pairwise distinct, we have $N_1 = 0$ and $N_2 \leq \alpha^2$ (since when i and j are fixed there is here at most one k such that $\lambda_i \oplus \lambda_j = \lambda_k$).
2. If $\forall i$, $1 \leq i \leq \alpha$, $\lambda_i = 0$, we have: $N_1 = \frac{\alpha(\alpha-1)}{2} \simeq \frac{\alpha^2}{2}$, and $N_2 = \alpha^3$.
3. It is easy to prove that $N_2 \leq \alpha^2(N_1 + 1)$ (since when i and j are fixed, and we have at most α^2 possibilities for i and j, for k such that $\lambda_i \oplus \lambda_j = \lambda_k$ we have at most $1 + N_1$ possibilities). However we will prove that we have $N_2 \leq 2\alpha N_1 + \alpha^2$, not only $N_2 \leq \alpha^2(N_1 + 1)$.

Proof of theorem 11. For all $\beta \in I_n$, let n_β be the number of indices i, $1 \leq i \leq \alpha$, such that $\lambda_i = \beta$. We have:

$$\sum_{\beta \in I_n} n_\beta = \alpha \tag{1}$$

$$\sum_{\beta \in I_n} \frac{n_\beta(n_\beta - 1)}{2} = N_1 \tag{2}$$

So

$$N_1 = \sum_{\beta \in I_n} \frac{n_\beta^2}{2} - \frac{\alpha}{2} \tag{3}$$

We also have:

$$N_2 = \text{Number of } (i,j,k)/\lambda_i \oplus \lambda_j \oplus \lambda_k = 0$$

$$N_2 = \sum_{i=1}^{\alpha} \left[\sum_{\beta \in I_n} (\text{Number of } j/\lambda_i \oplus \lambda_j = \beta) \cdot (\text{Number of } k/\lambda_k = \beta) \right]$$

$$N_2 = \sum_{i=1}^{\alpha} \left[\sum_{\beta \in I_n} n_{\beta \oplus \lambda_i} \cdot n_\beta \right]$$

The function $\varphi : I_n \to I_n, x \mapsto x \oplus \lambda_i$ is a bijection. So here we have

$$N_2 = \sum_{i=1}^{\alpha} \left[\sum_{\beta \in I_n} n_{\varphi(\beta)} \cdot n_\beta \right]$$

where φ is a bijection (4). If

```
$\vec{v}\left(\icvec[\gamma]{l}\right)$ and
$\vec{v'}\left(\begin{array}{c} l_{\varphi(1)}\\ \vdots\\
l_{\varphi(\gamma)}
\end{array}\right)$
```

where φ is a bijection, we have (Cauchy-Schwartz theorem):

$||\boldsymbol{v} \cdot \boldsymbol{v'}|| \leq ||\boldsymbol{v}|| \cdot ||\boldsymbol{v'}||$, but here $||\boldsymbol{v}|| = ||\boldsymbol{v'}||$, so $||\boldsymbol{v} \cdot \boldsymbol{v'}|| \leq ||\boldsymbol{v}||^2$. So from (4) we have:

$$N_2 \leq \sum_{i=1}^{\alpha} \sum_{\beta \in I_n} n_\beta^2.$$

So from (3) we have:

$$N_2 \leq \sum_{i=1}^{\alpha} [2N_1 + \alpha].$$

So $N_2 \leq 2\alpha N_1 + \alpha^2$, as claimed.

More generally we have:

Theorem 12. *For all integer $l \geq 2$, we have: $N_l \leq 2\alpha^{l-1} N_1 + \alpha^l$.*

Proof.

$$N_l = \sum_{i_1=1}^{\alpha} \sum_{i_2=1}^{\alpha} \cdots \sum_{i_{l-1}=1}^{\alpha} \Big(\sum_{\beta \in I_n} [\text{Number of } i_l/\lambda_{i_1} \oplus \ldots \oplus \lambda_{i_l} = \beta]$$

$$\cdot [\text{Number of } i_{l+l}/\lambda_{i_{l+1}} = \beta] \Big)$$

So

$$N_l = \sum_{i_1=1}^{\alpha} \sum_{i_2=1}^{\alpha} \cdots \sum_{i_{l-1}=1}^{\alpha} \left(\sum_{\beta \in I_n} n_{\beta \oplus \lambda_{i_1} \oplus \ldots \oplus \lambda_{i_{l-1}}} \cdot n_\beta \right) \quad (1)$$

The function $\varphi : I_n \to I_n, x \mapsto x \oplus \lambda_{i_1} \oplus \ldots \oplus \lambda_{i_{l-1}}$ is a bijection. So from Cauchy-Schwartz theorem we have:

$$N_l \leq \alpha^{l-1} \left(\sum_{\beta \in I_n} n_\beta^2 \right)$$

so $N_l \leq 2\alpha^{l-1} N_1 + \alpha^l$ as claimed.

8 Conclusion for $\xi_{max} = 2$

From section 7 and theorem 5 we get that the minimal values for H_α are obtained when the λ_i are pairwise distinct. Then for such values from theorem 5, we get

$$\frac{h_{\alpha+2}}{2^n} \geq h_\alpha \left(1 - \frac{2\alpha}{2^n}\right) + (\alpha(\alpha-2) - O(\alpha)) h'_\alpha$$

$$\geq h_\alpha \left(1 - \frac{2\alpha}{2^n} + \frac{\alpha^2 - O(\alpha)}{2^{2n}}\right)$$

So

$$H_{\alpha+2} \geq (2^{2n} - 2^n(2\alpha) + (\alpha^2 - O(\alpha))) H_\alpha \quad (1)$$

and

$$J_{\alpha+2} = (2^{2n} - 2^n(2\alpha+1) + (\alpha(\alpha-1))) J_\alpha \quad (2)$$

From (1) and (2) and $H_2 > J_2$, we get by induction on α that for sufficiently large n, $H_\alpha \geq J_\alpha$, as claimed: $\frac{H_{\alpha+2}}{J_{\alpha+2}} = \frac{1 + \frac{-2\alpha}{2^n} + \frac{\alpha^2 - O(\alpha)}{2^{2n}}}{1 + \frac{-2\alpha-1}{2^n} + \frac{\alpha(\alpha-1)}{2^{2n}}} \frac{H_\alpha}{J_\alpha}$ so as long as $\alpha \ll 2^n$ we have $\frac{H_{\alpha+2}}{J_{\alpha+2}} \geq \frac{H_\alpha}{J_\alpha} \geq 1$ by induction on α (since $1 + \frac{-2\alpha}{2^n} + \frac{\alpha^2 - O(\alpha)}{2^{2n}} > 1 + \frac{-2\alpha-1}{2^n} + \frac{\alpha(\alpha-1)}{2^{2n}}$ when $2^n > O(\alpha) - \alpha$ and this holds when $\alpha \ll 2^n$).

Part III - Analysis for any ξ_{max}

We can proceed for any ξ_{max} in a similar way as we did for $\xi_{max} = 2$. This is done in appendix B

9 Conclusion

In this paper we have proved the conjecture 8.1 of [7]. As already mentioned in [7], from this conjecture we can prove that there does not exist any adaptive chosen plaintex/ciphertext attack against a random Feistel scheme of $2n$ bits

$\rightarrow 2n$ bits with ≥ 5 rounds when the number m of queries is small compared with 2^n. This number of rounds, 5, is minimal since chosen plaintext attacks with $m \simeq \sqrt{2^n}$ are known against 4-round random Feistel schemes (see [1] or [6]). This result on 5-round random Feistel schemes solves an open problem of [6] and [1].

References

1. W. Aiello and R. Venkatesan, *Foiling Birthday Attacks in Lenght-Doubling Transformations - Benes: a non-reversible alternative to Feistel. EUROCRYPT '96*, LNCS nb 1070, pp. 307–320, Springer.
2. M. Luby and C. Rackoff. *How to construct pseudorandom permutations from pseudorandom functions. SIAM Journal on Computing*, vol. 17, n2, pp. 373–386, April 1988.
3. U. Maurer. *A simplified and generalized treatment of Luby-Rackoff pseudorandom permutation generators. EUROCRYPT '92*, LNCS nb 658, pp. 239–255, Springer-Verlag.
4. U. Maurer and K. Pietrzak. *The security of Many-Round Luby-Rackoff Pseudo-Random Permutations. EUROCRYPT '03*, LNCS nb 2656, pp. 544–561, Springer.
5. M. Naor and O. Reingold, *On the construction of pseudo-random permutations: Luby-Rackoff revisited*, Journal of Cryptology, vol. 12, 1999, pp. 29–66. Extended abstract was published in Proc. 29th Ann. ACM Symp. on Theory of Computing, 1997, pp. 189–199.
6. J. Patarin. *New results on pseudorandom permutation generators based on the DES scheme. CRYPTO '91*, LNCS nb 576, pp. 301–312, Springer-Verlag.
7. J. Patarin. *Luby-Rackoff: 7 Rounds are Enough for $2^{n(1-\epsilon)}$ Security. CRYPTO '03*, LNCS nb 2729, pp.513–529, Springer.
8. J. Patarin, *Security of Random Feistel Scemes with 5 or more rounds. CRYPTO '04*, LNCS nb 3152, pp. 106–122, Springer.

A Appendix: An Example with an Unusual Value for H

The Theorem $P_i \oplus P_j$ says that when $H_{(A)} \neq 0$, when ξ_{max} is fixed, and when $\alpha \ll 2^n$, then $H_{(A)}$ is always $\geq J_\alpha$ (where J_α can be seen as the average value of $H_{(A)}$ over all the λ_i). Moreover, sometimes $H_{(A)}$ is much larger than this average value J_α, even when $\xi_{max} = 2$, as we will see in this example.
Let λ be an element of $I_n, \lambda \neq 0$. We want to compute the number $h_{(A)}$ of pairwise distinct $P_1, P_2, \ldots, P_\alpha$ such that: $P_2 = P_1 \oplus \lambda$, $P_4 = P_3 \oplus \lambda$, $P_6 = P_5 \oplus \lambda, \cdots$, $P_\alpha = P_{\alpha-1} \oplus \lambda$ (for this system we have $\xi_{max} = 2$ and all the λ_i are equal). Let $x_1 = P_3 \oplus P_1, x_2 = P_5 \oplus P_1, \ldots, x_{\alpha/2-1} = P_{\alpha-1} \oplus P_1$.
Since we have 2^n possibilities for P_1, here $h_{(A)}$ is exactly 2^n times the number of $x_1, \ldots, x_{\alpha/2-1}$ such that:

$$
\begin{aligned}
& x_1 \notin \{0, \lambda\} \\
& x_2 \notin \{0, \lambda, x_1, x_1 \oplus \lambda\} \\
& x_3 \notin \{0, \lambda, x_1, x_1 \oplus \lambda, x_2, x_2 \oplus \lambda\} \\
& \cdots \\
& x_{\alpha/2-1} \notin \{0, \lambda, x_1, x_1 \oplus \lambda, x_2, x_2 \oplus \lambda, \cdots, x_{\alpha/2-2}, x_{\alpha/2-2} \oplus \lambda\}
\end{aligned}
$$

For x_1 we have exactly $2^n - 2$ possibilities. Then, when x_1 is fixed, we have exactly $2^n - 4$ solutions for x_2, then if (x_1, x_2) are fixed, we have exactly $2^n - 6$ solutions for x_3 etc. Thus, since we have $\alpha/2$ equations, here $H_{(A)} = 2^{\alpha/2n} h_{(A)}$ (because we have $\alpha/2$ equations), so here we have:

$$H_{(A)} = 2^{(\alpha/2)n} 2^n (2^n - 2)(2^n - 4)(2^n - 6) \cdots (2^n - \alpha + 2)$$

$$J_{(A)} = 2^n (2^n - 1)(2^n - 2)(2^n - 3) \cdots (2^n - \alpha + 1)$$

$$\frac{J_{(A)}}{H_{(A)}} = \left(1 - \frac{1}{2^n}\right)\left(1 - \frac{3}{2^n}\right)\left(1 - \frac{5}{2^n}\right) \cdots \left(1 - \frac{\alpha - 1}{2^n}\right)$$

When $\alpha \gg 2^{n/2}$ (but still $\alpha \ll 2^n$) this expression can be arbitrarily small. So $H_{(A)}$ can be much larger the J_α (even when $\xi_{max} = 2$ and $\alpha \ll 2^n$).

B Appendix: Analysis for Any ξ_{max}

B.1 First Results for Any ξ_{max} When $\alpha^2 \ll 2^n$, or $\alpha^3 \ll 2^{2n}$

Our general strategy is to compare H_α and J_α by induction on α, by adding one more block of ξ variables at each time ($\xi \leq \xi_{max}$). We have $J_\alpha = 2^n(2^n - 1) \cdots (2^n - \alpha + 1)$.

So $J_{\alpha+\xi} = (2^n - \alpha)(2^n - \alpha - 1) \cdots (2^n - \alpha - \xi + 1) J_\alpha$.

So $J_{\alpha+\xi} = (2^{\xi n} + 2^{(\xi-1)n}(-\xi\alpha - \frac{\xi(\xi-1)}{2}) + 2^{(\xi-2)n}(\frac{\xi(\xi-1)}{2}\alpha^2 + O(\alpha)) + O(2^{(\xi-3)n}\alpha^3))J_\alpha$ (1)

We also have: $H_{\alpha+\xi} \geq 2^{(\xi-1)n}(2^n - \xi\alpha)H_\alpha$ (2)

Proof of (2). When $P_1, \cdots, P_\alpha$ are fixed pairwise distinct, we look for solutions $P_{\alpha+1}, \cdots, P_{\alpha+\xi}$ such that: $P_{\alpha+1} \oplus P_{\alpha+2} = \lambda_{\alpha/2}$, $P_{\alpha+1} \oplus P_{\alpha+3} = \lambda_{\alpha/2+1}$, $\cdots$, $P_{\alpha+1} \oplus P_{\alpha+\xi} = \lambda_{\alpha/2+(\xi-2)}$. So $P_{\alpha+2}, P_{\alpha+3}, \cdots, P_{\alpha+\xi}$ are fixed when $P_{\alpha+1}$ is fixed, and we want

$P_{\alpha+1} \notin \{P_1, \cdots, P_\alpha,$
$\lambda_{\alpha/2} \oplus P_1, \cdots, \lambda_{\alpha/2} \oplus P_\alpha,$
$\cdots,$
$\lambda_{\alpha/2+(\xi-2)} \oplus P_1, \cdots, \lambda_{\alpha/2+(\xi-2)} \oplus P_\alpha\}.$

So for $(P_{\alpha+1}, P_{\alpha+2}, \cdots, P_{\alpha+\xi})$ we have between $2^n - \xi\alpha$ and $2^n - \alpha$ solutions when $P_1, \cdots, P_\alpha$ are fixed. Now from (1) and (2) we have:

$$\frac{H_{\alpha+\xi}}{J_{\alpha+\xi}} \geq \left[1 + \frac{2^{(\xi-1)n}\left(\frac{\xi(\xi-1)}{2}\right) + 2^{(\xi-2)n}\left(\frac{-\xi(\xi-1)}{2}\alpha^2 + \mathcal{O}(\alpha)\right) + \mathcal{O}\left(2^{(\xi-3)n}\alpha^3\right)}{2^{\xi n} + o(2^{\xi n})}\right] \frac{H_\alpha}{J_\alpha}$$

So we have:

$$\frac{H_{\alpha+\xi}}{J_{\alpha+\xi}} \geq \left[1 - \frac{2^{(\xi-2)n}\left(\frac{\xi(\xi-1)}{2}\alpha^2 + \mathcal{O}(\alpha)\right)}{2^{\xi n} + o(2^{\xi n})}\right]^{\alpha} \frac{H_\xi}{J_\xi}$$

We also have $H_\xi > J_\xi$ since $H_\xi = 2^{\xi n} > J_\xi = 2^n(2^n-1)\cdots(2^n-\xi+1)$. So

$$H_{\alpha+\xi} \geq J_{\alpha+\xi}\left[1 - \frac{\frac{\xi(\xi-1)}{2}\alpha^3 + \mathcal{O}(\alpha^2)}{2^{2n} + o(2^{2n})}\right]$$

So if $\xi^2\alpha^3 \ll 2^{2n}$, $H_\alpha \geq J_\alpha(1-\varepsilon)$, where ε is very small. (3)
Now to extend this result (3) with the condition $\xi\alpha \ll 2^n$ instead of $\xi^2\alpha^3 \ll 2^{2n}$, we will improve the evaluation (2) of $H_{\alpha+\xi}$ from H_α.

B.2 General Properties for Any ξ_{max}

h_α is by definition the number of $P_1, \cdots, P_\alpha$ pairwise distinct, elements of I_n, and solution of (A), where (a) is a system of equations $P_i \oplus P_j = \lambda_k$. We say that P_i and P_j are "in the same block" if when P_i is fixed, then P_j is fixed from the equations of (A). We denote by ξ_{max} the maximum number of P_i in the same block. The idea is to evaluate h_α by induction on the number of blocks, i.e. to evaluate $h_{\alpha+\xi}$ from h_α, where $h_{\alpha+\xi}$ is the number of $P_1, \cdots, P_\alpha, P_{\alpha+1}, \cdots, P_{\alpha+\xi}$ pairwise distinct, elements of I_n, solution of (A) and solution of this block of $(\xi-1)$ equations $P_{\alpha+1}, \cdots, P_{\alpha+\xi}$:

$$\begin{gathered} P_{\alpha+2} = P_{\alpha+1} \oplus \lambda'_2, \\ P_{\alpha+3} = P_{\alpha+1} \oplus \lambda'_3, \\ \cdots \\ P_{\alpha+\xi} = P_{\alpha+1} \oplus \lambda'_\xi \\ (\xi \leq \xi_{max}). \end{gathered}$$

We will say that $P_1, \cdots, P_\alpha$ are solution of h_α when they are solution of (A). We start from a solution $P_1, \cdots, P_\alpha$ of (A) and we want to complete it to get the solution of $h_{\alpha+\xi}$. For this we have to choose $x = P_{\alpha+1} \oplus P_1$ such that x will not create a collision $P_j = P_{\alpha+1}$ or $P_j = P_{\alpha+2}$, $\cdots$, $P_j = P_{\alpha+\xi}$, $1 \leq j \leq \alpha$. This means: $x \oplus P_1 \neq P_j$, $x \oplus \lambda'_2 \oplus P_1 \neq P_j$, $\cdots$, $x \oplus \lambda'_\xi \oplus P_1 \neq P_j$, $1 \leq j \leq \alpha$. So this means $x \notin V$ with $V = \bigcup_{i=1}^{\xi} V_i$, with $V_i = \{P_1 \oplus \lambda'_i \oplus P_j, 1 \leq j \leq \alpha\}$ (by convention we define $\lambda'_1 = 0$). We have $\forall i$, $1 \leq i \leq \xi$, $|V_i| = \alpha$ (since the P_j values, $1 \leq j \leq \alpha$, are pairwise distinct).

$|V| = |\bigcup_{i=1}^{\xi} V_i| = \sum_{i=1}^{\xi} |V_i| - \sum_{i<j}^{\xi} |V_i \cap V_j| + \sum_{i<j<k}^{\xi} |V_i \cap V_j \cap V_k| + \cdots + (-1)^{\xi+1}|V_1 \cap \cdots \cap V_\xi|$
So

$$h_{\alpha+\xi} = \sum_{(P_1,\cdots,P_\alpha) \text{ solution of } h_\alpha} (2^n - |V|)$$

So we have:

Theorem 13

$$h_{\alpha+\xi} = \sum_{(P_1,\cdots,P_\alpha)\ solution\ of\ h_\alpha} (2^n - \xi\alpha + \sum_{i_1<i_2}^{\xi} |V_{i_1} \cap V_{i_2}|$$

$$- \sum_{i_1<i_2<i_3}^{\xi} |V_{i_1} \cap V_{i_2} \cap V_{i_3}| + \cdots + (-1)^{\xi} |V_1 \cap \cdots \cap V_\xi|)$$

When i_1 and i_2 are fixed,

$$|V_{i_1} \cap V_{i_2}| = \sum_{1\le j\le\alpha, 1\le j'\le\alpha} \text{Number of } P_1,\cdots,P_\alpha \text{ solution of } h_\alpha$$

$$\text{plus equation } P_j \oplus P_{j'} = \lambda'_{i_1} \oplus \lambda'_{i_2}$$

Now when we add to (A) the equality $P_j \oplus P_{j'} = \lambda'_{i_1} \oplus \lambda'_{i_2}$, 3 cases can occur:

Case 1. $\lambda'_{i_1} \oplus \lambda'_{i_2} = P_i \oplus P_j$ was already an equation of (A). Here this means $\lambda'_{i_1} \oplus \lambda'_{i_2} = \lambda_i$ for all value i, $1 \le i \le \alpha$. Remark: $\lambda'_{i_1} \oplus \lambda'_{i_2} = \lambda_i$ creates 2 collisions in $V_{i_1} \cap V_{i_2}$: it creates $\lambda'_{i_1} \oplus P_1 \oplus P_i = \lambda'_{i_2} \oplus P_1 \oplus P_j$ and $\lambda'_{i_1} \oplus P_1 \oplus P_j = \lambda'_{i_2} \oplus P_1 \oplus P_i$.

Case 2. $\lambda'_{i_1} \oplus \lambda'_{i_2} = P_i \oplus P_j$ is in contradiction with the equations of (A). This can come from the fact that $P_i \oplus P_j = \lambda_k$ is in (A) and $\lambda_k \neq \lambda'_{i_1} \oplus \lambda'_{i_2}$. Or this can come from the fact that $P_i \oplus P_{i'} = \lambda_{i''}$ is in (A), $P_j \oplus P_{j'} = \lambda_{j''}$ is in (A), so from $P_i \oplus P_j = \lambda'_{i_1} \oplus \lambda'_{i_2}$ we get $P_{j'} = \lambda_{j''} \oplus \lambda'_{i_1} \oplus \lambda'_{i_2} \oplus P_i$, $P_{i'} = \lambda_{i''} \oplus \lambda'_{i_1} \oplus \lambda'_{i_2} \oplus P_j$, and $P_{i'} \oplus P_{j'} = \lambda_{i''} \oplus \lambda_{j''} \oplus \lambda'_{i_1} \oplus \lambda'_{i_2}$. This is impossible if $\lambda'_{i_1} \oplus \lambda'_{i_2} = \lambda_{j''}$, $\lambda'_{i_1} \oplus \lambda'_{i_2} = \lambda_{i''}$ or $\lambda'_{i_1} \oplus \lambda'_{i_2} = \lambda_{i''} \oplus \lambda_{j''}$, since the P_k values are pairwise distinct.

Case 3. The equation $\lambda'_{i_1} \oplus \lambda'_{i_2} = P_i \oplus P_j$ is not in contradiction with the equations of (A), and is not a consequence of the equations of (A). We will say that this case is the "generic" case, and we will denote by h'_α the number of $P_1,\cdots,P_\alpha$ solution of (A) and $\lambda'_{i_1} \oplus \lambda'_{i_2} = P_i \oplus P_j$ when we are in such "generic" case.

The value of h'_α is dependent on the λ_i values. However we will see in section B.3 that all the values h'_α are very near. In this section B.3, we will compare h'_α and h_α in order to get from theorem 13 a relation between $h_{\alpha+\xi}$ and h_α.

B.3 Relations Between h'_α and h_α for Any ξ_{max}

The sets W_1 and W_2. The aim of this section is to prove that $h'_\alpha \ge h_\alpha(1 - O(\frac{\xi^2_{max}\alpha}{2^{2n}}))$ (by hypothesis here all the equations are independent and compatible, as explained in the definition of h'_α). Notice that here we look for an evaluation in $\frac{1}{2^{2n}}$, not only in $\frac{1}{2^n}$. For this, the idea is to evaluate h_α and h'_α from $h_{\alpha-\xi-\xi'}$ (see figure 5).

Notations. We denote by ξ the number of variables P_i of the first block, and by ξ' the number of variables P_i of the second block. A "block" of variables P_i is a set of variables P_i such that when one variable P_i of this block is fixed, then the other variables P_j of the block are fixed from the equations (A). We have: $\xi \leq \xi_{max}$ and $\xi' \leq \xi_{max}$. The ξ equations of the first block will be denoted like this: $\forall i,\ 1 \leq i \leq \xi,\ P_i = P_\xi \oplus \lambda_i$ (1). So by convention, we denote $\lambda_\xi = 0$. The ξ' equations of the second block will be denoted like this: $\forall i,\ \xi + 1 \leq i \leq \xi + \xi',\ P_i = P_{\xi+\xi'} \oplus \lambda_i$ (2). So by convention, we denote $\lambda_{\xi+\xi'} = 0$. We denote by $\lambda' = P_\xi \oplus P_{\xi+\xi'}$ (3) and $x = P_{\xi+\xi'} \oplus P_{\xi+\xi'+1}$ (4).

Theorem 14. *We have:*

$$h'_\alpha = \sum_{(P_{\xi+\xi'+1},\cdots,P_\alpha)\ solution\ of\ h_{\alpha-\xi-\xi'}} (2^n - (|W_1| + |W_2|) + |W_1 \cap W_2|) \qquad (5)$$

and

$$h_\alpha \leq \sum_{(P_{\xi+\xi'+1},\cdots,P_\alpha)\ solution\ of\ h_{\alpha-\xi-\xi'}} (2^n - |W_1|)(2^n - |W_2|) \qquad (6)$$

with $W_1 = \bigcup_{i=1}^{\xi} A_i$ *with* $\forall i,\ 1 \leq i \leq \xi,\ A_i = \{P_j \oplus \lambda_i \oplus \lambda', \xi + \xi' + 1 \leq j \leq \alpha\}$ *and* $W_2 = \bigcup_{i=1}^{\xi'} A'_i$ *with* $\forall i,\ 1 \leq i \leq \xi',\ A'_i = \{P_j \oplus \lambda_{\xi+i}, \xi + \xi' + 1 \leq j \leq \alpha\}$.

Proof of (5). We start from a solution $P_{\xi+\xi'+1}, \cdots, P_\alpha$ of $h_{\alpha-\xi-\xi'}$ and we want to complete it to get the solutions of h'_α. For this we have to choose $x = P_{\xi+\xi'} \oplus$

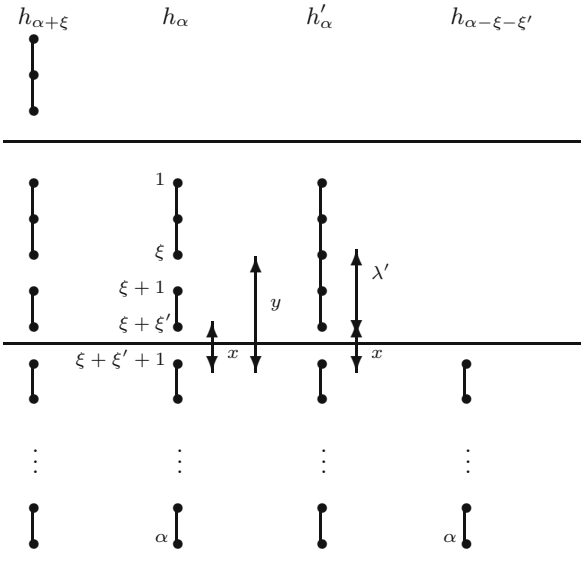


Fig. 5. We want to compare h_α and h'_α. This figure illustrates that we will do this by evaluating h_α from $h_{\alpha-2}$ blocks$(= h_{\alpha-\xi-\xi'})$ and h'_α from $h_{\alpha-2}$ blocks$(= h_{\alpha-\xi-\xi'})$.

$P_{\xi+\xi'+1}$ such that x will not create a collision $P_j = P_i$, $\xi + \xi' + 1 \le j \le \alpha$, $1 \le i \le \xi + \xi'$.

So from (1) and (2) and (5), we want x such that: $\forall i$, $1 \le i \le \xi$, $\forall j$, $\xi+\xi'+1 \le j \le \alpha$, $P_\xi \oplus \lambda_i \ne P_j$, i.e. $P_{\xi+\xi'+1} \oplus x \oplus \lambda' \oplus \lambda_i \ne P_j$ and $\forall i$, $\xi + 1 \le i \le \xi + \xi'$, $\forall j$, $\xi + \xi' + 1 \le j \le \alpha$, $P_{\xi+\xi'} \oplus \lambda_i \ne P_j$, i.e. $P_{\xi+\xi'+1} \oplus x \oplus \lambda_i \ne P_j$.

So we want $x \notin W_1'$ and $x \notin W_2'$ with $W_1' = \{P_{\xi+\xi'+1} \oplus \lambda' \oplus \lambda_i \oplus P_j, 1 \le i \le \xi, \xi+\xi'+1 \le j \le \alpha\}$ and $W_2' = \{P_{\xi+\xi'+1} \oplus \lambda_i \oplus P_j, \xi+1 \le i \le \xi+\xi', \xi+\xi'+1 \le j \le \alpha\}$ and

$$h'_\alpha = \sum_{(P_{\xi+\xi'+1},\cdots,P_\alpha) \text{ solution of } h_{\alpha-\xi-\xi'}} (2^n - |W_1' \cup W_2'|)$$

Now $|W_1' \cup W_2'| = |W_1'| + |W_2'| - |W_1' \cap W_2'|$, and we have $|W_1'| = |W_1|$, $|W_2'| = |W_2|$ and $|W_1' \cap W_2'| = |W_1 \cap W_2|$, since a translation by $P_{\xi+\xi'+1}$ does not change the number of elements. So we have (5) as claimed.

Proof of (6). We start from a solution $P_{\xi+\xi'+1}, \cdots, P_\alpha$ of $h_{\alpha-\xi-\xi'}$ and we want to complete it to get the solutions of h_α. For this we have to choose $x = P_{\xi+\xi'} \oplus P_{\xi+\xi'+1}$ and $y = P_\xi \oplus P_{\xi+\xi'+1}$ such that x will not create a collision $P_j = P_i$, $\xi + \xi' + 1 \le j \le \alpha$, $\xi + 1 \le i \le \xi + \xi'$ (7), y will not create a collision $P_j = P_i$, $\xi+\xi'+1 \le j \le \alpha$, $1 \le i \le \xi$ (8), and x and y will not create a collision $P_i = P_k$, $1 \le i \le \xi$, $\xi + 1 \le k \le \xi + \xi'$ (9). So

$$h_\alpha \le \sum_{(P_{\xi+\xi'+1},\cdots,P_\alpha) \text{ solution of } h_{\alpha-\xi-\xi'}} (2^n - |W_1|)(2^n - |W_2|)$$

as claimed. (We have here $\le$ and not $=$ since we removed condition (9)).

In order to compare h'_α and h_α, we see from theorem 14 that we want to evaluate

$$\chi = \sum_{(P_{\xi+\xi'+1},\cdots,P_\alpha) \text{ solution of } h_{\alpha-\xi-\xi'}} \left(\frac{|W_1 \cap W_2|}{2^n} - \frac{|W_1||W_2|}{2^{2n}} \right).$$

We have $W_1 = \bigcup_{i=1}^{\xi} A_i$, $W_2 = \bigcup_{i=1}^{\xi'} A'_i$, and we have $\forall i$, $1 \le i \le \xi$, $|A_i| = \alpha - \xi - \xi'$ and $\forall i$, $1 \le i \le \xi'$, $|A'_i| = \alpha - \xi - \xi'$ (because the P_j solutions of $h_{\alpha-\xi-\xi'}$ are pairwise distinct).

Evaluation of $\sum_P \frac{|W_1||W_2|}{2^{2n}}$. For simplicity, we will denote from now on: $\sum_P$ for

$$\sum_{(P_{\xi+\xi'+1},\cdots,P_\alpha) \text{ solution of } h_{\alpha-\xi-\xi'}}.$$

We have $W_1 = \bigcup_{i=1}^{\xi} A_i$. So

$$|W_1| = \sum_{i=1}^{\xi} |A_i| - \sum_{i<j}^{\xi} |A_i \cap A_j| + \sum_{i<j<k}^{\xi} |A_i \cap A_j \cap A_k| + \cdots + (-1)^{\xi+1} |A_1 \cap \cdots \cap A_\xi|.$$

$$= \xi(\alpha-\xi-\xi') - \sum_{i<j}^{\xi} |A_i \cap A_j| + \sum_{i<j<k}^{\xi} |A_i \cap A_j \cap A_k| + \cdots + (-1)^{\xi+1} |A_1 \cap \cdots \cap A_\xi|.$$

Similarly, $|W_2| = \xi'(\alpha - \xi - \xi') - \sum_{i<j}^{\xi'} |A'_i \cap A'_j| + \sum_{i<j<k}^{\xi'} |A'_i \cap A'_j \cap A'_k| + \cdots +$

$$(-1)^{\xi'+1} |A'_1 \cap \cdots \cap A'_\xi|.$$

$$\frac{|W_1||W_2|}{2^{2n}} = \frac{1}{2^{2n}} [\xi(\alpha - \xi - \xi') - \sum_{i<j}^{\xi} |A_i \cap A_j| + \sum_{i<j<k}^{\xi} |A_i \cap A_j \cap A_k|$$
$$+ \cdots + (-1)^{\xi+1} |A_1 \cap \cdots \cap A_\xi|] \cdot [\xi'(\alpha - \xi - \xi') - \sum_{i<j}^{\xi'} |A'_i \cap A'_j|$$
$$+ \sum_{i<j<k}^{\xi'} |A'_i \cap A'_j \cap A'_k| + \cdots + (-1)^{\xi'+1} |A'_1 \cap \cdots \cap A'_\xi|]$$

Now when $i_1, \cdots, i_k$ are pairwise distinct, since here we have by hypothesis a system of independent equalities with no contradiction, we have:

$$\sum_P |A_{i_1} \cap \cdots \cap A_{i_k}| = h_{\alpha-\xi-\xi'} \left(\frac{\alpha^k}{2^{n(k-1)}} + O\left(\frac{\alpha^{k-1}}{2^{n(k-1)}} \right) \right).$$

Similarly

$$\sum_P |A'_{i_1} \cap \cdots \cap A'_{i_k}| = h_{\alpha-\xi-\xi'} \left(\frac{\alpha^l}{2^{n(l-1)}} + O\left(\frac{\alpha^{l-1}}{2^{n(l-1)}} \right) \right).$$

So we have:

$$\sum_P \frac{|W_1||W_2|}{2^{2n}} = \frac{1}{2^{2n}} [\xi(\alpha-\xi-\xi') - \frac{\xi(\xi-1)}{2} \left(\frac{\alpha^2 + O(\alpha)}{2^n} \right) + \frac{\xi(\xi-1)(\xi-2)}{6} \left(\frac{\alpha^3 + O(\alpha^2)}{2^{2n}} \right) + \cdots$$
$$+ \frac{\xi(\xi-1)\cdots(\xi-i+1)}{i!} \left(\frac{\alpha^i + O(\alpha^{i-1})}{2^{n(i-1)}} \right) + \cdots] \cdot [\xi'(\alpha-\xi-\xi') - \frac{\xi'(\xi'-1)}{2} \left(\frac{\alpha^2 + O(\alpha)}{2^n} \right)$$
$$+ \frac{\xi'(\xi'-1)(\xi'-2)}{6} \left(\frac{\alpha^3 + O(\alpha^2)}{2^{2n}} \right) + \cdots + \frac{\xi'(\xi'-1)\cdots(\xi'-i+1)}{i!} \left(\frac{\alpha^i + O(\alpha^{i-1})}{2^{n(i-1)}} \right) + \cdots] h_{\alpha-\xi-\xi'}$$

So we have:

$$\sum_P \frac{|W_1||W_2|}{2^{2n} \cdot h_{\alpha-\xi-\xi'}} = \frac{\xi\xi'(\alpha^2 - 2(\xi+\xi')\alpha + (\xi+\xi')^2)}{2^{2n}}$$

.

$$- \frac{\alpha^3 + O(\alpha^2)}{2^{3n}} \left(\frac{\xi'(\xi'-1)\xi}{2} + \frac{\xi(\xi-1)\xi'}{2} \right)$$

$$+\frac{\alpha^4+O(\alpha^3)}{2^{4n}}\left(\frac{\xi\xi'(\xi'-1)(\xi'-2)}{6}+\frac{\xi(\xi-1)\xi'(\xi'-1)}{4}+\frac{\xi\xi'(\xi-1)(\xi-2)}{6}\right)$$

$$+\cdots+\frac{\alpha^i+O(\alpha^{i-1})}{2^{in}}\left(\frac{\xi\xi'(\xi'-1)\cdots(\xi'-i+2)}{(i-1)!}\right.$$

$$+\frac{\xi(\xi-1)\xi'(\xi'-1)\cdots(\xi'-i+3)}{2!(i-2)!}+\cdots$$

$$\left.+\frac{\xi(\xi-1)\cdots(\xi-j+1)\xi'(\xi'-1)\cdots(\xi'-i+j+1)}{j!(i-j)!}+\cdots\right) \tag{10}$$

Let $\mu_i=\sum_{j=1}^{i-1}\frac{\xi(\xi-1)\cdots(\xi-j+1)\xi'(\xi'-1)\cdots(\xi'-i+j+1)}{j!(i-j)!}$ (with by convention each product here is equal to 0 if its value is <0). Then (10) can be written like this:

$$\sum_P\frac{|W_1||W_2|}{2^{2n}\cdot h_{\alpha-\xi-\xi'}}=\frac{\xi\xi'}{2^{2n}}(\alpha^2-2(\xi+\xi')\alpha+(\xi+\xi')^2)+\sum_{i=3}^{\xi\xi'}(-1)^i\mu_i\frac{\alpha^i+O(\alpha^{i-1})}{2^{in}}.$$

Remark. A simple approximation of μ_i is $\mu_i\le\xi^i$.
Proof.

$$\mu_i\le\xi\cdot\frac{\xi^{i-1}}{(i-1)!}+\frac{\xi^2\xi^{i-2}}{2!(i-2)!}+\frac{\xi^3\xi^{i-3}}{3!(i-3)!}+\cdots+\frac{\xi^{i-1}\xi}{(i-1)!}.$$

Now if $1\le x\le i-1$, $\frac{2}{x!(i-x)!}\le\frac{2}{2^{x-1}\cdot 2^{i-x-1}}\le\frac{2}{2^{i-2}}\le\frac{8}{2^i}$. So $\mu_i\le\xi^i\cdot 4\cdot\frac{i}{2^i}$, so if $i\ge 4$, $\mu_i\le\xi^i\cdot 4\cdot\frac{i}{2^i}\le\xi^i$. Moreover $\mu_i\le\xi^i$ is also true for $i=1,2,3$ since: $\mu_1=0$, $\mu_2=\xi^2$, $\mu_3\le\xi^2(\xi-1)\le\xi^3$ ($\mu_4\le\xi^4(\frac{1}{4}+\frac{2}{6})\le\xi^4$).

Evaluation of $\sum_P\frac{|W_1\cap W_2|}{2^n}$. We have $W_1=\bigcup_{i=1}^{\xi}A_i$ and $W_2=\bigcup_{i=1}^{\xi'}A'_i$. So $W_1\cap W_2=(\bigcup_{i=1}^{\xi}A_i)\cap(\bigcup_{i=1}^{\xi'}A'_i)$.

Theorem 15.

$$|W_1\cap W_2|=|(\bigcup_{i=1}^{\xi}A_i)\cap(\bigcup_{i=1}^{\xi'}A'_i)|$$

$$=\sum_{g=1}^{\xi'}\sum_{f=1}^{\xi}(-1)^{f+g}$$

$$\sum_{1\le i_1<i_2<\cdots<i_f\le\xi,1\le j_1<j_2<\cdots<j_g\le\xi'}|(A_{i_1}\cap\cdots A_{i_f})\cap(A'_{j_1}\cap\cdots A'_{j_g})|$$

Proof. This is a classical result on sets (the proof is very easy by induction on ξ and ξ').

Let $\nu_i = \sum_{j=1}^{i-1} \frac{\xi(\xi-1)\cdots(\xi-j+1)}{j!} \cdot \frac{\xi'(\xi'-1)\cdots(\xi'-i+j+1)}{(i-j)!}$. From theorem 15 we get:

$$\sum_P \frac{|W_1 \cap W_2|}{2^n} = h_{\alpha-\xi-\xi'} \left(\sum_{i=2}^{\xi\xi'} (-1)^i \frac{\nu_i \alpha^i + O(\alpha^{i-1})}{2^{ni}} \right)$$

Moreover, we have for all integer i, $\mu_i = \nu_i$ (μ_i was defined in section B.3). So we have:$|\chi| \leq O\left(\frac{\xi\xi'\alpha}{2^{2n}}\right) h_{\alpha-\xi-\xi'}$ so we have $|\chi| \leq O\left(\frac{\xi^2_{max}\alpha}{2^{2n}}\right) h_{\alpha-\xi-\xi'}$ so $h'_\alpha \geq h_\alpha \left(1 - O\left(\frac{\xi^2_{max}\alpha}{2^{2n}}\right)\right)$ as wanted.

C Proof of Security for ψ^5

As mentioned in [7], from the Theorem $P_i \oplus P_j$ we can prove that there does not exist any adaptive chosen plaintext/ciphertext attack against a random Feistel scheme of $2n$ bits $\rightarrow 2n$ bits with ≤ 5 rounds when the number of queries is small compared with 2^n. We give here the main ideas to prove this results on 5 rounds random Feistel scheme from the Theorem $P_i \oplus P_j$.

An exact formula for H with 5 rounds. Let us define a "framework" as a set of equations $X_i = X_j$ or $Y_i = Y_j$ or $Z_i = Z_j$, $i < j$, where the X_i, Y_i, Z_i values are in I_n. We will say that two frameworks are equal if they imply exactly the same set of equalities in X, Y and Z. Let H be the number of $(f_1, \cdots, f_5) \in F_n^5$ (where F_n is the set of all functions of $I_n \rightarrow I_n$) such that $\forall i$, $1 \leq i \leq m$, $\psi^5(f_1, \cdots, f_5)[L_i, R_i] = [S_i, T_i]$ where $\psi^5(f_1, \cdots, f_5)$ is a Feistel scheme with $f_1, \cdots, f_5$ as round functions. Then from [7] we know that the exact value of H is:

$$H = \frac{|F_n|^5 \cdot 2^{n(r+s)}}{2^{5nm}} \cdot \sum_{all\ frameworks \mathcal{F}} 2^{n(x+y+z)} [Number\ of\ X_i, Z_i\ satisfying\ (C_1)]$$

$$\cdot [Number\ of\ Y_i\ satisfying\ (C_2)]$$

where r (respectively s, x, y, z) is the number of independent equalities $R_i = R_j$, $i < j$ (respectively $S_i = S_j$, $X_i = X_j$, $Y_i = Y_j$, $Z_i = Z_j$) and with

$$(C_1) : \begin{cases} R_i = R_j \Rightarrow X_i \oplus X_j = L_i \oplus L_j \\ Y_i = Y_j \text{ is in } \mathcal{F} \Rightarrow X_i \oplus X_j = Z_i \oplus Z_j \\ S_i = S_j \Rightarrow Z_i \oplus Z_j = T_i \oplus T_j \\ \text{The only equations } X_i = X_j, i < j, \text{ are exactly those implied by } \mathcal{F} \\ \text{The only equations } Z_i = Z_j, i < j, \text{ are exactly those implied by } \mathcal{F}. \end{cases}$$

$$(C_2) : \begin{cases} X_i = X_j \text{ is in } \mathcal{F} \Rightarrow Y_i \oplus Y_j = R_i \oplus R_j \\ Z_i = Z_j \text{ is in } \mathcal{F} \Rightarrow Y_i \oplus Y_j = S_i \oplus S_j \\ \text{The only equations } Y_i = Y_j, i < j, \text{ are exactly those implied by } \mathcal{F}. \end{cases}$$

How to use this formula and Theorem $P_i \oplus P_j$. We know that with 5 rounds (unlike what will happen with 6 rounds) we can have $[L_i, R_i], [S_i, T_i]$ values with H much larger or much smallest than the average value of H (see [8] pp.120–122). However in these examples we have a small H only when we have large chains of equations in R and S that cannot be generated in CPCA-2.

In CPCA-2 we have two types of queries: direct and inverse.

In a direct query, $[L_i, R_i]$ can be chosen from the values $[L_j, R_j, S_j, T_j]$, $j < i$, but then $[S_i, T_i]$ is almost perfectly random if we analyze a random permutation f of $2n$ bits $\rightarrow$ $2n$ bits. In this case, the number of indices $j < i$ such that $S_j = S_i$ is very small: about $\frac{m}{2^n}$ in average, so the number N of (i, j) such that $j < i$ and i is an index of a direct query and $S_j = S_i$ is $N \leq O\left(\frac{m^2}{2^n}\right)$.

In an inverse query, $[S_i, T_i]$ can be chosen from the values $[L_j, R_j, S_j, T_j]$, $j < i$, but then $[L_i, R_i]$ is almost perfectly random if we analyze a random permutation f of $2n$ bits $\rightarrow$ $2n$ bits. In this case, the number of indices $j < i$ such that $R_j = R_i$ is very small: about $\frac{m}{2^n}$ in average, so the number N of (i, j) such that $j < i$ and i is an index of an inverse query and $R_j = R_i$ is $N \leq O\left(\frac{m^2}{2^n}\right)$.

In the formula for H above, the condition (C2) will not create any problem: most of the frameworks will generate in (C2) a system in Y_i variables with $\xi_{max}\alpha \ll 2^n$, and we will use on it the Theorem $P_i \oplus P_j$.

The analysis of (C1) is a bit more complex since we can have large lines of equalities in R, or in S. Let ξ be a fixed integer. Let E be the set of all $[L_i, R_i, S_i, T_i]$, $1 \leq i \leq m$, such that for most of the frameworks $\mathcal{F}$ we have: for all index k, $1 \leq k \leq m$, [The length of the lines in R, Y arriving in k is $\leq \xi$] or [The length of the lines in S, Y arriving in k is $\leq \xi$]. By "the length of the lines in R, Y arriving in k", we mean the maximum number of pairwise distinct indices $i_1, \cdots, i_\xi$ such that $i_\xi = k$ and $\forall j$, $1 \leq j \leq \xi - 1$, we have $R_{i_j} = R_{i_{j+1}}$ or $Y_{i_j} = Y_{i_{j+1}}$ is in $\mathcal{F}$ (similar definition for lines in S, Y).

From our analysis above of direct and inverse queries, we can prove:

Theorem 16. *For all super distinguishing circuit ϕ with m oracle gates, the probability that $[L_i, R_i, S_i, T_i](\phi)$, $1 \leq i \leq m$, be in this set E is $\geq 1 - \beta$, when ϕ acts on a random permutation of $I_{2n} \rightarrow I_{2n}$, where β can be chosen $\ll 1$ when $m \ll 2^n$.*

Now we will analyze (C1) like this:

Case 1. When k is an index such that the length of the lines in R, Y arriving in k is $\leq \xi$, then we keep X_k as a variable, and we fix Z_k.

Case 2. If we are not in Case 1, then k is an index such that the length of the lines in S, Y arriving in k is $\leq \xi$. Then we keep Z_k as a variable, and we fix X_k.

Then the condition (C1) will be analyzed by introducing pairwise distinct variables P_i of I_n, and here (C1) will give a system with $\alpha\xi \ll 2^n$ in these variables, so we can use the Theorem $P_i \oplus P_j$, to prove that H is always near the average value for all $[L_i, R_i, S_i, T_i]$ that can be generated in CPCA-2. This proves CPCA-2 by using theorem 3.4 of [7] p.518 (called "coefficient H technique").

An FPGA Implementation of CCM Mode Using AES

Emmanuel López-Trejo, Francisco Rodríguez-Henríquez, and Arturo Díaz-Pérez

Computer Science Section, Electrical Engineering Department, Centro de Investigación y de Estudios Avanzados del IPN, Av. Instituto Politécnico Nacional No. 2508, México D.F
elopez@computacion.cs.cinvestav.mx, {francisco, adiaz}@cs.cinvestav.mx

Abstract. Due to the exponential growth of wireless and mobile applications, security has become a paramount design aspect. New techniques have been proposed for replacing the broken Wired Equivalent Privacy (WEP) protocol, which arguably is the most widely security tool used up to now in wireless environments. Under this scenario, AES in CCM (Counter with CBC-MAC) mode has been included in the IEEE 802.11i wireless standard as a promising alternative to the compromised WEP protocol. In this contribution, we present an FPGA implementation of the CCM mode of operation using AES as its block cipher. Our design achieves a throughput of 1.05 Gbits/Sec with reasonable area requirements.

1 Introduction

Over the years, the Wired Equivalent Privacy (WEP) protocol has been the most widely security tool used for protecting information in wireless environments. However, this protocol was broken in 2001 by Fluhrer et al. [1]. Based on that attack, nowadays there exist a variety of programs that can be downloaded from Internet to break the WEP Protocol in few seconds and with almost no effort. This situation has led to a search for new security mechanisms for guaranteeing reliable ways of protecting information in wireless mobile environments.

Under this scenario, AES in CCM (Counter with CBC-MAC) mode has become one of the most promising solutions for achieving security in wireless networks. This mode simultaneously offers two key security services, namely, data Authentication and Encryption. In this contribution, we present an efficient reconfigurable hardware implementation of AES in CCM mode. We discuss main implementation aspects as well as the experimental results obtained.

The rest of this paper is organized as follows. Section 2 includes a brief introduction to the CCM Mode. Then, in Section 3 a short description of the AES encryption process is given. Section 4 presents main design considerations

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 322–334, 2006.
© Springer-Verlag Berlin Heidelberg 2006

for the CCM Mode and AES Encryptor Core architectures. In section 5, implementation results are summarized and then compared with other similar designs previously reported. Finally, concluding remarks are given in Section 6.

2 The CCM Mode

Proposed by Whiting et. al. [2], CCM stands for Counter with CBC-MAC. This means that two different modes are combined into one, namely, the CTR mode and the CBC-MAC. CCM is a generic authenticate-and-encrypt block cipher scheme that has been specifically designed for being use in combination with a 128-bit block cipher, such as AES. Presently, CCM mode has become part of the new 802.11i IEEE standard. In the rest of this section we give a short summary of the CCM Mode and the corresponding Authentication, Encryption, Verification and Decryption algorithms associated to it.

2.1 CCM Primitives

Before sending a message, a sender must provide the following information [2]: (1) A suitable encryption key K for the block cipher to be used. (2) A nonce N of $15 - L$ bytes. Nonce value must be unique, meaning that the set of nonce values used with any given key shall not contain duplicate values. (3) The message m, consisting of a string of $l(m)$ bytes where $0 \leq l(m) < 2^{8L}$. (4) Additional authenticated data a, consisting of a string of $l(a)$ bytes where $0 \leq l(a) < 2^{64}$. This additional data is authenticated but not encrypted, and it is not included in the output of this mode.

Figure 2 shows CCM Authentication and Verification processes dataflow. Notice that because of the CBC feedback nature of the CCM mode, a pipeline approach for implementing AES is not possible. Therefore we have no option but to implement AES encryption core in an iterative fashion.

CCM Authentication starts by defining a sequence of blocks $B_0, B_1, ..., B_n$ and thereafter CBC-MAC is applied to those blocks so that the authentication field T can be obtained. Blocks B_is are defined as explained below.

First, the authentication data a is formatted by concatenating the string that encodes $l(a)$ with a itself, followed by organizing the resulting string in chunks of 16-byte blocks. The blocks so constructed are appended to the first configuration block B_0 [2]. Then, message blocks are added right after the (optional) authentication blocks a. Message blocks are formatted by splitting the message m into 16-byte blocks which will be the main part of the sequence of blocks $B_0, B_1, ..., B_n$ needed by the authentication mode. Finally, the CBC-MAC is computed as,

$$\begin{aligned} X_1 &:= AES_E(K, B_0) \\ X_{i+1} &:= AES_E(K, X_i \oplus B_i) \text{ for } i = 1, ..., n \\ T &:= firstMbytes(X_{n+1}) \end{aligned} \tag{1}$$

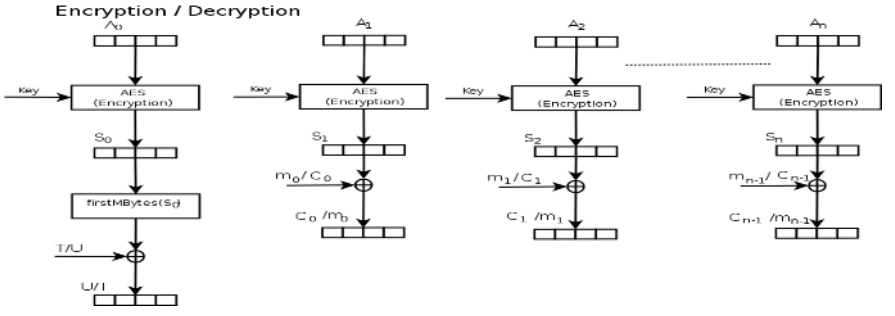


Fig. 1. Encryption and Decryption Process for the CCM Mode

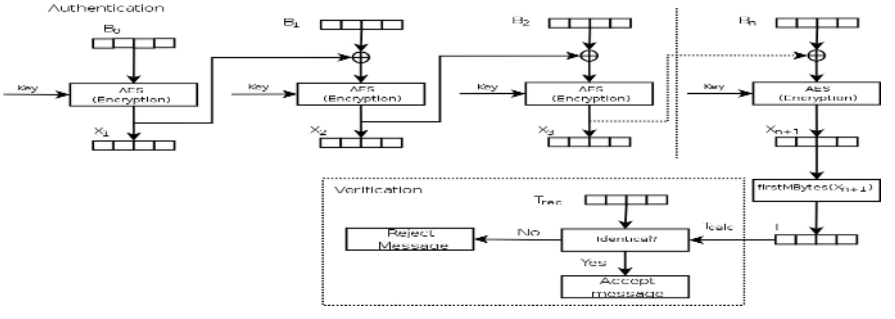


Fig. 2. Authentication and Verification Process for the CCM Mode

Where AES_E is the AES block cipher selected for encryption, and T is the MAC value defined as above. If it is needed, the ciphertext would be truncated in order to obtain T.

Figure 1 shows the CCM Encryption/Decryption process dataflow. CCM encryption is achieved by means of Counter (CTR) mode as,

$$\begin{aligned} S_i &:= AES_E(K, A_i) \text{ for } i = 0, 1, 2, ..., \\ C_i &:= S_i \oplus m_i \end{aligned} \tag{2}$$

See [2] for details about the construction of the A_i blocks. Plaintext m is encrypted by XORing each of its bytes with the first $l(m)$ bytes of the sequence produced by concatenating the cipher blocks $S_1, S_2, S_3, ...$, produced by Eq. 2. The authentication value is computed by encrypting T with the key stream block S_0 truncated to the desired length as,

$$U := T \oplus firstMbytes(S_0) \tag{3}$$

The final result c consists of the encrypted message m, followed by the encrypted authentication value U.

At the receiver side, the decryption process starts by recomputing the key stream to recover the message m and the MAC value T. Figure 1 shows how the Decryption Process is made in the CCM Mode.

Message and additional authentication data is then used to recompute the CBC-MAC value and check T. If the T value is not correct, the receiver should not reveal the decrypted message, the value T, or any other information. Figure 2 describes how the Verification Process is done.

It is important to notice that the Encryption Function of AES is used in Encryption as well as in Decryption. Therefore, AES Decryption functionality is not necessary in CCM-mode, which redounds in valuable hardware resources saving. That is why in this paper we only report the AES's implementation of the Encryption Function only.

3 AES Encryption

In this section we give a brief summary of the AES Encryption process. We describe the basic ideas for producing a ciphertext as well as some algorithmic shortcuts based on the work in [3].

3.1 Brief Description of the AES

The basic structure of AES consist of a message input (128 bits), a secret user key (128 bits) and a cipher message (128 bits) as the output. The AES cipher treats the input 128 bit block as a group of 16 bytes organized in a 4×4 matrix called *State* matrix. The algorithm consists of an initial transformation, followed by a main loop where nine iterations called *rounds* are executed. Each *round transformation* is composed of a sequence of four transformations: ByteSubstitution (BS), ShiftRows (SR), MixColumns (MC) and AddRoundKey (ARK). For each round of the main loop, a round key is derived from the original key through a process called *Key Scheduling*. Finally, a last round consisting of three transformations BS, SR and ARK is executed. Figure 3 shows the AES encryption process. See [4] for more details about the AES steps.

It is worth to remark that although an AES complete implementation includes the decryptor core, in our case, as we are interested in CCM mode only, which does not require the AES decryption we did not cover it (see [4] for a complete AES explanation).

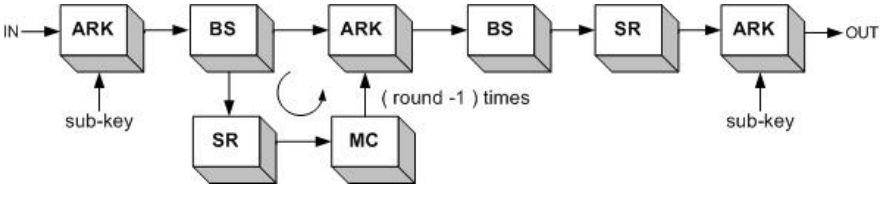


Fig. 3. AES Encryption Process

4 Implementation

In this Section we discuss design details utilized for CCM Mode and AES Encryptor Core implementations.

4.1 CCM Mode Implementation

In this Subsection we describe the rationale and main design considerations behind our implementation. We assumed that the user must provide the additional authentication data a as two blocks of 16 bytes each. This size was selected considering the typical length of a TCP/IP header information. Furthermore, it was assumed that the message m to be processed has a maximum length of 1024 bytes. Above design considerations are assumed in order to make our architecture simpler without loosing the main aspects of the Mode.

Figure 4 shows the CCM-mode general architecture, which comprises three main building modules, namely, Authentication module, Encryption module and a Control Unit module. All those three blocks together perform necessary operations for generating a valid cipher text and an encrypted authentication value. Notice that extra hardware is needed for the Verification and Decryption phases. In the rest of this Section we will explain how those three blocks were implemented, as well as experimental performance results in terms of time and hardware area.

CCM Authentication: Figure 5.A depicts the CCM Authentication module architecture. This module consists of an Authentication Block Generator, a CBC-MAC module and a Control Unit.

Authentication Block Generator: Authentication Block Generator is the architecture component responsible of generating the B_i blocks, (see [2] for details). Those blocks are generated according to the instructions indicated in the Control Word that the Control Unit sends in the "CW" line. That Control Word stipulates which block should be generated, either the B_0 block or the blocks

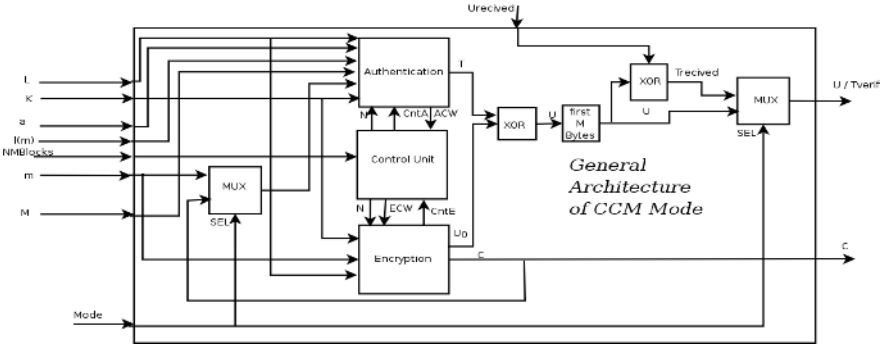


Fig. 4. CCM Mode General Architecture

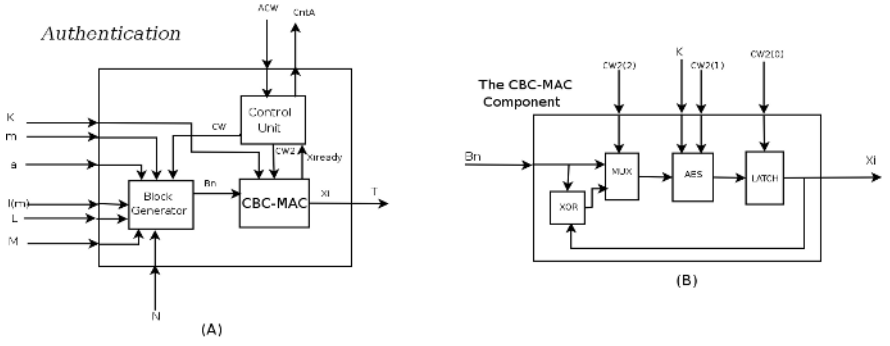


Fig. 5. Authentication Block

that corresponds to the additional data a or the ones corresponding to the message m. Each block is generated only when the previous block has already been ciphered by the CBC-MAC module described below.

CBC-MAC: Blocks B_i that were generated by the Block Generator are the inputs for the CBC-MAC. Any input block B_i (except for the block B_0) is XORed with the X_i that was computed previously. The result of this operation is encrypted using AES, and the resulting cipher text X_{i+1} is fed back to the next block B_{i+1}. Figure 5.B depicts the CBC-MAC process just outlined.

Authentication Control Unit: This Control Unit orchestrates the authentication process by receiving control signals from the General Control Unit. This block generates the appropriate Control Word for the Block Generator module and the one that indicates to the CBC-MAC Encryptor that a new B_i block can be processed. A 5-bit control word is utilized, where the LSB is used for controlling the CBC-MAC's latch. The second bit is used to start encryption with the AES block; the third bit controls which input will be selected by the MUX included in the CBC-MAC component, and finally, the last two bits indicate which type of block should be generated.

Authentication control Unit receives a signal when a B_i block has been processed within the CBC-MAC module. Thereafter, the control unit module produces the appropriate Control Word to generate the next block B_{i+1} and process it. Control Unit runs a counter that indicates which control word should be generated. The process of authentication begins when the General Control Unit indicates so with the "ACW" word. After receiving this signal, the whole process is controlled by the local Control Unit.

CCM Encryption: Figure 6.A shows the CCM encryption architecture. This module consists of an Encryption Block Generator, a CTR block and a Control Unit.

Encryption Block Generator: This module is responsible of generating the A_i blocks, (see section 2 for details), generated according to the Counter Function. In this implementation, the counter begins in 0 and it is incremented one by

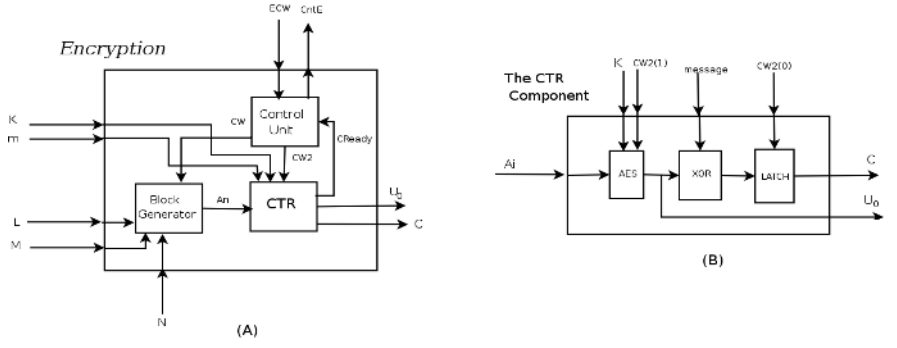


Fig. 6. Encryption Block

one. The blocks are formed with the Nonce and the counter value. Each block is generated when the CTR module has finished processing the previous one. Let us recall that Encryption begins only when the Authentication module is processing the second block with the additional data a. Based on this observation, we can parallelize the process of Authentication and Encryption by generating the A_1 Block first and all subsequent A_i blocks but the first one (A_0). When the CCM Authentication module has finished processing the last block, encryption Block Generator may proceed to generate the A_0 block in order to get S_0 (see figure 7.A for details).

The CTR Mode: The CTR Mode is the last step in the Encryption process. It encrypts A_i blocks using AES as block cipher in order to generate the S_i stream blocks. In our implementation, when a S_i block is ready, it is XORed with the appropriate message m block. Figure 6.B shows the internal composition of the CTR Mode. Notice that the U_0 value shown in Figure 6.B corresponds to the S_0 block. U_0 is not XORed with any block message, but instead, this value is used for encrypting the authentication value T computed as a part of the Authentication Process as it was shown in figure 2

Encryption Control Unit: The Implementation of this module is quite similar to the one described for the Authentication process. This Control Block is responsible of counting while indicates which block is the next to be generated. At the same time, it starts the CTR Mode by processing a block so that a valid ciphertext can be obtained. When a ciphertext is ready, it indicates to the General Control Unit that a new ciphertext can be stored. The implementation is based on the counter process that begins when the General Control Unit signals that is time to Encrypt the Message m and it keeps counting until the General Control Unit indicates to stop. This Control Unit receives a 2 bit Control Word that can select 4 different actions: "00" or "10" do nothing, remains in initial state, "01" Generate the S_0 Block, and "11" begin and continue counting.

General Control Unit: This module is the one that controls the Authentication and Encryption Processes. It synchronizes the information flow in order to parallelize the entire process and to achieve a good performance. The Control

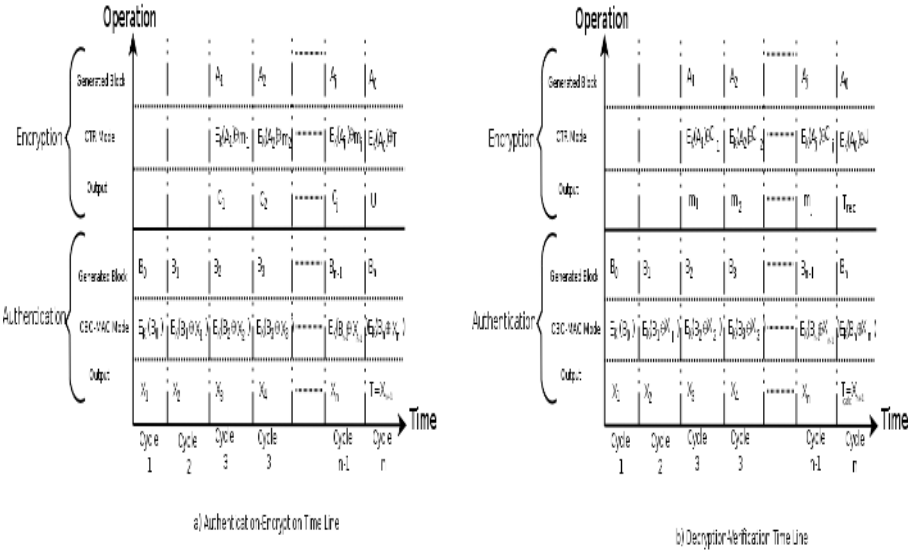


Fig. 7. Implementation Time line

Unit orders the Authentication Process to begin. The Encryption Process begins only after that the Authentication process has processed the first block (B_0) and has started processing the second of the two Additional Data a blocks.

Notice that the Authentication Process must authenticate the other a block and all message blocks. The Encryption Process, on the other hand, must encrypt only the message blocks and since it starts first, one could think that the Encryption process will finish first. However, since it is necessary the extra processing of the block S_0, our architecture manages to accomplish the execution of both processes at the same time. In this way we can compute the Encrypted Authentication Data U wich is done with extra hardware placed right after the Authentication and Encryption processes. Figure 7.A shows this process dataflow. Every unit in the time line represents 12 clock cycles.

Decryption and Verification: In our design, additional hardware is used to select if the input data is going to be authenticated and encrypted or decrypted and verified. In figure 4 it is shown that extra hardware as a MUX before the Authentication Process. This MUX is used to select if the source message is the one provided by the user (in case of authentication) or if it is the message that has been decrypted (when verifying). This way, when the input signal *Mode* is '0' the architecture performs authentication and encryption. When $Mode$='1', it decrypts and verifies.

The second Mux is used for selecting the output for the U value; when authenticating, the selection should be the computed value U, this is done with the function $firstMBytes$ and the XOR operation between the computed T

value and the S_0 Block. When Verifying, computed U value is XORed with the $Ureceived$ (sent by the transmitter entity) in order to verify whether the message integrity has been corrupted or not, if the XOR output is equal to zero, then the message is correct, otherwise, it is assumed that the received message is corrupted.

As in the case of the Authentication and Encryption the Verification process begins first, and the Decryption process starts after the Verification Process has processed two blocks. In this way, Verification can certify the received message that was just decrypted. Figure 7.B shows the dataflow of this process, every unit in time line represents 12 clock cycles.

4.2 AES Encryptor Core Implementation

In this Section we describe the AES encryptor core used for implementing the CCM mode just described in the previous Section. The General Architecture of an AES Encryptor core is shown in figure 8.

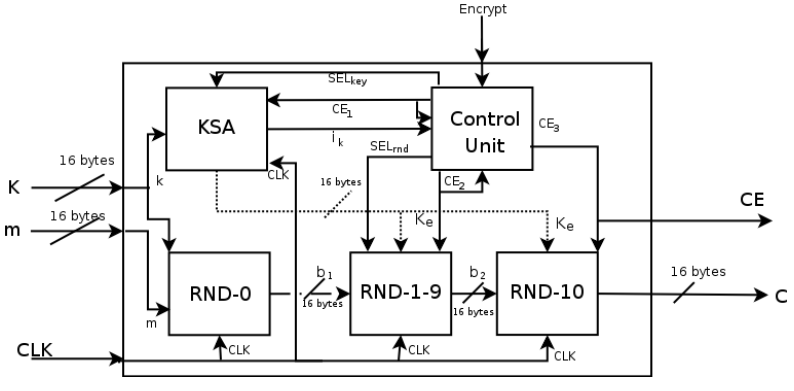


Fig. 8. General Architecture of an AES Encryptor Core

AES Rounds Implementation

- **The Initial Round ARK:** ARK step implementation is quite easy, as it only requires bitwise XOR operation.
- **Round 1 to 9.** We implemented AES main nine rounds in an iterative way. Therefore, only one round was designed, and that one was replicated eight more times. Figure 9 shows the block diagram of this process. This circuit uses a multiplexor to select if we are going to process the first round or the other eight ones. At the end of the circuit we use a Latch to store the current computed state matrix. As shown in Figure 9, in order to improve performance the new state matrix is fed back before the latch.

  Byte Substitution and Shift Rows Steps Implementations. As it is shown in Figure 9, rounds 1 to 9, were implemented using two main building

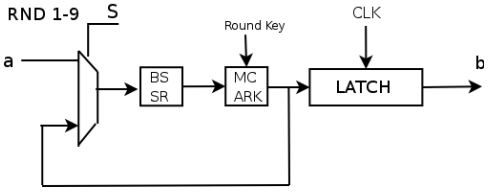


Fig. 9. Block Diagram of the rounds 1 to 9

blocks. The first one is the BS/SR block that was instrumented by using the BRAMs (BlockRAMs) embedded in the targeted FPGA device. Sixteen 8×256 BRAMs were configured for implementing AES S-BOX as a look-up table. By doing so, we are able to compute 16 byte substitutions at the same time.

Mix Columns and AddRoundKey Steps Implementation. Mix Columns and AddRoundKey Steps can be implemented jointly by doing some modifications. In fact, the implementation of the MC and ARK steps is very simple as it only uses XOR gates. For polynomial multiplication the $xtime(v)$ look up table method described in [4] was used . The table is implemented using an array of 16 8×256 BRAMs. By consulting that table, we can compute $02 \times v$ in a fast way.

– **Implementation of the Last Round:** AES final round was implemented using separated resources, so another sixteen 8×256 BRAMs were necessary for implementing the BS step.

Key Scheduling Implementation. In [3] several optimizations based on redundant computation for parallelizing the Key Scheduling process were implemented. As a result, it takes us two steps to compute a new round key [3].

AES Control Unit. The AES Control Unit synchronizes the whole process and controls the information flow. In addition, it produces the signals to control the multiplexors and latches that are used in the AES components. These synchronization signals are crucial because each component should select the correct state matrix.

The signal generated to control the Final Round Latch is also used as an indicator that the ciphertext is ready. This is done by a change in the CE output of the AES block. When the plaintext is being processed, the CE output value is '0', but when a ciphertext is ready, this value changes to '1'. In addition, the AES block has an extra input called "Encrypt" that indicates to the Control Unit that a new plaintext is given and that a new process has to begin. This control signal must be high by one single CLK's cycle and after that it must be set to low.

5 Results

In this section we discuss the results obtained by our design. Moreover, we compare our results with other similar implementations found on open literature.

All implementations reported in this paper were realized on a 3s4000fg900-4 Spartan 3 device using VHDL Language and Xilinx's development tools. We used the Xilinx ISE 6.3i, and the Model Sim Xilinx Edition II v5.8c to simulate and synthesize our design.

Table 1 summarizes the hardware resources required by design's main building blocks. As it can be seen, our design is efficient in terms of hardware resources. It

Table 1. Results of our Design

Block	Slices	BRAMs
Authentication	1031	53
Encryption	713	53
Control Unit and extra Hw	410	0
CCM-Mode	2154	106

Table 2. Comparasion Table of different CCM implementations

Product	Company	Architecture	Throughput(Mbps)
HTRU Aerolink	Ntru	VHDL/Software	0.25
SCAES-CCM	SiWorks	RTL, ASIC, FPGAs	582 at 100MHz 192 at 33 MHz
AES-CCM core	Helion	FPGAs and ASIC	Tiny Version 15, Standard 200
AES-CCM	Our design	VHDL - FPGA	1,051

Table 3. AES Comparasion

Author	Device	Mode	Slices(BRAMs)	Throughput(Mbps)	T/A
Charot et al. [5]	Altera APEX	CTR	N/A	512	N/A
Weaver et al. [6]	XVE600-8	ECB	460(10)	690	1.5
Labbé et al. [7]	XCV1000-4	ECB	2151(4)	390	0.18
Saggese et al. [8]	XCVE2000-8	ECB	446(10)	1000	2.3
Chodwiec et al. [9]	XC2530-5	ECB	222(3)	139	0.62
Chodwiec et al. [9]	XC2530-6	ECB	222(3)	166	0.74
Standaert et al. [10]	VIRTEX2300E	ECB	542(10)	1450	2.6
Gaj et al. [11]	XCV1000	ECB	2902	331.5	0.11
Saqib [3]	XCV812E	ECB	2744	258.5	0.09
Amphion CS5220 [12]	XVE-8	ECB	421(4)	290	0.69
Amphion CS5230 [12]	XVE-8	ECB	573(10)	1060	1.9
Segredo et al. [13]	XCV-100-4	ECB	496(10)	417	0.84
Segredo et al. [13]	XCV600E-8	ECB	496(10)	743	1.49
Calder et al. [14]	Altera EPF10K	ECB	1584	637.24	0.40
Our Design	Spartan 3 3s4000	ECB	633(53)	1067	1.68
Our Design	Spartan 3 3s4000	CBC	1031(53)	1067	1.03
Our Design	Spartan 3 3s4000	CTR	731(53)	1067	1.45

only occupies 2154 slices, but it requires 106 BRAMS, mainly because of the AES module which uses BRAMs arrays for implementing some of the Steps in the Encryption Process. The Throughput achieved is high and therefore allows us to process information in an efficient way. Throughput was computed as follows.

As it was mentioned before, we considered that the additional information a consisted of a fixed two 16-bytes blocks. The maximum length of the plaintext is 1024 bytes which results in 64 blocks of 16 bytes each. Then, we have to compute 67 effective blocks (we add the B_0 block, but this is an overhead in the process, so it was omitted from the throughput computation). The processing of a total of 67 blocks can be accomplished by our design in 804 clock cycles, (each block is computed in 12 cycles). Since design's achieved clock frequency is of about $100.08MHz$ ($9.992ns$) we can compute the throughput as:

$$Throughput = \frac{66 * 128bits}{804 * 9.992ns} = 1.05158Gbits/s$$

5.1 Result Comparison

There exist few reported implementations of CCM mode of operation. In Table 2 we present some commercial implementations from which our design showed the fastest throughput. A more comprehensive comparison can be obtained by comparing reported AES designs implemented using an iterative architecture in different modes. It makes sense to do such a comparison since an overwhelming part of the CCM Mode processing is computed by the AES block. As it can be seen in Table 3, our design is the second fastest throughput for iterative AES architectures. Furthermore, our design is economical yielding the fourth best throughput/area coefficient among compared designs. Notice that the performance achieved by our design was obtained in a Xilinx Spartan FPGA which is technologically inferior to the powerful Xilinx Virtex FPGAs used in most designs featured in Table 3.

6 Conclusions

In this paper an end-to-end design of the CCM Mode using AES as a block cipher was presented. Taking advantage of the potential of reconfigurable hardware platforms for obtaining high parallelism we were able to obtain a parallel version of the CCM mode, which yielded competitive results in terms of both, time and area performances.

The AES in CCM mode is going to be a crucial piece for the next generation of wireless network cards, so it is worth to study it so that faster and more efficient implementations can be achieved. As many of future CCM mode designs are going to be Hardware based, we believe that competitive implementations of this mode in reconfigurable hardware platforms have an enormous potential that should not be neglected.

References

1. Stubblefield Adam, John Ioannidis, and Aviel D. Rubin. Using the Fluhrer, Mantin, and Shamir Attack to Break WEP. Technical report, ATT Labs TD-4ZCPZZ, Available at: http://www.cs.rice.edu/~astubble/wep., August 2001.
2. Doug Whiting, Russ Housley, Niels Ferguson. Counter with CBC-MAC (CCM). In *Submission to NIST*, 2002.
3. F. Rodriguez-Henriquez, N. A. Saqib and A. D. Diaz-Perez. 4.2 Gbit/s Single-Chip FPGA Implementation of AES Algorithm. In *IEE Electronic Letters*, volume 39 (15), pages 1115–1116, July 2003.
4. Joan Daemen, Vincent Rijmen. *The Design of Rijndael: AES The Advanced Encryption Standard*. Springer-Verlag, First edition, 2002.
5. Francois Charot, Eslam Yahya and Charles Wagner. Efficient Modular-Pipelined AES Implementation in Counter Mode on ALTERA FPGA. In *Lecture Notes in Computer Science*, volume 2778, pages 282–291, January 2003.
6. Nicholas Weaver and John Wawrzynek. High Performance, Compact AES implementations in Xilinx FPGAs. Technical report, U.C. Berkeley BRASS group, available at http://www.cs.berkeley.edu/ nnweaver/sfra/rijndael.pdf, 2002.
7. A. Labbé, A. Pérez. AES Implementations on FPGA: Time Flexibility Tradeoff.
8. G.P. Saggese, A. Mazzeo, N. Mazzocca and A.G.M. Strollo. An FPGA-Based Performance Analysis of the Unrolling, Tiling, and Pipelining of the AES Algorithm. In *Lecture Notes in Computer Science*, volume 2778, pages 292–302, January 2003.
9. Pawel Chodowiec and Kris Gaj. Very Compact FPGA Implementation of the AES Algorithm. In *Cryptographic Hardware and Embedded Systems-CHES 2003*, pages 319–333, 2003.
10. Francois-Xavier Standaert, Gael Rouvroy, Jean-Jacques Quisquart and Jean-Didier Legat. Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware: Improvements and Design Tradeoffs. In *Cryptographic Hardware and Embedded Systems-CHES 2003*, pages 334–350, 2003.
11. Kris Gaj and Pawel Chodowiec. Comparison of the hardware performance of the AES candidates using reconfigurable hardware. In *The Third AES Candidate Conference, New York*, 2000.
12. Amphion Semiconductor, available at: http://www.amphion.com/cs5210.html. *CS5210-40: High Performance AES Encryption Cores*, 2003.
13. Alejandro Segredo, Enrique Zabala and Gustavo Bello. Diseño de un procesador criptográfico Rijndael en FPGA. In *X Workshop IBERCHIP*, pages 64–65, 2004.
14. Germán Jácome-Calderon, Jaime Velasco-Medina, Julio López Hernández. Implementación en Hardware del algoritmo Rijndael (in spanish). In *X Workshop IBERCHIP*, pages 113–114, 2004.

New Architecture for Multiplication in $GF(2^m)$ and Comparisons with Normal and Polynomial Basis Multipliers for Elliptic Curve Cryptography*

Soonhak Kwon[1], Taekyoung Kwon[2], and Young-Ho Park[3]

[1] Inst. of Basic Science and Dept. of Mathematics, Sungkyunkwan University, Suwon 440-746, Korea
shkwon@skku.edu
[2] School of Computer Engineering, Sejong University, Seoul 143-747, Korea
tkwon@sejong.ac.kr
[3] Dept. of Information Security, Sejong Cyber University, Seoul 143-747, Korea
youngho@cybersejong.ac.kr

Abstract. We propose a new linear multiplier which is comparable to linear polynomial basis multipliers in terms of the area and time complexity. Also we give a very detailed comparison of our multiplier with the normal and polynomial basis multipliers for the five binary fields $GF(2^m), m = 163, 233, 283, 409, 571$, recommended by NIST for elliptic curve digital signature algorithm.

Keywords: linear multiplier, NIST recommended binary fields, elliptic curve cryptography.

1 Introduction

For every cryptographic application, efficient arithmetic of finite field is a critical factor for a fast implementation of cryptographic hardware or software. Finite field multiplication is one of the most basic arithmetic operations and it is used in many cryptographic algorithms such as ECC or AES. Though one may design a finite field multiplier in a software implementation, a hardware arrangement has a strong advantage when one wants a high speed multiplier. Moreover arithmetic of $GF(2^m)$ is easily realized in a circuit design using a few logical gates. A good multiplication algorithm depends on the choice of basis for a given finite field.

There are three major types of basis being used, which are polynomial, dual, and normal basis. For many applications, a polynomial basis [18,19] is usually preferred over a normal or a dual basis because of its simple field arithmetic and

* This work was supported by grant No. R01-2005-000-11261-0 from Korea Science and Engineering Foundation in Ministry of Science & Technology.

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 335–351, 2006.
© Springer-Verlag Berlin Heidelberg 2006

flexibility of the many available algorithms. Moreover the multiplication algorithms with LSB (least significant bit) first scheme or MSB (most significant bit) first scheme give straightforward realization of a hardware architecture [18] for a linear multiplier over $GF(2^m)$. It should be mentioned that, for cryptographic purposes, one should use a linear multiplier because a two dimensional (or bit parallel) multiplier is impractical in current state of technology due to the huge amount of gate counts. Massy-Omura multiplier is also one of the most popular multipliers these days because of simple squaring operation. Since the original suggestion of Massey and Omura [1], many new normal basis multipliers have been proposed [2,4,5,13,15]. There is another type of multiplier called Berlekamp dual basis bit serial multiplier [7]. Though considerable improvements have been made [3,8,9,10,11] on this multiplier, it does not seem to get much attention these days for cryptographic purposes because of long critical path delay and inconvenient basis conversion process. The multipliers in [3,7,10] do not consider basis conversion and therefore have different bases for input and output values, which make it complicated for practical applications in cryptography.

Our aim in this paper is to show that a suitably modified multiplier from [7] is excellent also for cryptographic purposes. Our proposed multiplier uses single basis consistently for input and output values and has a sequential structure (i.e. parallel in parallel out), which are our salient features that distinguish from previous results in [7,10]. Consequently our multiplier has a significantly reduced critical path delay and a low area complexity which are superior or comparable to those of normal or polynomial basis multipliers. We will give a very detailed comparison of our multiplier with other proposed multipliers for the five binary fields $GF(2^m), m = 163, 233, 283, 409, 571$, recommended by NIST for ECDSA (elliptic curve digital signature algorithm).

2 Review of Previous Works

Let $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$ be a basis for $GF(2^m)$, which is a finite field with characteristic two. We will now briefly discuss Berlekamp's idea [7] on bit serial multiplication. We will give a modified version in [8,9] for generality and clarity.

Definition 1. *We say that two bases* $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$ *and* $\{\beta_0, \beta_1, \cdots, \beta_{m-1}\}$ *of* $GF(2^m)$ *are dual to each other if the trace map,* $Tr : GF(2^m) \to GF(2)$ *with* $Tr(\alpha) = \alpha + \alpha^2 + \cdots + \alpha^{2^{m-1}}$, *satisfies* $Tr(\alpha_i \beta_j) = \delta_{ij}$ *for all* $0 \le i, j \le m-1$, *where* $\delta_{ij} = 1$ *if* $i = j$, *and zero if* $i \ne j$.

It is easy [6] to see that a unique dual basis exists for a given basis. Also if $\{\beta_0, \beta_1, \cdots, \beta_{m-1}\}$ is the dual basis of $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$, then for any nonzero $\beta \in GF(2^m)$, $\{\beta^{-1}\beta_0, \beta^{-1}\beta_1, \cdots, \beta^{-1}\beta_{m-1}\}$ is dual to $\{\beta\alpha_0, \beta\alpha_1, \cdots, \beta\alpha_{m-1}\}$. Let $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$ be a polynomial basis for $GF(2^m)$ and let

$$f(X) = f_0 + f_1 X + f_2 X^2 + \cdots + f_{m-1} X^{m-1} + X^m \in GF(2)[X] \qquad (1)$$

be the unique irreducible polynomial of α over $GF(2)$. Then the dual basis of the polynomial basis is well known and it is expressed as follows [6].

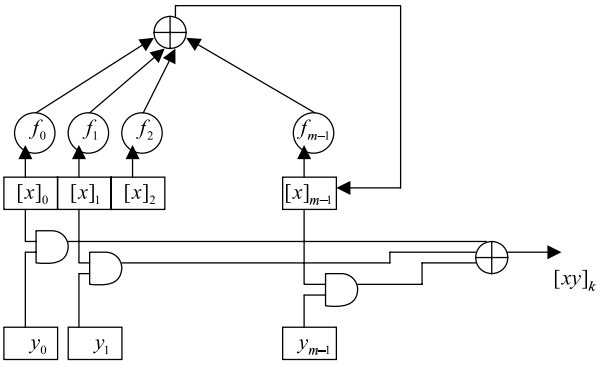


Fig. 1. A bit serial dual basis multiplier of Berlekamp

Lemma 2. *Let* $f(X) = (X - \alpha)(g_0 + g_1 X + \cdots + g_{m-1}X^{m-1})$ *where* $g_i, 0 \leq i \leq m-1$, *are in* $GF(2^m)$. *Then the dual basis of* $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$ *is* $\left\{\frac{g_0}{f'(\alpha)}, \frac{g_1}{f'(\alpha)}, \cdots, \frac{g_{m-1}}{f'(\alpha)}\right\}$.

Let $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$ be the dual basis of $\{\beta, \beta\alpha, \beta\alpha^2, \cdots, \beta\alpha^{m-1}\}$ where $\beta \neq 0 \in GF(2^m)$. For given $x, y \in GF(2^m)$, we may express x and the product xy with respect to the basis $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$ and we write y with respect to the basis $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$,

$$x = \sum_{i=0}^{m-1} [x]_i \gamma_i,\ y = \sum_{i=0}^{m-1} y_i \alpha^i,\ \ xy = \sum_{k=0}^{m-1} [xy]_k \gamma_k. \tag{2}$$

Using the duality between $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$ and $\{\beta, \beta\alpha, \beta\alpha^2, \cdots, \beta\alpha^{m-1}\}$, we have

$$[xy]_k = Tr(\beta\alpha^k xy) = \sum_{i=0}^{m-1} y_i Tr(\beta\alpha^{i+k}x) = \sum_{i=0}^{m-1} y_i [\alpha^k x]_i, \tag{3}$$

where $[\alpha^k x]_i$ is the ith coefficient of the expression of $\alpha^k x$ with respect to the basis $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$. Moreover we have

$$[\alpha^k x]_i = Tr(\beta\alpha^i\alpha^k x) = Tr(\beta\alpha^{i+1}\alpha^{k-1}x) = [\alpha^{k-1}x]_{i+1}, \quad 0 \leq i \leq m-2. \tag{4}$$

Also since $f(X) = X^m + \sum_{i=0}^{m-1} f_i X^i$ is the irreducible polynomial of α over $GF(2)$,

$$[\alpha^k x]_{m-1} = Tr(\beta\alpha^{m-1}\alpha^k x) = Tr(\beta \sum_{i=0}^{m-1} f_i \alpha^i \alpha^{k-1} x) = \sum_{i=0}^{m-1} f_i [\alpha^{k-1}x]_i. \tag{5}$$

Therefore the coefficients of $x, \alpha x, \alpha^2 x, \cdots, \alpha^{m-1}x$ with respect to the basis $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$ are recursively computed by the above relations (4) and (5)

and the multiplication $[xy]_k = \sum_{i=0}^{m-1} y_i[\alpha^k x]_i$ is realized in the shift register arrangement shown in Fig. 1. After k clock cycles, we get $[xy]_k$, the kth coefficient of xy with respect to $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$ which is the dual basis of $\{\beta, \beta\alpha, \beta\alpha^2, \cdots, \beta\alpha^{m-1}\}$.

3 New Linear Multiplier with Reduced Critical Path Delay and Lower Hardware Complexity

The dual basis bit serial multiplier discussed in previous section has two serious drawbacks. First it needs a basis conversion which requires extra circuitry. Second it has a long critical path delay which is rather unsuitable for cryptographic purposes, especially for elliptic curve cryptography where one should choose m large enough so that $m \geq 163$ [12]. We will show that these two problems are in fact minor and can be easily modified. The first problem can be solved using the method of Stinson [11], which is a generalization of the basis conversion using trinomials $X^m + X^k + 1$ [8] and special types of pentanomials $X^m + X^{k+1} + X^k + X^{k-1} + 1$ [9]. Our idea is to stick to the basis $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$ which is dual to $\{\beta, \beta\alpha, \beta\alpha^2, \cdots, \beta\alpha^{m-1}\}$ for some β so that the basis conversion costs only few XOR gates but no extra flip-flop. The number of necessary XOR gates depends on the weight of the given polynomial $f(X)$ and the cost is negligible from a cryptographic point of view. For example, this method can be applied to all five binary fields $GF(2^m)$ with $m = 163, 233, 283, 409, 571$, recommended by NIST [12] for elliptic curve cryptography. The suggested polynomials in [12] are either trinomials ($m = 233, 409$) or pentanomials ($m = 163, 283, 571$) and our method is easily applicable to these polynomials. The problem of long critical path delay can also be solved. By rearranging the summation of the coefficients of xy with respect to the basis $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$, we will derive a linear multiplier for ECC whose critical path delay is comparable to polynomial basis multipliers and is shorter than previously known linear normal basis multipliers for the five fields $GF(2^m)$ with $m = 163, 233, 283, 409, 571$.

3.1 Techniques of Basis Conversion

Let $f(X) = 1 + X^{n_1} + X^{n_2} + \cdots + X^{n_t} + X^m$ be the irreducible polynomial of $\alpha \in GF(2^m)$ with $0 = n_0 < n_1 < n_2 < \cdots < n_t < m$. Let $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$ be the dual basis of $\{\beta, \beta\alpha, \beta\alpha^2, \cdots, \beta\alpha^{m-1}\}$ for some $\beta \in GF(2^m)$. Recall that, from the expression of $f(X)$ in Lemma 2,

$$\begin{aligned} f(X) &= (X - \alpha) \sum_{i=0}^{m-1} g_i X^i = \sum_{i=0}^{m-1} g_i X^{i+1} - \sum_{i=0}^{m-1} \alpha g_i X^i \\ &= \sum_{i=0}^{m-2} g_i X^{i+1} - \sum_{i=0}^{m-2} \alpha g_{i+1} X^{i+1} + g_{m-1} X^m - \alpha g_0 \end{aligned} \tag{6}$$

$$= \sum_{i=0}^{m-2} (g_i - \alpha g_{i+1}) X^{i+1} + g_{m-1} X^m - \alpha g_0.$$

From the above equations, it is not difficult to see [11] that

$$g_i = \alpha^{-i-1} \sum_{n_j \le i} \alpha^{n_j} = \alpha^{m-i-1} + \alpha^{-i-1} \sum_{n_j > i} \alpha^{n_j}. \tag{7}$$

This is because

$$g_0 = \alpha^{-1}, \quad g_{m-1} = 1 = \alpha^{-m} \sum_{n_j \le m-1} \alpha^{n_j} = \alpha^{-m}(1 + \alpha^{n_1} + \cdots + \alpha^{n_t}), \tag{8}$$

and

$$g_i - \alpha g_{i+1} = \alpha^{-i-1} \sum_{n_j \le i} \alpha^{n_j} - \alpha\alpha^{-i-2} \sum_{n_j \le i+1} \alpha^{n_j} = \alpha^{-i-1} \sum_{i < n_j \le i+1} \alpha^{n_j}. \tag{9}$$

Thus $g_i - \alpha g_{i+1}$ is 1 if there exists n_j such that $n_j = i+1$ and is zero if there is no such n_j. In other words, $g_i - \alpha g_{i+1}$ is the coefficient of X^{i+1} of the polynomial $f(X) = 1 + X^{n_1} + X^{n_2} + \cdots + X^{n_t} + X^m$.

Since one should choose only one basis for the consistency of the field arithmetic, we will choose the basis $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$ for our multiplication of x and y. Therefore, in view of Fig. 1, we need a basis conversion from $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$ to $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$ to express y in terms of $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$ from our initial choice of the basis $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$. Let

$$\gamma_i = \sum_{j=0}^{m-1} c_{ij} \alpha^j \tag{10}$$

for all $0 \le i \le m-1$ and let $y = \sum_{i=0}^{m-1} y_i \alpha^i = \sum_{i=0}^{m-1} [y]_i \gamma_i$ be the expression of y with respect to $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$ and $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$, respectively. Then

$$\begin{aligned} \sum_{i=0}^{m-1} [y]_i \gamma_i &= ([y]_0, [y]_1, \cdots, [y]_{m-1})(\gamma_0, \gamma_1, \cdots, \gamma_{m-1})^T \\ &= ([y]_0, [y]_1, \cdots, [y]_{m-1})(c_{ij})(1, \alpha, \alpha^2, \cdots, \alpha^{m-1})^T \\ &= (y_0, y_1, \cdots, y_{m-1})(1, \alpha, \alpha^2, \cdots, \alpha^{m-1})^T = \sum_{i=0}^{m-1} y_i \alpha^i, \end{aligned} \tag{11}$$

where (c_{ij}) is the m by m matrix defined by the equation (10) and $(\gamma_0, \gamma_1, \cdots, \gamma_{m-1})^T$ (resp. $(1, \alpha, \alpha^2, \cdots, \alpha^{m-1})^T$) is the transposition of the row vector $(\gamma_0, \gamma_1, \cdots, \gamma_{m-1})$ (resp. $(1, \alpha, \alpha^2, \cdots, \alpha^{m-1})$). Thus we have

$$(y_0, y_1, \cdots, y_{m-1}) = ([y]_0, [y]_1, \cdots, [y]_{m-1})(c_{ij}) \tag{12}$$

and (c_{ij}) can be regarded as the basis conversion matrix from $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$ to $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$. We define the excess number of the basis conversion to be 'the number of nonzero entries of the matrix (c_{ij}) minus m'. As long as the excess number is small, we can get a fairly simple basis conversion. For example, if the excess number is *zero*, then the rows of (c_{ij}) are the permutated ones of the identity matrix so that the basis conversion is just a permutation. Note that from Lemma 2, we have the dual basis of $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$ as $\left\{\frac{g_0}{f'(\alpha)}, \frac{g_1}{f'(\alpha)}, \cdots, \frac{g_{m-1}}{f'(\alpha)}\right\}$. Now choose the mysterious constant $\beta \in GF(2^m)$ as $\beta = (\alpha^{n_s} f'(\alpha))^{-1}$ where $s = \frac{t+1}{2}$, i.e. n_s is the power of the exact middle term of the irreducible $f(X) = 1 + X^{n_1} + \cdots + X^{n_s} + \cdots + X^{n_t} + X^m$. Then $\{\alpha^{n_s} g_0, \alpha^{n_s} g_1, \cdots, \alpha^{n_s} g_{m-1}\}$ is the dual basis of $\{\beta, \beta\alpha, \beta\alpha^2, \cdots, \beta\alpha^{m-1}\}$. Using this technique, Stinson [11] clarified and generalized previous results on basis conversion of Wang and Blake [8] using trinomials $X^m + X^k + 1$ and of Morii et al. [9] using special pentanomials $X^m + X^{k+1} + X^k + X^{k-1} + 1$ to the case of general irreducible polynomials $f(X) = X^m + \sum_{i=0}^{t} X^{n_i}$ as follows.

Theorem 3. [11] *Let $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$ be a polynomial basis for $GF(2^m)$ and let $f(X) = X^m + \sum_{i=0}^{t} X^{n_i}$ be the irreducible polynomial of α over $GF(2)$ with $0 = n_0 < n_1 < \cdots < n_t < m$. Write $s = \frac{t+1}{2}, \beta = (\alpha^{n_s} f'(\alpha))^{-1}$ and $f(X) = (X - \alpha)(\sum_{i=0}^{m-1} g_i X^i)$. Then $\{\alpha^{n_s} g_0, \alpha^{n_s} g_1, \cdots, \alpha^{n_s} g_{m-1}\}$ is the dual basis of $\{\beta, \beta\alpha, \beta\alpha^2, \cdots, \beta\alpha^{m-1}\}$ and the excess number of the basis conversion is $\sum_{i=s+1}^{t} n_i - \sum_{i=1}^{s-1} n_i$.*

Sketch of Proof. The duality is clear from Lemma 2. Also from the equation (7), write $\alpha^{n_s} g_i$ as (because this expression produces minimal number of summands since n_s is the power of the exact middle term of $f(X)$.)

$$\begin{aligned} \alpha^{n_s} g_i &= \sum_{n_j \le i} \alpha^{n_s - i - 1 + n_j} && \text{if } i < n_s \\ &= \alpha^{n_s - i - 1 + m} + \sum_{n_j > i} \alpha^{n_s - i - 1 + n_j} && \text{if } i \ge n_s. \end{aligned} \tag{13}$$

Then in all cases we have

$$0 \le n_s - i - 1 + n_j \le m - 1 \tag{14}$$

and a counting argument [11] on the number of summands of the equations in (13) shows that the number of nonzero entries of the basis conversion matrix is $m + \sum_{i=s+1}^{t} n_i - \sum_{i=1}^{s-1} n_i$, i.e. the excess number is $\sum_{i=s+1}^{t} n_i - \sum_{i=1}^{s-1} n_i$. □

For example, as is showed in [8], when $f(X) = X^m + X^k + 1$ is an irreducible trinomial, then the excess number is *zero* and the basis conversion from $\{\alpha^{n_s} g_0, \alpha^{n_s} g_1, \cdots, \alpha^{n_s} g_{m-1}\}$ to $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$ is just a permutation. Thus no extra circuitry is needed in this case. Also when irreducible $f(X)$ is the following special pentanomial $X^m + X^{k+1} + X^k + X^{k-1} + 1$, then we have $t = 3, s = \frac{t+1}{2} = 2$ and thus the excess number is $2 = (k+1) - (k-1)$ which was

already shown in [9]. However for a general irreducible polynomial, the excess number may not be small and in that case, for each y_i which is a linear combination of the signals $[y]_0, [y]_1, \cdots, [y]_{m-1}$, the depth and the number of necessary XOR gates of the XOR tree with respect to y_i should be determined precisely.

Lemma 4. *Let $\gamma_i = \alpha^{n_s} g_i$ for all $0 \le i \le m-1$. Let $y = \sum_{i=0}^{m-1} [y]_i \gamma_i = \sum_{i=0}^{m-1} y_i \alpha^i$ be the expression of y with respect to the bases $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$ and $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$, respectively. Then for each $0 \le i \le m-1$, y_i is the sum of at most s elements of $[y]_j$ with $0 \le j \le m-1$. Thus each coefficient y_i is obtained by using an XOR tree of depth at most $\lceil \log_2 s \rceil$ with at most $s-1$ XOR gates. Also the total number of necessary XOR gates to generate $y_0, y_1, \cdots, y_{m-1}$ using the signals $[y]_0, [y]_1, \cdots, [y]_{m-1}$ is exactly $\sum_{i=s+1}^{t} n_i - \sum_{i=1}^{s-1} n_i$.*

Proof. By Theorem 3 and the equation (7), it is clear that $\sum_{i=s+1}^{t} n_i - \sum_{i=1}^{s-1} n_i$ is the total number of necessary XOR gates to generate all $y_0, y_1, \cdots, y_{m-1}$ using the signals $[y]_0, [y]_1, \cdots, [y]_{m-1}$. Also, since

$$\begin{aligned} y &= \sum_{k=0}^{m-1} y_k \alpha^k = \sum_{i=0}^{m-1} [y]_i \gamma_i = \sum_{i=0}^{m-1} [y]_i \alpha^{n_s} g_i \\ &= \sum_{i=0}^{n_s-1} [y]_i \Big(\sum_{n_j \le i} \alpha^{n_s - i - 1 + n_j} \Big) + \sum_{i=n_s}^{m-1} [y]_i \Big(\alpha^{n_s - i - 1 + m} + \sum_{n_j > i} \alpha^{n_s - i - 1 + n_j} \Big), \end{aligned} \tag{15}$$

we find that, for a fixed $0 \le k \le m-1$, $[y]_i$ appears as a summand of y_k if there is n_j with $n_j \le i < n_s$ or $n_j > i \ge n_s$ such that $k = n_s - i - 1 + n_j$ or if $k = n_s - i - 1 + m$ with $i \ge n_s$. Suppose that there exists a pair n_j, i with $n_j \le i < n_s$ such that $k = n_s - i - 1 + n_j$. Then for any pair $n_{j'}, i'$ with $n_{j'} > i' \ge n_s$, we have $k \ne n_s - i' - 1 + n_{j'}$ because if this is an equality then we have $0 \ge n_j - i = n_{j'} - i' > 0$ which is a contradiction. Also the number of pairs $n_{j'}, i'$ with $n_{j'} \le i' < n_s$ satisfying $k = n_s - i' - 1 + n_{j'}$ is at most s because j' is in the range $0 \le j' < s$ and i' is uniquely determined by the condition $n_j - i = n_{j'} - i'$. Thus y_k is the sum of at most s terms of $[y]_{i'}$ with $0 \le i' \le m-1$. The other case when there exists a pair n_j, i with $n_j > i \ge n_s$ such that $k = n_s - i - 1 + n_j$ can be dealt in the same manner. Finally, suppose that there is $i \ge n_s$ such that $k = n_s - i - 1 + m$. If there is another $i' \ge n_s$ such that $k = n_s - i' - 1 + m$, then from $n_s - i - 1 + m = k = n_s - i' - 1 + m$, we have $i = i'$. Note that there is no pair $n_{j'}, i'$ with $n_{j'} \le i' < n_s$ such that $k = n_s - i' - 1 + n_{j'}$, because this implies $0 < m - i = n_{j'} - i' \le 0$ which is a contradiction. Also the number of the pairs $n_{j'}, i'$ with $n_{j'} > i' \ge n_s$ satisfying $k = n_s - i' - 1 + n_{j'}$ is at most $s-1$ because i' is uniquely determined by the condition $m - i = n_{j'} - i'$ and j' is in the range $s < j' \le t$, i.e. the number of possible j' is $t - s = s - 1$. Counting the pair m, i, we have at most s terms of $[y]_{i'}$ which appears as a summand of y_k. □

Because of Lemma 4 and Theorem 3, we can now precisely describe the basis conversion using XOR trees. The necessary XOR gates for each tree can also

be determined easily. For example, when one has an irreducible pentanomial $f(X) = X^m + X^{n_3} + X^{n_2} + X^{n_1} + 1$ with $0 = n_0 < n_1 < n_2 < n_3 < m$, the total number of necessary XOR gates for the basis conversion is $n_3 - n_1$ and each XOR tree, if it exists, is just a single XOR gate with depth one because $t = 3$ and $s = \frac{t+1}{2} = 2$. That is, among all coefficients $y_0, y_1, \cdots, y_{m-1}$ of $y = \sum_{i=0}^{m-1} y_i\alpha^i = \sum_{i=0}^{m-1}[y]_i\gamma_i$, exactly $n_3 - n_1$ number of y_i are the sum of two elements from the set $\{[y]_0, [y]_1, \cdots, [y]_{m-1}\}$ and exactly $m - n_3 + n_1$ number of y_i are same to some members in $\{[y]_0, [y]_1, \cdots, [y]_{m-1}\}$.

3.2 Reducing the Critical Path Delay

Let us now explain the second problem of reducing the critical path delay of Berlekamp multiplier. Since a bit serial architecture usually has a long critical path delay, we want to modify the architecture to the sequential architecture using the symmetry of the multiplication table $Tr(\beta\alpha^{i+k}x)$ in (3). As usual, let $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\}$ be the dual basis of $\{\beta, \beta\alpha, \beta\alpha^2, \cdots, \beta\alpha^{m-1}\}$ with $\beta = (\alpha^{n_s} f'(\alpha))^{-1}$. Write $x, y \in GF(2^m)$ as $x = \sum_{i=0}^{m-1}[x]_i\gamma_i$ and $y = \sum_{i=0}^{m-1} y_i\alpha^i$. Letting $xy = \sum_{k=0}^{m-1}[xy]_k\gamma_k$, the equation (3) says that $[xy]_k = \sum_{i=0}^{m-1} y_i Tr(\beta\alpha^{i+k}x)$. That is,

$$\begin{aligned}
[xy]_0 &= y_0Tr(\beta x) + y_1Tr(\beta\alpha x) + \cdots + y_{m-1}Tr(\beta\alpha^{m-1}x),\\
[xy]_1 &= y_0Tr(\beta\alpha x) + y_1Tr(\beta\alpha^2 x) + \cdots + y_{m-1}Tr(\beta\alpha^m x),\\
&\cdots\\
&\cdots\\
[xy]_{m-1} &= y_0Tr(\beta\alpha^{m-1}x) + y_1Tr(\beta\alpha^m x) + \cdots + y_{m-1}Tr(\beta\alpha^{2m-2}x).
\end{aligned} \tag{16}$$

By defining the column vectors $Y = (y_0, y_1, \cdots, y_{m-1})^T, Z = ([xy]_0, [xy]_1, \cdots, [xy]_{m-1})^T$, we have $Z = \mathcal{A}Y$ where the m by m matrix $\mathcal{A} = (a_{ij})$ is defined as $a_{ij} = Tr(\beta\alpha^{i+j}x), 0 \le i, j \le m-1$. The crucial property of the matrix $\mathcal{A}$ is that it is symmetric. Note that in the bit serial construction of Berlekamp, each row vector of $\mathcal{A}$ is computed by a feedback shift register using the equations (4,5). Since $\mathcal{A}$ is symmetric, the column vectors of $\mathcal{A}$ are generated by the same shift register. Therefore we may compute the product xy sequentially. In other words, letting $A_j = (Tr(\beta\alpha^j x), Tr(\beta\alpha^{j+1}x), \cdots, Tr(\beta\alpha^{j+m-1}x))^T$ be the jth column vector of $\mathcal{A}$ with $0 \le j \le m-1$, we compute the multiplication as follows,

$$Z = (\cdots(((A_0y_0) + A_1y_1) + A_2y_2) + \cdots) + A_{m-1}y_{m-1}. \tag{17}$$

Note that at the jth clock cycle ($0 \le j \le m-1$), A_j is multiplied to the constant y_j and the value A_jy_j is added to the partial sum $A_0y_0 + \cdots + A_{j-1}y_{j-1}$ to get the result $A_0y_0 + \cdots + A_jy_j$ which is stored in the register D_i, $0 \le i \le m-1$, for a partial summation. At the same time, the upper shift register is loaded with the vector A_{j+1} via the relations (4,5) and the lower shift register is loaded with the vector $(y_{j+1}, y_{j+2}, \cdots, y_j)$ via left cyclic shifting, and the loaded values wait for the next cycle.

3.3 Main Result with Hardware Architecture

All these explanations on the critical path delay and the basis conversion are realized in the circuit arrangement shown in Fig. 2. In the figure, every input of our multiplier is expressed with respect to the basis $\{\gamma_0, \gamma_1, \cdots, \gamma_{m-1}\} = \{\alpha^{n_s} g_0, \cdots, \alpha^{n_s} g_{m-1}\}$. The input $y = \sum_{i=0}^{m-1} [y]_i \gamma_i$ is loaded via XOR trees to the lower shift register so that the vector in the register has the value $(y_0, \cdots, y_{m-1})$, and this vector is cyclically shifted to the left by one position at every cycle. The area complexity and the critical path delay of the proposed multiplier are explained in our main Theorem 5.

Theorem 5. *Let $f(X) = X^m + \sum_{i=0}^{t} X^{n_i}$ be an irreducible polynomial over $GF(2)$ with $0 = n_0 < n_1 < \cdots < n_t < m$ and let α be a zero of $f(X)$. Write $s = \frac{t+1}{2}$ and $f(X) = (X - \alpha)(\sum_{i=0}^{m-1} g_i X^i)$. Then, by using the basis $\{\alpha^{n_s} g_0, \alpha^{n_s} g_1, \cdots, \alpha^{n_s} g_{m-1}\}$, we can construct a sequential multiplier for the multiplication of x and y in $GF(2^m)$ with $3m$ flip-flops, m AND gates, and $m + t + \sum_{i=s+1}^{t} n_i - \sum_{i=1}^{s-1} n_i$ XOR gates such that*

1. *An XOR tree for the feed back shift register for the input x needs t XOR gates with depth of the tree $\lceil \log_2(t+1) \rceil$. Also the basis conversion of the input y costs $\sum_{i=s+1}^{t} n_i - \sum_{i=1}^{s-1} n_i$ XOR gates with m XOR trees such that each XOR tree consists of at most $\frac{t-1}{2}$ XOR gates with depth $\lceil \log_2(t+1) \rceil - 1$.*
2. *Our multiplier has a parallel-in parallel-out structure and produces an output of xy at a rate of one every m clock cycle. The critical path delay of our multiplier is $T_A + T_X$ if $t = 1$ (i.e. trinomial case) and it is $\lceil \log_2(t+1) \rceil T_X$ if $t > 1$, where T_A and T_X are the delay time of a two input AND gate and a two input XOR gate, respectively.*

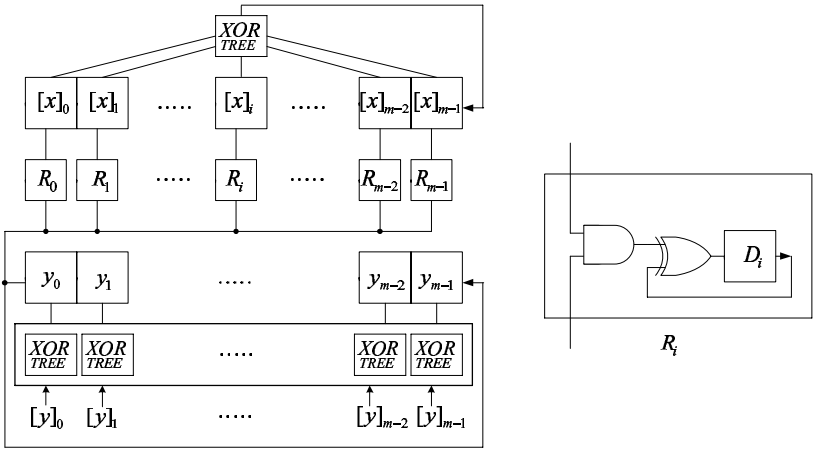


Fig. 2. A new linear multiplier using the basis $\{\alpha^{n_s} g_0, \cdots, \alpha^{n_s} g_{m-1}\}$ in $GF(2^m)$ with $s = \frac{t+1}{2}$ and $f(X) = X^m + \sum_{i=0}^{t} X^{n_i} = (X - \alpha)(\sum_{i=0}^{m-1} g_i X^i)$

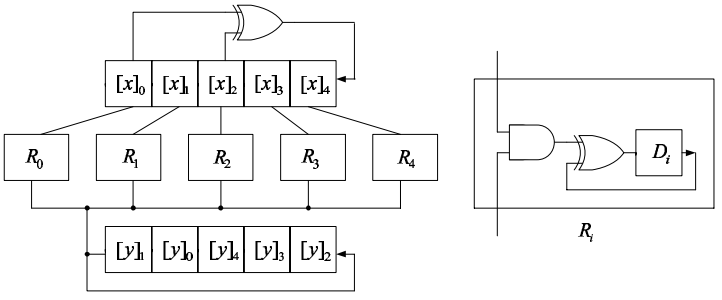


Fig. 3. A multiplication circuit using the basis $\{\alpha^2 g_0, \cdots, \alpha^2 g_4\}$ in $GF(2^5)$ with $f(X) = X^5 + X^2 + 1 = (X - \alpha)(\sum_{i=0}^{4} g_i X^i)$

Example 1. For a simple example, let us explain our multiplier for the case $m = 5$. Fig. 3 shows the circuit of our multiplier using $f(X) = X^5 + X^2 + 1$. Since $f(X)$ is a trinomial, the basis conversion of y is just a permutation. Note that our circuit in this case is quite similar to the circuit with a polynomial basis [18].

4 Comparison with Other Linear Normal and Polynomial Basis Multipliers for Elliptic Curve Cryptography

In view of the result in Theorem 5, we may now claim that our proposed multiplier is, in terms of the time and area complexity, superior to most of normal basis multipliers currently used. The lowest complexity normal basis multipliers which are cryptographically meaningful are the multipliers using type II ONB (optimal normal basis). Previously known normal basis multipliers using type II ONB have the critical path delay either $T_A + 2T_X$ [2,5,13] or $T_A + 3T_X$ [4]. On the other hand, Theorem 5 says that the critical path delay of our multiplier is $T_A + T_X$ when $f(X)$ is a trinomial and $2T_X$ when $f(X)$ is a pentanomial. Thus our multiplier using either trinomial or pentanomial has a shorter critical path delay than the normal basis multipliers [2,4,5,13] using type II ONB. Moreover type II ONB are pretty rare [6] compared with trinomial or pentanomial bases [16,17]. For example, all the five binary fields $GF(2^m), m = 163, 233, 283, 409, 571$, recommended by NIST [12] have either irreducible trinomials (when $m = 233, 409$) or pentanomials (when $m = 163, 283, 571$) while there is only one field $GF(2^{233})$ which has a type II ONB.

If one looks for the next possible low complexity normal basis after type II ONB, it is generally believed that a low complexity normal basis is a Gaussian normal basis of type k for some k. A Gaussian normal basis of type k in $GF(2^m)$ is a normal basis $\{\alpha, \alpha^2, \cdots, \alpha^{2^{m-1}}\}$ with

$$\alpha = \sum_{j=0}^{k-1} \beta^{\tau^j}, \tag{18}$$

where β is a primitive pth root of unity in $GF(2^{mk})$ with $p = mk+1$ a prime, and τ is an element of order k in $GF(p)^{\times}$. It is well known [6] that such Gaussian normal basis exists if and only if $gcd(mk/ord_p 2, m) = 1$. Note that a type II ONB is a Gaussian normal basis of type $k = 2$. NIST [12] suggested that, when one uses a normal basis, one uses the lowest complexity Gaussian normal basis for each binary field. That is, a Gaussian normal basis of type 2 when $m = 233$, type 4 when $m = 163, 409$, type 6 when $m = 283$, and type 10 when $m = 571$. To the authors' knowledge, there is no known normal basis (which is not Gaussian) whose complexity is lower than the recommended (by NIST) Gaussian normal basis for each field at this moment. Therefore it seems that the corresponding Gaussian normal basis for each $GF(2^m), m = 163, 233, 283, 409, 571$, is the lowest complexity normal basis for that field.

Table 1. Comparison with previously proposed linear multipliers

	Basis	Critical path delay	AND	XOR
[1]	normal	$T_A + \lceil \log_2 (mk) \rceil T_X$	C	$C - 1$
[2, 13*]	normal	$T_A + (1 + \lceil \log_2 k \rceil) T_X$	m	C
[4]	normal	$T_A + (1 + \lceil \log_2 (k+2) \rceil) T_X$	m	$\frac{1}{2}(C+1) + \lfloor \frac{m}{2} \rfloor$
[5]	normal	$T_A + (1 + \lceil \log_2 k \rceil) T_X$	m	$\frac{k}{2}(m-1) + \frac{m+1}{2}$
[7, 10**]	dual	$T_A + \lceil \log_2 m \rceil T_X$	m	$m + t - 1$
[18, 20]	polynomial (LSB)	$T_A + T_X$	m	$m + t$
[18]	polynomial (MSB)	$T_A + 2T_X$	m	$m + t$
This paper	$\{\alpha^{n_s} g_i \mid 0 \le i \le m-1\}$	$T_A + T_X$ or $\lceil \log_2 (t+1) \rceil T_X$	m	$m + t + \sum_{i=s+1}^{t} n_i - \sum_{i=1}^{s-1} n_i$

Remark: In the table, C is defined as $C = km-k+1$ if k is *even* and $C = (k+1)m-3k+2$ if k is *odd*. One has the lowest possible $C = 2m - 1$ when $k = 1$ or 2. Also, *the paper [13] is applicable only for the case of type II ONB, and **[10] considers only the basis conversion with regard to $X^m + X^k + 1$ and $X^m + X^{k+1} + X^k + X^{k-1} + 1$ and examples are given only for the values $m \le 10$.

Compared with normal basis multipliers, there are not so many kinds of linear polynomial basis multipliers partly because of its relatively simple field arithmetic. There are two standard ways of computing multiplications in $GF(2^m)$. One is LSB (least significant bit) first scheme and the other is MSB (most significant bit) first scheme. Namely, LSB first scheme computes the least significant bit of the operand first and MSB first scheme computes the most significant bit of the operand first. One can easily construct a linear multiplier [18,20] following either LSB or MSB first scheme and the resulting circuits have almost the same area complexity and critical path delay.

Remark : NIST [12] recommends to use the lowest complexity Gaussian normal basis for the above fields and they are of type $4, 2, 6, 4, 10$, respectively. For irreducible polynomials, NIST recommends to use $X^{163} + X^7 + X^6 + X^3 + 1, X^{233} + X^{74} + 1, X^{283} + X^{12} + X^7 + X^5 + 1, X^{409} + X^{87} + 1, X^{571} + X^{10} + X^5 + X^2 + 1$, respectively.

Table 2. Comparison of the critical path delay for the five binary fields

$GF(2^m)$ type k	$GF(2^{163})$ $k=4$	$GF(2^{233})$ $k=2$	$GF(2^{283})$ $k=6$	$GF(2^{409})$ $k=4$	$GF(2^{571})$ $k=10$
[1]	$T_A + 10T_X$	$T_A + 9T_X$	$T_A + 11T_X$	$T_A + 11T_X$	$T_A + 13T_X$
[2, 13]	$T_A + 3T_X$	$T_A + 2T_X$	$T_A + 4T_X$	$T_A + 3T_X$	$T_A + 5T_X$
[4]	$T_A + 4T_X$	$T_A + 3T_X$	$T_A + 4T_X$	$T_A + 4T_X$	$T_A + 5T_X$
[5]	$T_A + 3T_X$	$T_A + 2T_X$	$T_A + 4T_X$	$T_A + 3T_X$	$T_A + 5T_X$
[7, 10]	$T_A + 8T_X$	$T_A + 8T_X$	$T_A + 9T_X$	$T_A + 9T_X$	$T_A + 10T_X$
[18, 20] LSB	$T_A + T_X$	$T_A + T_X$	$T_A + T_X$	$T_A + T_X$	$T_A + T_X$
[18] MSB	$T_A + 2T_X$	$T_A + 2T_X$	$T_A + 2T_X$	$T_A + 2T_X$	$T_A + 2T_X$
This paper	$2T_X$	$T_A + T_X$	$2T_X$	$T_A + T_X$	$2T_X$

From Table 1, the critical path delay of [2,5] using a Gaussian normal basis of type k is $T_A + (1 + \lceil \log_2 k \rceil)T_X$. Also the critical path delay of the polynomial basis multipliers is $T_A + T_X$ with LSB first scheme and is $T_A + 2T_X$ with MSB first scheme. On the other hand, our multiplier has a critical path delay $T_A + T_X$ if $f(X)$ is a trinomial and $2T_X$ if $f(X)$ is a pentanomial. This implies that the critical path delay of our multiplier is significantly reduced from that of normal basis multipliers and is comparable to that of polynomial basis multipliers. See the difference of the critical path delay for $m = 163, 233, 283, 409, 571$ shown in Table 2. It should be mentioned that, since we are dealing with a linear multiplier, even a small increment of the critical path delay such as T_X results in a total delay of mT_X where m is the size of a field.

Table 3. Comparison of the number of necessary XOR gates for the five binary fields

$GF(2^m)$ type k	$GF(2^{163})$ $k=4$	$GF(2^{233})$ $k=2$	$GF(2^{283})$ $k=6$	$GF(2^{409})$ $k=4$	$GF(2^{571})$ $k=10$
[1]	648	464	1692	1632	5700
[2, 13]	649	465	1693	1633	5701
[4]	406	349	988	1021	3136
[5]	406	349	988	1021	3136
[7, 10]	165	233	285	409	573
[18, 20] LSB	166	234	286	410	574
[18] MSB	166	234	286	410	574
This paper	170	234	293	410	582

The area complexity of our multiplier is also far lower than that of Massey-Omura multipliers [1,2,4,5] and is comparable to that of polynomial basis multipliers. For a binary field $GF(2^m)$ with odd m (i.e. with even k), Table 1,3 shows that the normal basis multipliers in [4,5] need $\frac{C+1}{2} + \lfloor \frac{m}{2} \rfloor = \frac{k+1}{2}m - \frac{k-1}{2}$ XOR gates, which is $\frac{3}{2}m - \frac{1}{2}$ for a type II ONB and $\frac{k+1}{2}m - \frac{k-1}{2} > \frac{5}{2}m - 5$ for other Gaussian normal bases of type $k = 4, 6, 10$. On the other hand, our multiplier needs $m + 1$ XOR gates when trinomial is used and needs $m + 3 + n_3 - n_1$ ($< 2m + 3$) XOR gates when pentanomial is used. Note that the difference of

the number of necessary XOR gates between our multiplier and the polynomial basis multipliers comes from the base change of our multiplier and is practically negligible.

5 Explicit Circuits of Our Multiplier for the NIST Recommended Binary Fields

Let us explain how to construct a multiplication circuit for each binary field $GF(2^m)$, $m = 163, 233, 283, 409, 571$, using Theorem 5. The recommended irreducible polynomials [12] for these fields are $X^{163} + X^7 + X^6 + X^3 + 1, X^{233} + X^{74} + 1, X^{283} + X^{12} + X^7 + X^5 + 1, X^{409} + X^{87} + 1, X^{571} + X^{10} + X^5 + X^2 + 1$, respectively. Since these polynomials are either trinomials or pentanomials, we will discuss basis conversion of these polynomials and derive explicit multiplication architectures. First let us consider the trinomial $f(X) = X^m + X^k + 1$. Then we get the exact middle term $\alpha^{n_s} = \alpha^k$ and from the equations (7) or (13), we have

$$\begin{aligned} \alpha^k g_i = \alpha^{k-i-1} \sum_{n_j \le i} \alpha^{n_j} &= \alpha^{k-i-1} && \text{if } 0 \le i < k \\ &= \alpha^{k-i-1}(1+\alpha^k) = \alpha^{k-i-1+m} && \text{if } k \le i < m. \end{aligned} \tag{19}$$

Therefore the basis of our multiplier for the trinomial $f(X) = X^m + X^k + 1$ is

$$\{\alpha^k g_0, \cdots, \alpha^k g_{m-1}\} = \{\alpha^{k-1}, \alpha^{k-2}, \cdots, \alpha^0, \alpha^{m-1}, \alpha^{m-2}, \cdots, \alpha^k\}. \tag{20}$$

Next let us consider the pentanomial $f(X) = X^m + X^{n_3} + X^{n_2} + X^{n_1} + 1$. Then $\alpha^{n_s} = \alpha^{n_2}$ and again using the equations (7) or (13),

$$\begin{aligned} \alpha^{n_2} g_i &= \alpha^{n_2-i-1} && \text{if } 0 \le i < n_1 \\ &= \alpha^{n_2-i-1}(1+\alpha^{n_1}) = \alpha^{n_2-i-1} + \alpha^{n_2-i-1+n_1} && \text{if } n_1 \le i < n_2 \\ &= \alpha^{n_2-i-1}(\alpha^{n_3} + \alpha^m) = \alpha^{n_2-i-1+n_3} + \alpha^{n_2-i-1+m} && \text{if } n_2 \le i < n_3 \\ &= \alpha^{n_2-i-1}\alpha^m = \alpha^{n_2-i-1+m} && \text{if } n_3 \le i < m. \end{aligned} \tag{21}$$

Thus the basis $\{\gamma_0, \cdots, \gamma_{m-1}\} = \{\alpha^{n_2} g_0, \cdots, \alpha^{n_2} g_{m-1}\}$ of our multiplier for the pentanomial $f(X) = X^m + X^{n_3} + X^{n_2} + X^{n_1} + 1$ is

$$\begin{aligned} \{&\alpha^{n_2-1}, \cdots, \alpha^{n_2-n_1}, \alpha^{n_2-n_1-1} + \alpha^{n_2-1}, \cdots, \alpha^0 + \alpha^{n_1}, \\ &\alpha^{n_3-1} + \alpha^{m-1}, \cdots, \alpha^{n_2} + \alpha^{n_2-n_3+m}, \alpha^{n_2-n_3+m-1}, \cdots, \alpha^{n_2}\}. \end{aligned} \tag{22}$$

Example 2. Multiplication in $GF(2^{163})$ using $f(X) = X^{163} + X^7 + X^6 + X^3 + 1$: From the equations (21,22), we have the following basis $\{\gamma_0, \cdots, \gamma_{162}\}$ for the field $GF(2^{163})$,

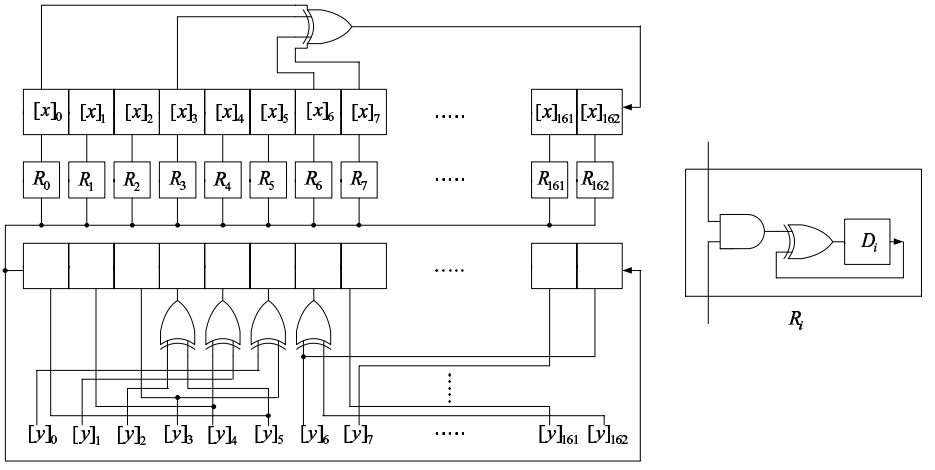


Fig. 4. A multiplication circuit using the basis $\{\gamma_0, \cdots, \gamma_{m-1}\} = \{\alpha^6 g_0, \cdots, \alpha^6 g_{162}\}$ in $GF(2^{163})$ with $f(X) = X^{163} + X^7 + X^6 + X^3 + 1 = (X - \alpha)(\sum_{i=0}^{162} g_i X^i)$

$$\{\alpha^5, \alpha^4, \alpha^3, \alpha^2 + \alpha^5, \alpha + \alpha^4, \alpha^0 + \alpha^3, \alpha^6 + \alpha^{162}, \alpha^{161}, \alpha^{160}, \cdots, \alpha^6\}. \quad (23)$$

Therefore letting $y = \sum_{i=0}^{162} [y]_i \gamma_i = \sum_{i=0}^{162} y_i \alpha^i$, the basis conversion is

$$\begin{aligned} y &= [y]_0\alpha^5 + [y]_1\alpha^4 + [y]_2\alpha^3 + [y]_3(\alpha^2 + \alpha^5) + [y]_4(\alpha + \alpha^4) + [y]_5(\alpha^0 + \alpha^3) \\ &\quad + [y]_6(\alpha^6 + \alpha^{162}) + [y]_7\alpha^{161} + [y]_8\alpha^{160} + \cdots + [y]_{162}\alpha^6 \\ &= [y]_5\alpha^0 + [y]_4\alpha + [y]_3\alpha^2 + ([y]_2 + [y]_5)\alpha^3 + ([y]_1 + [y]_4)\alpha^4 + ([y]_0 + [y]_3)\alpha^5 \\ &\quad + ([y]_6 + [y]_{162})\alpha^6 + [y]_{161}\alpha^7 + [y]_{160}\alpha^8 + \cdots + [y]_6\alpha^{162}, \end{aligned}$$

and the corresponding circuit for the multiplication is shown in Fig. 4.

Example 3. Multiplication in $GF(2^{233})$ using $f(X) = X^{233} + X^{74} + 1$: From the equation (20), we have the following basis $\{\gamma_0, \cdots, \gamma_{232}\}$ for the field $GF(2^{233})$,

$$\{\alpha^{73}, \alpha^{72}, \cdots, \alpha^0, \alpha^{232}, \alpha^{231}, \cdots, \alpha^{74}\}. \quad (24)$$

Thus the multiplication is easily realized in the shift register arrangement shown in Fig. 5. Note that one can construct a similar circuit for the case $GF(2^{409})$ with NIST recommended trinomial $f(X) = X^{409} + X^{87} + 1$.

Example 4. Multiplication in $GF(2^{283})$ using $f(X) = X^{283} + X^{12} + X^7 + X^5 + 1$: From the equations (21,22), we have the following basis $\{\gamma_0, \cdots, \gamma_{282}\}$ for the field $GF(2^{283})$,

$$\{\alpha^6, \cdots, \alpha^2, \alpha + \alpha^6, \alpha^0 + \alpha^5, \alpha^{11} + \alpha^{282}, \cdots, \alpha^7 + \alpha^{278}, \alpha^{277}, \cdots, \alpha^7\}. \quad (25)$$

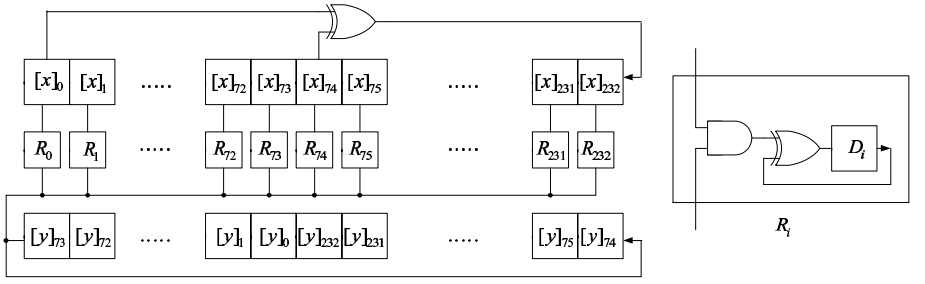


Fig. 5. A multiplication circuit using the basis $\{\alpha^{74}g_0, \cdots, \alpha^{74}g_{232}\}$ in $GF(2^{233})$ with $f(X) = X^{233} + X^{74} + 1 = (X - \alpha)(\sum_{i=0}^{232} g_i X^i)$

Thus we have

$$\begin{aligned} y &= [y]_0\alpha^6 + \cdots + [y]_4\alpha^2 + [y]_5(\alpha + \alpha^6) + [y]_6(\alpha^0 + \alpha^5) \\ &\quad + [y]_7(\alpha^{11} + \alpha^{282}) + \cdots + [y]_{11}(\alpha^7 + \alpha^{278}) + [y]_{12}\alpha^{277} + \cdots + [y]_{282}\alpha^7 \\ &= [y]_6\alpha^0 + \cdots + [y]_2\alpha^4 + ([y]_1 + [y]_6)\alpha^5 + ([y]_0 + [y]_5)\alpha^6 \\ &\quad + ([y]_{11} + [y]_{282})\alpha^7 + \cdots + ([y]_7 + [y]_{278})\alpha^{11} + [y]_{277}\alpha^{12} + \cdots + [y]_7\alpha^{282}, \end{aligned}$$

and the multiplication circuit is shown in Fig. 6, where the omitted cell R_i is same to the cell in Fig. 2. Also note that the other field $GF(2^{507})$ with NIST recommended $f(X) = X^{571} + X^{10} + X^5 + X^2 + 1$ can be dealt in the same manner.

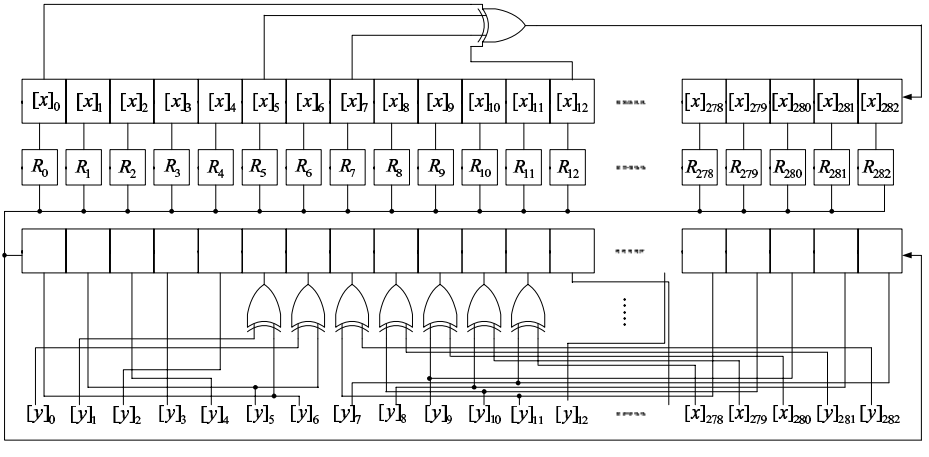


Fig. 6. A multiplication circuit using the basis $\{\alpha^7 g_0, \cdots, \alpha^7 g_{282}\}$ in $GF(2^{283})$ with $f(X) = X^{283} + X^{12} + X^7 + X^5 + 1 = (X - \alpha)(\sum_{i=0}^{282} g_i X^i)$

6 Conclusions

In this paper, we proposed a new linear multiplier and constructed explicit multiplication circuits for the five NIST recommended binary fields $GF(2^m), m = 163, 233, 283,$
$409, 571$, for ECC purpose. Our new linear multiplier has a reduced critical path delay and a simple basis conversion process compared with the dual basis multipliers in [7,10]. Compared with linear normal basis multipliers [2,4,5,13], our multiplier has a significantly reduced critical path delay and a lower hardware complexity as is explained in Table 1,2,3. Compared with linear polynomial basis multipliers [18,20] with LSB or MSB first scheme, the same tables also show that our proposed multiplier has almost the same hardware complexity and critical path delay. Although a normal basis has a simple squaring with no extra cost, it is not difficult to show that a squaring in our basis (or in a polynomial basis) needs a negligible cost as long as we choose a low weight irreducible polynomial such as a trinomial or a pentanomial. Computational evidence [16,17] shows that there are plenty of such low weight polynomials compared with low complexity normal bases. Therefore our proposed multiplier can be used in many cryptographic applications where a low cost and a high speed arithmetic unit is needed.

References

1. J.L. Massy and J.K. Omura, "Computational method and apparatus for finite field arithmetic," *US Patent No. 4587627*, 1986.
2. G.B. Agnew, R.C. Mullin, I. Onyszchuk, and S.A. Vanstone, "An implementation for a fast public key cryptosystem," *J. Cryptology*, Vol. 3, pp. 63–79, 1991.
3. H. Wu, M.A. Hasan, and I.F. Blake, "New low complexity bit-parallel finite field multipliers using weakly dual bases," *IEEE Trans. Computers*, Vol. 47, pp. 1223–1234, 1998.
4. A. Reyhani-Masoleh and M.A. Hasan, "Low complexity sequential normal basis multipliers over $GF(2^m)$," *16th IEEE Symposium on Computer Arithmetic*, Vol. 16, pp. 188–195, 2003.
5. S. Kwon, K. Gaj, C. Kim, and C. Hong, "Efficient linear array for multiplication in $GF(2^m)$ using a normal basis for elliptic curve cryptography," *CHES 2004, Lecture Notes in Computer Science*, vol. 3156, pp. 76–91, 2004.
6. A.J. Menezes, I.F. Blake, S. Gao, R.C. Mullin, S.A. Vanstone, and T. Yaghoobian, *Applications of Finite Fields*, Kluwer Academic Publisher, 1993.
7. E.R. Berlekamp, "Bit-serial Reed-Solomon encoders," *IEEE Trans. Inform. Theory*, Vol. 28, pp. 869–874, 1982.
8. M. Wang and I.F. Blake, "Bit serial multiplication in finite fields," *SIAM J. Disc. Math.*, Vol. 3, pp. 140–148, 1990.
9. M. Morii, M. Kasahara and D.L. Whiting, "Efficient bit-serial multiplication and the discrete-time Wiener-Hopf equation over finite fields," *IEEE Trans. Inform. Theory*, Vol. 35, pp. 1177–1183, 1989.
10. S.T.J. Fenn, M. Benaissa, and D. Taylor, "$GF(2^m)$ multiplication and division over the dual basis," *IEEE Trans. Computers*, Vol. 45, pp. 319–327, 1996.

11. D.R. Stinson, "On bit-serial multiplication and dual bases in $GF(2^m)$," *IEEE Trans. Inform. Theory*, Vol. 37, pp. 1733–1736, 1991.
12. NIST, "Digital Signature Standard," *FIPS Publication*, 186-2, February, 2000.
13. H. Wu, M.A. Hasan, I.F. Blake, and S. Gao, "Finite field multiplier using redundant representation," *IEEE Trans. Computers*, Vol. 51, pp. 1306–1316, 2002.
14. S. Feisel, J. von zur Gathen, M. Shokrollahi, "Normal bases via general Gauss periods," *Math. Comp.*, Vol. 68, pp. 271–290, 1999.
15. B. Sunar and Ç.K. Koç, "An efficient optimal normal basis type II multiplier," *IEEE Trans. Computers*, Vol. 50, pp. 83–87, 2001.
16. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handboook of Applied Cryptography*, CRC Press, 1996.
17. M. Zivkovic, "Table of primitive binary polynomials II," *Math. Comp.*, Vol. 63, pp. 301–306, 1994.
18. D. Hankerson, A.J. Menezes, and S.A. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer–Verlag, 2004.
19. D. Hankerson, J.L. Hernandez, and A.J. Menezes, "Software implementation of elliptic curve cryptography over binary fields," *CHES 2000, LNCS*, Vol. 1965, pp. 1–24 , 2000.
20. L. Song and K.K. Parhi, "Efficient finite field serial/parallel multiplication," *International Conference on Application Specific Systems, Architectures and Processors*, pp. 19–21, 1996.

An Efficient Design of CCMP for Robust Security Network*

Duhyun Bae, Gwanyeon Kim, Jiho Kim, Sehyun Park, and Ohyoung Song

School of Electrical and Electronic Engineering, Chung-Ang University,
221, HukSuk-Dong, DongJak-Gu, Seoul, Korea
{duhyunbae, cityhero, jihokim}@wm.cau.ac.kr,
{shpark, song}@cau.ac.kr

Abstract. For high data rate, new mechanisms such as Block Ack and frame aggregation are currently being discussed in IEEE 802.11e and IEEE 802.11n, respectively. These mechanisms need a short response time in each MPDU processing. In this paper, we propose an efficient design of CCMP for IEEE 802.11i to support these new MAC mechanisms. The proposed design adopts the mode toggling approach, in which MIC calculation and data encryption are sequentially performed for each 128 bits of the packet in only one AES-CCM core. In our design, the response time is reduced to a short constant period, which takes only 44 clock cycles. In addition, we can reduce hardware complexity and power consumption, because our design uses one AES-CCM core and obtains the reasonable data throughput and response time at even low clock frequency. We have implemented the proposed design, which is targeted to Altera Stratix FPGA device. As a result of the experiments, the CCMP features 285 Mbps data throughput and 0.88 μs response time at 50 MHz frequency.

1 Introduction

When the IEEE 802.11 standard was published, it included an optional security protocol called WEP. However, as the IEEE and Wi-Fi Alliance realized that WEP is not safe against attacks, IEEE 802.11 task group I presented the RSN (Robust Security Network) architecture that uses WEP (Wired Equivalent Privacy), TKIP (Temporal Key Integrity Protocol), and CCMP (Counter with CBC-MAC Protocol) to protect WLAN (Wireless Local Area Network) traffic in MAC (Medium Access Control) layer [1][2].

In CCMP, AES-CCM (Advanced Encryption Standard Counter with CBC-MAC) block cipher algorithm [2] is used to provide the data privacy. In order to guarantee crypto acceleration and satisfy the MAC layer data throughput, secure operation, and interoperability, CCMP operation should be embedded in WLAN device [3].

* This research was supported by the MIC (Ministry of Information and Communication), Korea, under the Chung-Ang University HNRC (Home Network Research Center)-ITRC support program supervised by the IITA (Institute of Information Technology Assessment).

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 352 – 361, 2006.
© Springer-Verlag Berlin Heidelberg 2006

On the other hand, the drive for high data rate and QoS support in wireless LANs has pushed the IEEE to develop IEEE 802.11n and IEEE 802.11e. For high data rate, new MAC mechanisms, such as Block Ack in IEEE 802.11e and frame aggregation in IEEE 802.11n, are currently being discussed in other IEEE 802.11 task group [4-6]. Because these aforementioned mechanisms have a short interval among MPDUs, the response time in a cipher core should be short. In this paper, we define the response time as an interval between the first data input time and the first processed data output time, where each data is of 128 bits. We may reduce the response time by increasing a clock frequency in the cipher core. But power consumption is a critical factor especially to wireless communication device of mobile terminal. As a clock frequency become faster, dynamic power consumption in circuit will increase. If we adopt faster clock frequency to reduce the response time, power consumption in the cipher core may increase. Our proposed design is motivated by the need to reduce the response time even without increasing the clock frequency.

Some works have been presented on hardware implementation of the AES algorithm. In [7-9], most of them are based on pipelining, sub-pipelining and loop unrolling to speed up the AES algorithm in non-feedback modes by duplicating hardware for implementing each round in order to increase data throughput and get the cost to be effective. But they required more logics than feedback modes, because of using pipeline registers in each round and duplicating hardware. In WLAN security, high data throughput is not needed because the MAC data processing speed is 11 Mbps in IEEE 802.11b and 54Mbps in IEEE 802.11a and IEEE 802.11g. So, it is important to make the cipher core for IEEE 802.11 fast in speed and simple architecture enough to still stay within performance constraints to reduce power consumption in wireless communication device.

AES-CCM algorithm consists of two processes: One is MIC calculation with CBC-MAC (Cipher Block Chaining-Message Authentication Code) and the other is data encryption with counter mode. In AES-CCM, almost the same structure can be used to both MIC (Message Integrity Code) calculation and data encryption [10]. So, in general, there are two methods in designing CCMP core in hardware: a sequential method and a parallel method. In the sequential method, its MIC calculation completes the MIC data for the overall payload and then the payload and the MIC data are encrypted in the same AES module. In [3], CCMP was implemented in this way. In CCMP encapsulation process, the AES module calculates the MIC data in CBC-MAC mode and then performs data encryption in counter mode. In the method, it can lead to a significant savings in code and hardware size because it uses only one AES module. However, its response time is directly proportional to the payload size. In the parallel one, it computes the MIC data in an AES module and performs encryption simultaneously in another one. It can reduce the total computing time to almost a half and the response time to short constant interval. However, it uses logic gates two times more than the sequential structure.

In this paper, we present an efficient hardware design for the CCMP. Instead of sequential and parallel structure approaches, we implement the CCMP core that calculates the MIC data and performs data encryption by turn. The proposed design is much more cost-effective than the parallel structure and its response time takes only 44 clock cycles in encapsulation and decapsulation processes.

2 Design Considerations

In the IEEE 802.11 task group E, the Block Acknowledgement (Block Ack) mechanism is currently being discussed to reduce the acknowledgement overhead. The message sequence chart for Block Ack mechanism in the 802.11e is illustrated in Fig. 1.

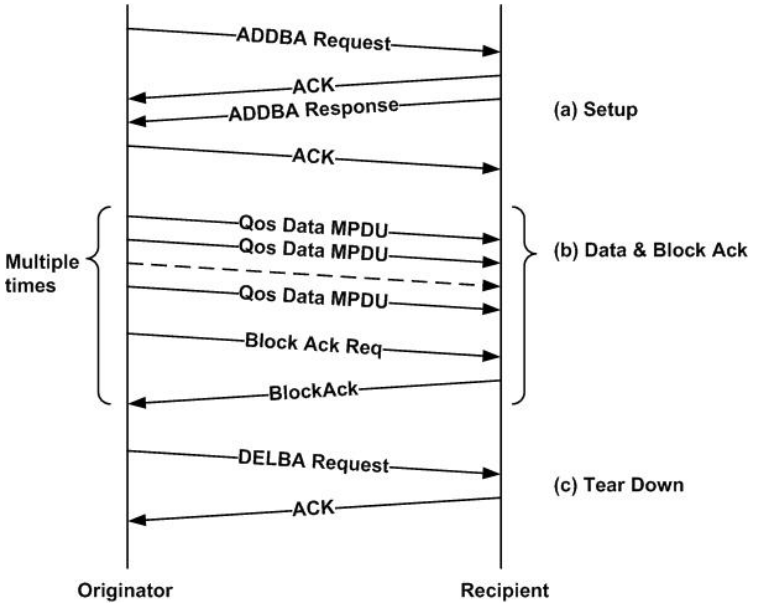


Fig. 1. Message Sequence Chart for Block Ack Mechanism in IEEE 802.11e[4]

The Block Ack mechanism allows a burst of frames to be transmitted before any acknowledgement. After sending the block of frames, the sender sends a block acknowledgement request (Block Ack Req) frame. The receiver sends a block acknowledgement (Block Ack) containing the information about which frames have been correctly received. Each frame is separated by a short interframe space (SIFS) period [4].

In the IEEE 802.11n task group, single and multiple destination frame aggregation is being discussed. Some proposals specify a standardized frame aggregation for both single and multiple destinations to improve network efficiency and interoperability. Frame aggregation mechanism is important for streaming applications including wireless voice over IP and multimedia content [5]. For example, frame aggregation in TGnSync proposal [5] bundles several MAC frames into a single PLCP (Physical Layer Convergence Procedure) frame for transmission. The frame aggregation process in TGnSync proposal is illustrated in Fig. 2. Several MAC frames are put into the same PLCP frame and are separated with an appropriate delimiter between them. In this mechanism, the time interval among MAC frames in an aggregation frame is very short and each frame in the frame aggregation process is separated by a SIFS period.

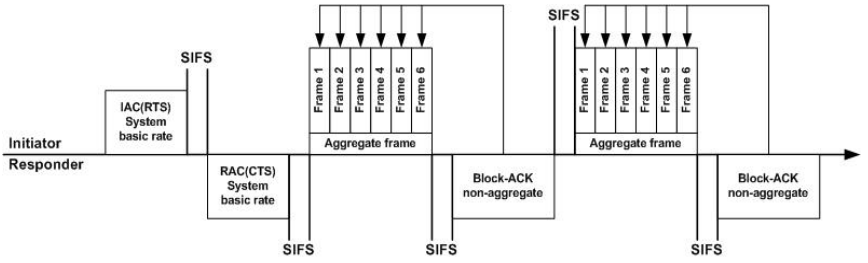


Fig. 2. Frame Aggregation Process in TGnSync Proposal [5]

In 802.11i, the encapsulation process is performed per MPDU. Because the two new mechanisms in IEEE 802.11e and IEEE 802.11n have short interval among MPDUs, the response time in the cipher core must be short. If the sender latency between the end time of a MPDU encapsulation and the start time of the next MPDU encapsulation is not much smaller than SIFS too large to be acceptable in IEEE 802.11e and IEEE 802.11n, neither Block Ack nor frame aggregation can work properly.

The response time may be reduced by increasing a clock frequency in the cipher core. But in wireless device, which are usually resource-constrained, power consumption is a critical factor especially to mobile terminal. In general, as a clock frequency becomes faster, the dynamic power consumption will increase. If we adopt faster clock frequency to reduce the response time, power consumption in the cipher core will increase. In our proposed design, we reduce the response time even without increasing the clock frequency, keeping high data throughput.

We use SIFS as a factor to consider the maximum allowable response time. The total delay in MAC, PHY, and the cipher core must be within the SIFS. Thus, the delay in the cipher core must be much smaller than the SIFS specified in other 802.11 standards. SIFS is 10μs in 802.11b and 16μs in 802.11a and 802.11n. In this paper, we assume the maximum allowable response time in the cipher core to about 3μs.

3 CCMP Protocol and Previous Designs

3.1 CCMP Protocol Overview

CCMP operations are illustrated in Fig. 3 and Fig. 4. CCMP is based upon the CCM mode of the AES encryption algorithm. CCMP utilizes 128-bit keys, with a 48-bit packet number (PN) for replay attack detection. The counter mode of CCMP is the algorithm providing data privacy. The Cipher Block Chaining Message Authentication Code (CBC-MAC) of CCMP provides authentication and data integrity [1].

In CBC-MAC mode, NONCE and Additional Authentication Data (AAD) is calculated firstly and then Framebody is computed. In counter mode, a count value is encrypted by AES core and this result is XORed with 128-bit block data.

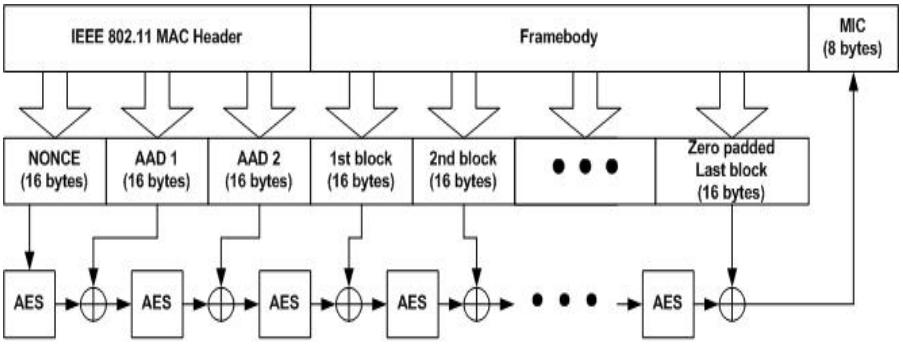


Fig. 3. CCMP MIC data calculation with CBC-MAC

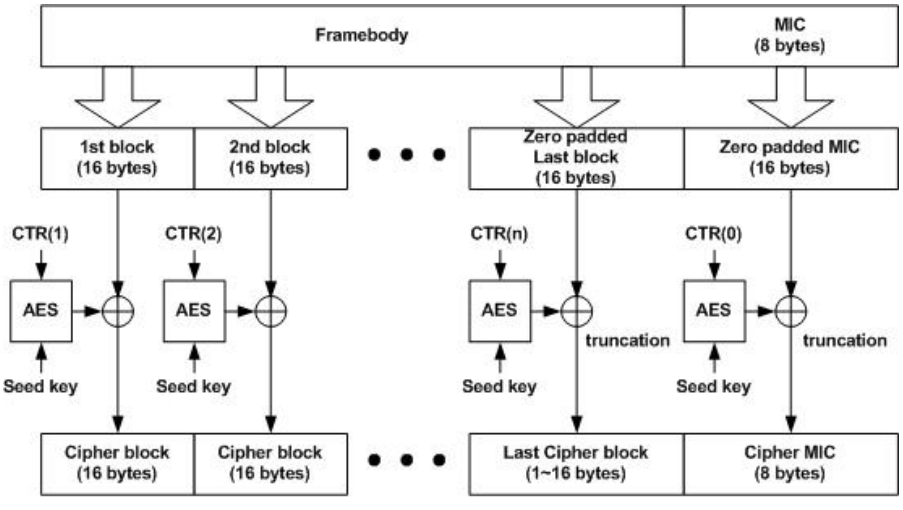


Fig. 4. CCMP data encryption with counter mode

3.2 CCMP Design in Sequential Structure

As mentioned in Section 1, CCMP can be implemented in the sequential structure. Its MIC data are calculated for the overall payload and then the payload and the MIC data are encrypted in the same AES module. The encapsulation timing diagram of a sequential CCMP core is shown in Fig. 5.

In this structure, as a payload size becomes larger, the response time increases linearly. For example, if the payload size is 1024 bytes and AES module takes 11 clock cycles for MIC calculation of 128 bit data, the total MIC calculation will take 748 clock cycles as described in Fig. 5. At 100 MHz clock frequency, it will take 7.48 μs. If 2048 bytes is a payload size, the total MIC calculation will take 14.52 μs, which is larger than SIFS in IEEE 802.11b. Therefore, the sequential structure can not support new MAC mechanisms because of long response time.

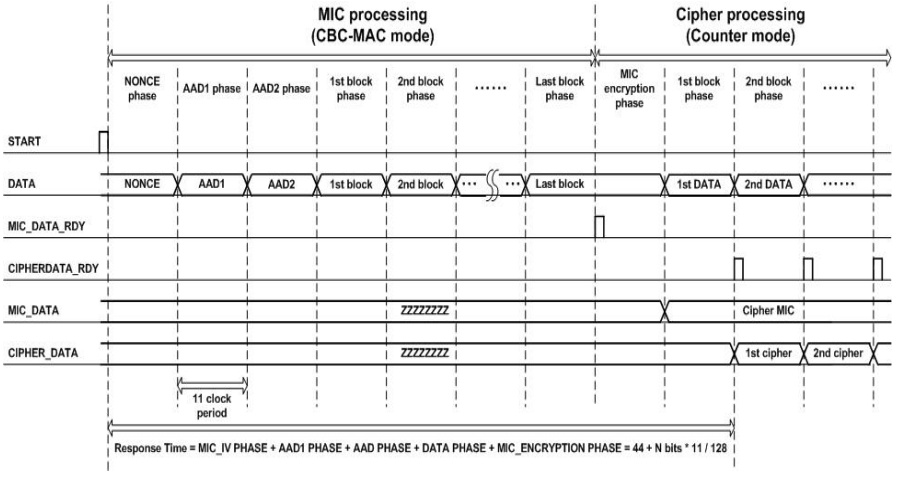


Fig. 5. The encapsulation timing diagram of a sequential CCMP core

3.3 CCMP Design in Parallel Structure

A parallel structure CCMP core uses two AES modules. One AES module is used for MIC calculation in CBC-MAC mode and the other is used for data encryption in counter mode. The encapsulation timing diagram of a parallel CCMP core is illustrated in Fig. 6.

In the parallel structure, it takes only 44 clock cycles to obtain the first 128-bits cipher data and the response time doesn't depend on the payload size. The data

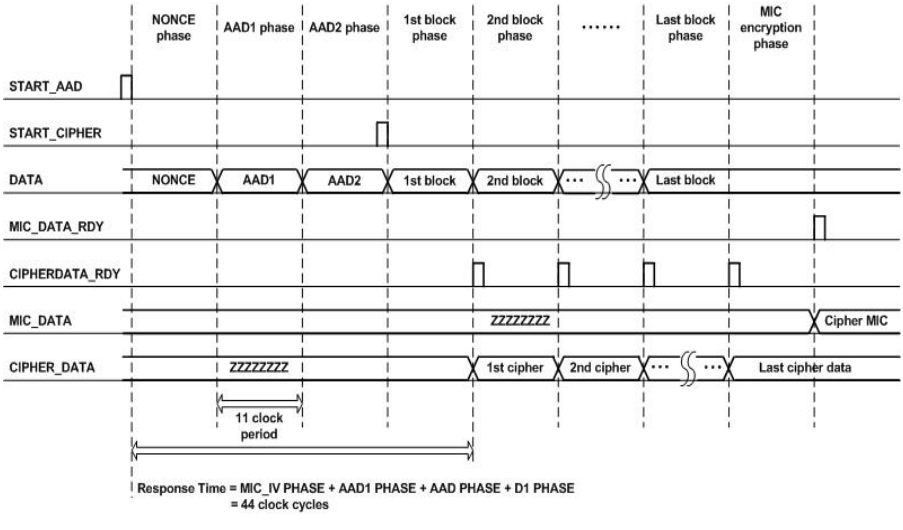


Fig. 6. The encapsulation timing diagram of a parallel CCMP core

throughput is improved nearly 2 times than the CCMP core in the sequential structure. However, the parallel structure uses logic gates as 2 times more than the sequential structure.

4 Efficient Architecture of CCMP

In our proposed design, we calculate the MIC data and perform data encryption with one AES module by controlling input data to an AES module. An CCMP calculates the MIC data in CBC-MAC mode and performs data encryption in counter mode by turn. The encapsulation timing diagram of the CCMP is illustrated in Fig. 7.

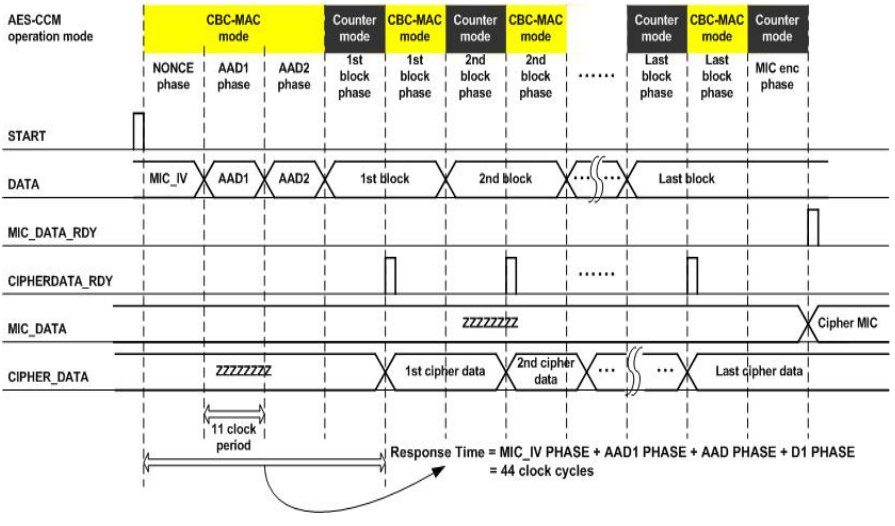


Fig. 7. The encapsulation timing diagram of the CCMP

Firstly, our CCMP core operates in CBC-MAC mode to compute NONCE, Additional Authentication Data 1 (AAD1), and AAD2 during 3 phases. Secondly, it operates in counter mode for data encryption and operates in CBC-MAC mode for MIC calculation by turn. Each phase in our CCMP core takes 11 clock cycles. Thus, it takes only 44 clock cycles to obtain first 128-bits cipher data and it is the same as the response time in the parallel structure. The proposed architecture of CCMP is illustrated in Fig. 8.

It consists of 6 blocks −AES module, Round Key Generator, Construction Block, Counter, Mode Controller, I/O Interface. Construction Block generates NONCE and AAD from PN and MAC header. Round Key Generator generates round keys on the fly. Mode Controller controls the operation mode of the AES module, control signals, and input data to the AES module. Counter block is a simple 16-bits counter and generates proper counter value for each data block.

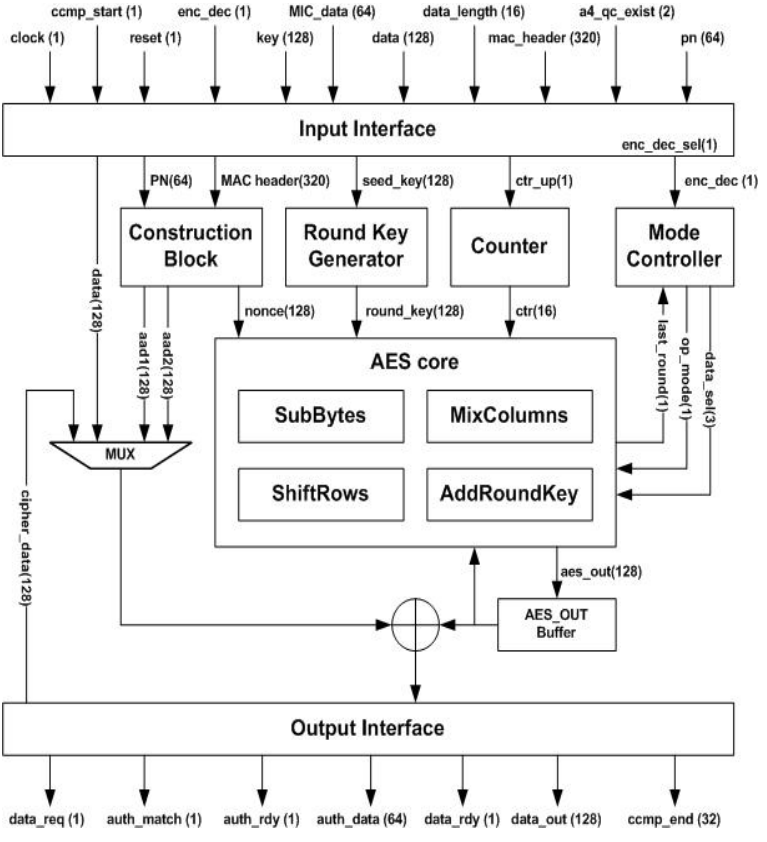


Fig. 8. The architecture of the implemented CCMP

5 Implementation Results and Comparisons

Our architecture was synthesized and implemented using Quartus Ⅱ compiler and targeted to Altera Stratix FPGA (Field Programmable Gate Array) device, which was supported by IDEC (IC Design Education Center).

Table 1 shows clock cycles for each phase in the implemented CCMP. Table 2 shows the performance comparison with other structures. When we obtained

Table 1. Clock cycles for each phase in the implemented CCMP

Category	Cycles
CBC mode encryption	11 cycles
CBC mode MIC encryption	11 cycles
Counter mode encryption	11 cycles

performance results, initialization cycles in CCMP core are not considered because their overheads are small.

In Table 2, we can figure out that the data throughput of our design is the same as that of the sequential structure and the response time has been improved significantly. The response time is not dependant on the payload size but only dependant on a clock frequency.

Table 2. Performances for 1024 bytes payload at 50 MHz clock frequency

Category	Sequential Structure	Parallel Structure	Proposed Design
Data Throughput (Mbps)	285	562	285
Response Time (㎲)	14.96	0.88	0.88

Table 3 shows the design comparison with other structures. Considering the ratio of throughput over logic usage, our design is nearly the same as others. But in the ratio of logic usage over the response time for 1024 bytes frame size at 50 MHz frequency, our design is nearly 2 times better than the parallel structure and 16 times better than the sequential structure.

Table 3. Comparison with other implementation approaches

Category	Sequential Structure	Parallel Structure	Proposed Design
Logic Used (# : number of logic cells)	5437	9702	5605
Throughput / Logic usage (Kbps / #)	53.68	59.32	52.07
1 / Logic usage / Response Time (1 / # / sec)	12.29	117.13	202.74

Our design can support up to 285 Mbps at 50 MHz clock frequency. This satisfies MAC data processing speed of all standards. The data speed of MAC layer is 11 Mbps in IEEE 802.11b and 54 Mbps in IEEE 802.11a and 802.11g. Our design can support all these standards [6]. The response time in our design is 0.88 ㎲ at 50 MHz clock frequency. It is sufficient to support new MAC mechanisms in other 802.11 standards.

6 Conclusion

To support new MAC mechanisms, the response time in the cipher core should have a short interval. In addition, hardware complexity should be considered for low power consumption and manufacturing cost. In this paper, we propose an efficient design of

CCMP which supports new MAC mechanisms such as Block Ack in 802.11e and frame aggregation in other 802.11 standards by adopting the mode toggling approach.

As a result, our design can support up to 285 Mbps at 50 MHz clock frequency and the response time in our design is 44 clock cycles and it is not dependent on the payload size but only dependent on a clock frequency. We can conclude that our design can support new MAC mechanisms and reduce power consumption because our design has the short response time and the data throughput achievable even at the low clock frequency.

Acknowledgments

We would like to thank the members (Suk-Kyu Lee and Yoon-Joo Kim) on Next Generation WLAN Research Team, Electronics and Telecommunications Research Institute, forgiving us a PCI (Peripheral Component Interconnect) board for FPGA verification and embedded software for controlling the board. This work was supported by IT R&D Project funded by Korean Ministry of Information and Communications.

References

1. IEEE standard 802.11i, July 2004.
2. D.Whiting, Hifn, R.Housley, N.Ferguson, MacFergus, “Counter with CBC-MAC(CCM)” RFC 3610, September, 2003
3. Ho Yung Jang, Joon Hyoung Shim, Jung Hee Suk, In Cheol Hwang and Jun Rim Choi, “COMPATIBLE DESIGN OF CCMP AND OCB AES CIPHER USING SEPERATED ENCRYPTOR AND DECRYPTOR FOR IEEE 802.11I”, ISCAS 2004.
4. IEEE standard 802.11e/D13.0 January 2005
5. Matthew S. Gast, “802.11 Wireless Networks – 2nd Edition”, O’REILLY, 2005.
6. Jon Edney, William A. Arbaugh, “Real 802.11 security: Wi-Fi Protected Access and 802.11i”, Addison Wesley, 2003.
7. IEEE P802.11, The Working group for Wireless LANs, http://www.ieee802.org/11/
8. K.U.Jarvinen, M.T.Tommiska and J.O.Skytta, “A fully pipelined memoryless 17.8 Gbps AES-128 encryptor”, Proc. International Symposium on Field-Programmable Gate Arrays (FPGA 2003), pp, 207-215, Monterey, CA, Feb. 2003.
9. G.P.Saggese, A.Mazzeo, N.Mazocca and A.G.M. Strollo, “An FPGA Based Performance Analysis of the Unrolling, Tiling and Pipelining of the AES algorithm”, Proc. FPL 2003, Portugal, Sep. 2003.
10. Xinmiao Zhang and Keshab K. Parhi, “AN EFFICIENT 21.56GBPS AES IMPLEMENTATION ON FPGA”, Conference Record of the Thirty-Eighth Asilomar Conference on volume 1, Nov. 2004.
11. D.Whiting, Hifn, R.Housley, N.Ferguson, MacFergus, "Counter with CBC-MAC(CCM)" RFC 3610, September, 2003

Software-Based Copy Protection for Temporal Media During Dissemination and Playback

Gisle Grimen, Christian Mönch, and Roger Midtstraum

Department of Computer and Information Science,
Norwegian University of Science and Technology
{grimen, moench, roger}@idi.ntnu.no

Abstract. We present a software-based protection mechanism to prevent unauthorized copying of media documents during their presentation on a clients host. Our solution enforces the execution and continuous replacement of security mechanisms on the clients host. Each protection mechanism is only used for a short time interval before it is replaced. The short duration of the time interval prevents a successful analysis and attack of the mechanism. In this way we solve a shortcoming of current solutions. As those employ fixed protection mechanisms they will eventually be circumvented because attackers have virtually unlimited time to analyze them.

1 Introduction

Digital distribution of media documents over networks like the Internet provides both users and content owners with a number of benefits. From the users perspective the new distribution channels provide the opportunity to access the content the user wants from almost any place at any time. The content providers on the other hand benefit from cheap distribution channels that reach the worlds probably largest audience. Despite its obvious advantages, digital distribution of media has presently not been fully embraced. The reason for this is related to the *digital dilemma* [11]. The term digital dilemma refers to the fact, that the same technology that offers new opportunities to content providers and users can be abused to create and illegally distribute unauthorized digital copies of media documents without compensating the content owner.

The problem of unauthorized copies arises from the fact that media documents have to be digitally copied to the users environment in order to be presented. Any mechanism that protects the media document from being copied has to perform its function in the users environment. But this environment is under complete control of the user and a malicious user can attack the protection mechanisms in various ways in order to circumvent them. Many protection mechanisms have been successfully attacked, e. g. Microsoft's digital rights management [12] or Apples FairPlay [18].

Current solutions suffer from two conceptual problems. First, they are usually based on a single, fixed set of mechanisms. Second, the protection mechanisms are integrated in the presentation device and together with the device, they are fully exposed to an attacker. This enables the attacker to analyze and eventually circumvent the mechanisms by modifying the presentation device. Therefore none of the current solutions is capable of solving the problem of unauthorized copying.

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 362–377, 2006.
© Springer-Verlag Berlin Heidelberg 2006

It is a common belief that secure protection mechanisms can only be implemented by hardware-based systems. This is not necessarily true, the deployment of hardware based mechanisms is not guaranteed to increase the security of presentation devices. The effectiveness of hardware-based mechanisms is often not proven. In the past Mod-chips were built to circumvent region code restrictions in DVD-players or copy-protection mechanisms in game consoles. Even advanced protection mechanisms have been successfully attacked, e. g. Microsoft's XBox protection [15]. The biggest drawback of hardware-based solutions is that it is almost impossible to recover from a successful attack. The best known breach is probably the circumvention of CSS [3], which completely disabled the protection of all DVDs.

In this paper, we present a new purely software-based solution to prevent unauthorized copying of media documents in a hostile environment. Our solution solves two conceptual problems that exist in current solutions. We enforce a continuous exchange of protection mechanisms. Each individual protection mechanism is only employed for a brief time interval that is too short to successfully analyze the mechanism.

Our solution allows to quickly detect modifications of presentation devices and to stop the presentation of media documents before more than a few seconds of a media document are copied. Attacks that are based on execution traces and memory dumps become infeasible because our solution makes such attempts very costly in terms of computational and human resources.

The remainder of the paper is structured as follows. In section 2 we identify the threats to media distribution in an untrusted environment. Section 3 describes the basic ideas of our solution. Two main concepts of our approach, the mobile guard and the key exchange protocol, are described in section 4 and section 5 respectively. The architecture of our prototype system is presented in section 6. Related work is discussed in section 7. Section 8 concludes the paper.

2 The Problem with Dissemination and Playback

A media document is owned by a *content owner*. The media document is encoded, for example using an MPEG4 compliant codec, in order to facilitate handling and distribution. The *encoded media document* is of high value to the content owner. It is made available to customers by a *content provider*. The content provider hosts the encoded media document in his environment, the *provider environment*. The provider environment is secure, i. e. the content provider has full control over the data that leaves the provider environment.

The content provider distributes the media document through a distribution channel so that the media document becomes accessible to a presentation program, that is run on the *user's host*. We refer to this presentation program as a *viewer*. The distribution channel might be a network connection or a tangible media that is shipped to the user.

Our main goal is to protect the encoded media document against unauthorized access from the user or from third parties. This means it should not be possible to create a copy of the encoded media document outside of the provider environment.

In order to prevent third parties from gaining unauthorized access to the encoded media document during the distribution process, the encoded media document is usu-

ally encrypted before it is distributed. The encryption is performed in the provider environment, yielding an *encrypted, encoded media document (EEMD)*. The encryption effectively protects the encoded media document against unauthorized access by third parties, allowing it to be safely transported through public distribution channels, e. g. the Internet.

2.1 Presentation Is a Vulnerable Process

While encryption protects the encoded media document from being accessed by third parties, it alone does not guarantee that the encoded media document is protected from an malicious user. The reason for this is, that the EEMD has to be decrypted in the user environment in order to decode the encoded media document and present it to the user. In current systems, the viewer usually receives a decryption key in form of a license. Alternatively, the decryption key might be embedded in the viewer code, or partly in the viewer code and partly in the transport media, e. g. DVD.

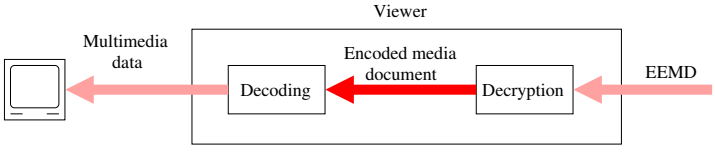


Fig. 1. Decoding of the EEMD

The presentation process is illustrated in figure 1. The viewer first decrypts the EEMD yielding an encoded media document. The encoded media document is then processed by a decoder yielding multimedia data, e. g. video- and audio-frames. Consequently, the following sensitive data exists in the context of the viewer, i. e. in the memory image of the viewer, during the presentation process:

The encoded media document. During the presentation the media document has to be accessed by the viewer. Consequently the unencrypted encoded media document will be present in the viewer's memory, usually distributed in time, because the viewer will only decrypt and decode a part of the EEMD at once.

The decryption key. In order to perform the decryption, the memory image of the viewer has to contain the decryption key for the EEMD.

None of this information is to be disclosed to the user. The encoded media document itself is the data we intend to protect. The possession of the decryption key would allow the user to decrypt the EEMD and so gain access to the encoded media document.

2.2 Threats to the Presentation Process

During the presentation the encoded media document and the decryption keys are accessible to the viewer. If the viewer is executed in the user environment, this data might

be spied out by the user. This problem is analog to the malicious host problem for mobile agents [19]. The problem arises from the fact that the user has full control over the environment in which the viewer is executed. This gives the user the opportunity to change one or more of the following aspects of the environment:

Hardware. The user could change the semantics of opcodes, for example, by running specialized hardware or by executing the viewer on a virtual machine.

Host residual software. The user can change software that is residing on his host and that the viewer depends on. This software includes for example the operating system and dynamically linked libraries. Host residual software includes also the services and processes that are necessary for the execution of programs, for example an X-Windows server.

The viewer code. The user might analyze and change the viewer code before the viewer is executed or during its execution.

The possibility to change these aspects of the environment together with the opportunity for the user to analyze the viewer executable enables different attacks on the viewer. The following attacks can be performed individually or in combination, in order to retrieve the encoded media document or the decryption keys:

Analyzation attack. Since the viewer code is not protected against inspection, a user might analyze the code. Any sensible analysis of code will eventually uncover all secrets embedded in the code.

Modification attack. Since the viewer code is not protected against modifications, a malicious user might change the data- and/or control flow of the viewer in order to retrieve confidential information. This change might be performed statically before the viewer is executed or dynamically during execution of the viewer. Modification attacks can be implemented by changing the viewer code itself or by changing code that is executed in the viewer's context, for example, dynamically linked libraries or operating system code.

Spying attack. A spying attack aims at extracting confidential data from the viewer. Spying attacks might for example be performed by changes in the operating system, e. g. running the viewer process with hardware-trace enabled or by changes in the environment of the viewer, e. g. instantiating programs that monitor changes in the viewers memory, or by building specialized hardware that records every state change in the viewer during the presentation process.

In order to protect the encoded media document against unauthorized access, we have to prevent the user from successfully carrying out any of these attacks, in isolation or in any combination.

3 Copy Protection for Temporal Media

Our copy protection solution is based on continuously downloading code into the viewer. This code is itself protected against attacks and implements security measures against modification and spying attacks on the viewer. The implemented protection mechanisms have the following properties:

Difficult to break. It should be very resource intensive to break the protection mechanism. The cost of breaking should be in the same magnitude as the value of the protected media document. In other words, it has to be so high that only professional distribution of unauthorized copies would earn the costs. In this case non-technical means can be used to detect and stop the distribution.

Break once, break everywhere resistance. An attack to the protection mechanism should not be generalizable. Even if a part of an encoded media document was copied, the same attack should not be applicable to other parts of the same document, to other documents or to other viewer instances.

Facilitate detection. The protection method should support the detection of attacks. This allows the content provider to react on attacks, for example by exchanging compromised encryption keys and protection mechanisms, or by taking legal steps against an attacker.

In the remainder of this section, we will describe specific properties of temporal media and how these can be leveraged to enforce a cooperation between the user and the content provider in order to implement a protection mechanism with the desired properties.

3.1 Temporal Media

With temporal media we refer to media that has a temporal dimension, e. g. audio and video documents. Temporal media exhibits the following important properties:

- Temporal media contains a number of elementary presentation steps that all have to be performed in a given sequence in order to present the whole document.
- The elementary presentation steps are computationally independent from each other and can be processed independently.
- The presentation takes a considerable amount of time, e. g. minutes to hours and elementary presentation steps are performed at a typical rate, e. g. 24 frames per seconds for video documents.

We strive to protect video and audio documents created by the entertainment industry, e. g. movies and TV-series. The value of these media documents arises from performing all, or almost all, elementary presentation steps. A ten minute fraction of a movie document is usually of no considerable value to a costumer. Even getting nearly all presentation steps can be insufficient, as an incomplete movie is usually not of much value.

3.2 Enforcing Continuous Cooperation Through Media Keys

The properties of temporal media allow us to enforce a continuous cooperation between the user and the content provider. This is possible because:

- The content provider controls the data that the user wants to access, and
- The properties of temporal media allow the content provider to give this information in small parts to the user, requiring him to cooperate in order to receive the following part.

In order to control the access on the level of presentation steps, we partition the document into a large number of sets, each containing a few presentation steps. Every set is encrypted with an individual key. We refer to these keys as *media keys*. A typical media document, e. g. a movie, will be encrypted with thousands of different media keys, as illustrated in figure 2. In this model, the user is forced to cooperate in order to receive the media key for the next presentation steps.

The use of media keys allows for fine grained control over the access to media documents. This is in contrast to current protection systems, e. g. DRM systems, where access to a media document is controlled by a single piece of information, e. g. a license [12]. The use of a single license to unlock the complete media document effectively neutralizes the temporal property of the media and leaves no possibility for enforced cooperation.

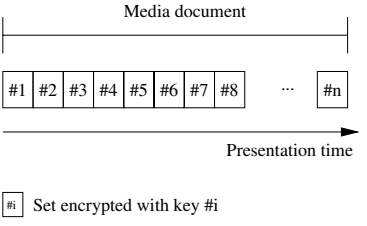


Fig. 2. Encrypting a Temporal Media Document with a Large Number of Media Keys

3.3 Establishing and Maintaining Trust Through Mobile Guards

We use the enforced cooperation to establish protection mechanisms in the viewer and to continously replace them with updated protection mechanisms. To present a document, the viewer has to request media keys from the content provider. In order for these requests to be fulfilled, the content provider requests the execution of protection mechanisms in the viewer. These protection mechanisms are implemented through code that is downloaded from the content provider into the viewer. We refer to this code as a *mobile guard*. The concept of the mobile guard allows a continuous update of the mechanisms that protect the viewer. The task of the mobile guard is twofold:

- It reconfigures the viewer and verifies its new configuration in order to prevent and detect modification attacks.
- It shields the viewer image against the environment in order to make spying attacks difficult.

Each mobile guard reconfigures the viewer and performs a checksum calculation on the new viewer configuration in order to verify that the viewer is unmodified (see section 4.1). Every request for a media key contains this verification information. The content provider will only deliver a new media key to the viewer, if the viewer's integrity is verified. If the checksum indicates that the viewer has been compromised, no further media keys will be sent to the viewer. In addition, the content provider is informed about a possible attack.

The mobile guard itself is obfuscated to protect it against tampering (see section 4.2). Obfuscation protects a mobile guard for a certain time interval. After this time interval has expired, we can not be sure that the mobile guard is performing its task correctly. Therefore every mobile guard protects the viewer only for the duration of this time interval. We refer to this interval as *trust interval*. If the trust interval has expired, we

download the next mobile guard to the viewer and extend the trust into the viewer for another trust interval. Figure 3 illustrates the concept of trust intervals. By downloading new mobile guards before the end of the current trust interval, we ensure that an uncompromised mobile guard is executed on the user's host at all times during a presentation.

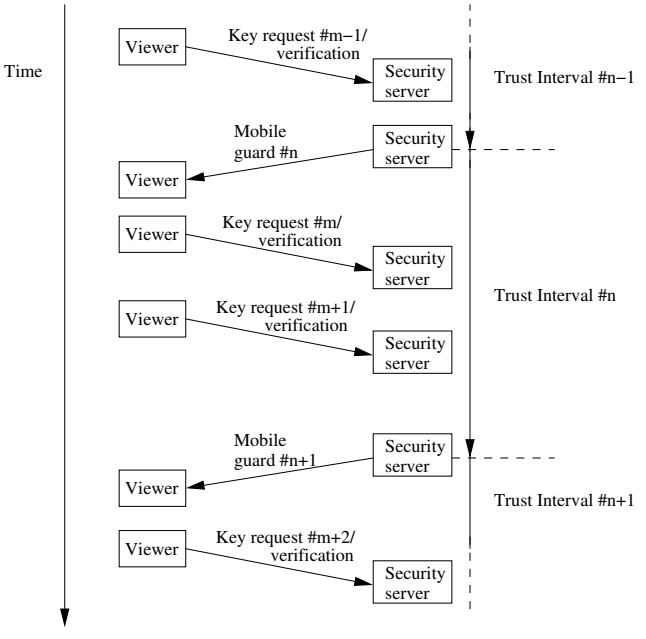


Fig. 3. Mobile guards and trust intervals

4 Mobile Guards

A mobile guard consists of machine code that is downloaded to the viewer during the presentation process. This code does not rely on external code, e. g. dynamically linked libraries, to implement its algorithms. The mobile guard implements the verification of the viewer code, the reconfiguration of the viewer and the shielding algorithms.

4.1 Protecting the Mobile Guard

As the mobile guard executes in the same environment as the viewer, the mobile guard itself is vulnerable to attacks. A malicious user might try to modify the mobile guard in order to circumvent the protection methods implemented by it. It is a fair assumption that a malicious user will apply all automated tools available to him in order to perform an attack efficiently. If we make sure, that a completely automated attack is virtually impossible, we force an attacker to perform intellectual work in order to complete the attack.

Preventing Automated Attacks on the Mobile Guard. To prevent automated attacks from succeeding, we make sure that the mobile guards differ from one another. That is, no two mobile guards have the same code structure. We achieve this by two means. First the algorithms in the mobile guard are partly randomly created, which yields a different control and data flow for each mobile guard. Secondly, we apply obfuscation transformations to the mobile guards.

Obfuscation and randomization prevent a fully automated attack because of the semantic gap between program code and the mental model of the program, as Hohl described in [13]. While the program code specifies how operations on data are performed, it alone does not reveal the purpose of the set of operations that are executed in the mobile guard. The purpose of the set of operations is determined by the mental model that describes the function of the mobile guard. The mental model can only by reconstructed from the code by applying intellectual reasoning. In order to make sensible changes to a mobile guard, an attacker has to reconstruct the model and map it to the machine code. Therefore he has to perform an intellectual analysis of the mobile guard.

Preventing Intellectual Attacks on the Mobile Guard. The only possible attack to the obfuscated randomized mobile guard is an attack that requires intellectual work. While it is possible for an attacker to eventually reverse engineer the software, it is a complex and time consuming process. This allows us to effectively prevent intellectual attacks by selecting an appropriate duration for the trust interval. By restricting the trust interval to a few seconds, we ensure that it is virtually impossible to perform an intellectual attack before the trust interval expires and before the mobile guard is replaced with a new mobile guard.

4.2 Protecting and Verifying the Viewer

Protection against Static Modification Attacks. We use randomly created checksum algorithms to protect the viewer against static modification attacks. A new checksum algorithm is embedded in every mobile guard. Due to the protection of the mobile guard and the random nature of the checksum algorithm, the correct checksum is unpredictable to an attacker. Therefore an attacker is forced to actually execute the mobile guard in order to obtain the correct checksum. As a consequence any static modification of the viewer, i. e. change of the program text before it is loaded and executed, would be detected at the next key request.

Protection against Dynamic Modification Attacks. Due to the obfuscation of the mobile guard, the checksum algorithm itself can not be modified by an attacker. But an attacker might try to deceive the checksum algorithm by modifying its input. An attacker could, for example, exchange modified and unmodified code every time the checksum algorithm might be executed, i. e. every time the control flow enters the mobile guard. The obfuscation of the mobile guard ensures that such an attack has to be performed automatically because the trust interval is much too short to allow for intellectual analysis and sensible modification of the current mobile guard. Therefore the attack has to be based on information like position of control flow, executed instruction, etc.

In order to counter these attacks we randomly reconfigure the viewer. This reconfiguration process consists of rearranging and modifying the viewers instructions. The viewers text segment is partitioned into small parts that are then relocated to randomly selected positions. The code is modified to account for the relocation. We refer to this process as *runtime obfuscation*. It should be noted that runtime obfuscation has to be performed in order to calculate the right checksum.

Runtime obfuscation makes it very difficult for an attacker to provide the checksum algorithm with faked data because it interleaves data reads that are due to the checksum calculation with data reads that are due to runtime obfuscation. A successful attack would require an automated analysis that distinguishes between checksum related and reconfiguration related computations.

Runtime obfuscation makes it also very difficult to automatically replace parts of the viewer, when the control flow leaves the mobile guard. The reason for this is that an attacker would need automated means to identify the corresponding instructions in two different configurations. This can be made even harder through modification of the code [1] and through randomizing the control flow by randomly changing the order in which the mobile guard calls independent functions.

Protection against Spying Attacks. To prevent spying attacks the runtime obfuscation also includes the relocation and disguising of sensitive data areas, e. g. the buffer for decrypted encoded media data and the buffer for transport keys (see section 5). The obfuscation process adjusts instructions that access sensitive data to account for the modifications. This makes an automated extraction of sensitive data very difficult, because it would require to automatically locate the sensitive data, to track it, and to identify the moment in which the data is undisguised. This is very difficult in an automated attack.

An intellectual attack would require an attacker to dump all state changes and intellectually process the dumped memory image in order to identify the undisguised sensitive data.

5 The key Exchange Protocol

The media keys that are needed to decrypt the media document have to be made available to the viewer. In order to retain the temporal property of the media document only one media key should be made available to the viewer at once. The media key will be transported from the security server (see section 6) to the viewer upon request from the viewer. To ensure that the keys are kept confidential during distribution we designed the *key exchange protocol* which enables the viewer to safely obtain the media key of the next segment.

A central element of the key exchange protocol is the *transport key*. Transport keys are generated by the viewer and they are used to protect the media keys while they are sent from the security server to the viewer. The key exchange protocol is illustrated in figure 4. It can be divided into two phases, the request phase, in which the viewer sends a request to the security server and a reply phase, in which the security server sends the requested key to the viewer. The request phase starts with the viewer generating a new transport key (TK_i in figure 4). The transport key is encrypted with the public key of

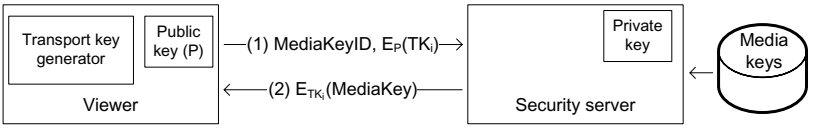


Fig. 4. The Key Exchange

the security server (yielding $E_P(TK_i)$ in figure 4). The encrypted transport key is then transmitted to the security server, together with the identifier of the requested media key (shown as (1) in figure 4).

The reply phase starts when the security server receives the key request message from the viewer. Upon reception of the message, the security server decrypts the transport key and uses it to encrypt the requested media key (yielding $E_{TK_i}(MediaKey)$ in figure 4). The encrypted media key is then transmitted to viewer (shown as (2) in figure 4). By using the previously generated transport key, the viewer is able to decrypt the received data and obtain the requested media key.

5.1 Generation and Protection of Transport Keys

It is crucial to our system, that transport keys are kept hidden from the user because knowledge of the transport key would enable him to retrieve the media key directly from the communication between the viewer and the security server. That means we can not rely on secrets in the viewer or the mobile guard since a malicious user will eventually reveal all secrets in the viewer and the mobile guard, such an approach would make the transport keys available to the user. Consequently the viewer has to generate transport keys on demand. To keep the transport keys secret the following two requirements have to be fulfilled:

1. The generated transport keys have to be unpredictable to a user.
2. The transport keys have to be kept secret from the user.

Generation of Unpredictable Transport Keys. Our solution to this problem is to implement a pseudo-random generator that incorporates information about the system state into the creation of transport keys. The core of this random generator is provided by the mobile guard. Entropy is gathered from the system scheduler, i. e. from the way tasks are scheduled and interrupted. It is very difficult to predict or enforce scheduling orders and times, especially if the mobile guards differ in the numbers and structure of the threads they implement. The use of the scheduler as an entropy source ensures that transport keys are unpredictable. In addition they are virtually unreproducable by rerunning the generation process. This holds because it is nearly impossible to restart the generation process in an identical system state.

The random generation process consists of creating several threads that work on different computational tasks. When a transport key is required, data from the different computational tasks are used as input to a secure hash algorithm along with data from

the current state of the viewer and the executing mobile guard. The result from the secure hash algorithm is then used as transport key.

Hiding of Transport Keys. Since the user can not predict the generated transport keys, his only chance to retrieve a transport key is to dump all state changes in the environment of the viewer in order to retrieve the transport key from this memory image. Due to runtime obfuscation the number of state changes is quite large and due to the random way of generating the transport key it is not possible to determine a priori at which time and/or memory location the transport key is created. The only way to retrieve the transport key would be to analyze the complete memory dump. This is theoretically possible, but requires a large amount of intellectual work. Also if this attack succeeds, it reveals one media key out of thousands. We therefore believe that this attack is unfeasible.

5.2 Reverse Authentication and Man-in-the-Middle Attacks

A potential problem with our solution so far is that the viewer generating the transport key might not be the same as the one checked by our mobile guard. To ensure that the viewer checked by our mobile guard is the same as the one generating the transport key, we extend the input to the checksum calculation to include the transport key that is used in the request for a media key.

The input to the checksum calculation is shown in figure 5. It consists of the instructions of the viewer, the public key of the security server, and the transport key used in the request for a media key. The checksum is included with every key request. This allows the security server to verify that the viewer that created the transport key and sent the key request is actually the viewer which integrity was checked by the mobile guard. As the security server knows the memory layout of the viewer, the transport key and the random algorithm used to compute the checksum, the server can compute the expected checksum and compare it with the checksum it received in the key request. If the checksums match, the security server sends the requested media key, that is encrypted with the transport key, to the viewer. Since the transport key is randomly created, only the viewer that sent the original key request will be able to decrypt the reply and obtain the media key.

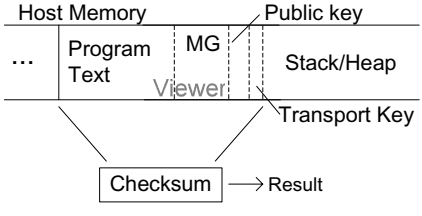


Fig. 5. Input to the Checksum Calculation

6 System Architecture

We have designed a system that implements our protection solution. The main components of our design are shown in figure 6. These are the viewer on the client side and the *security server*, the *streaming server*, and the *protection tool* on the side of the content provider, respectively of the content owner.

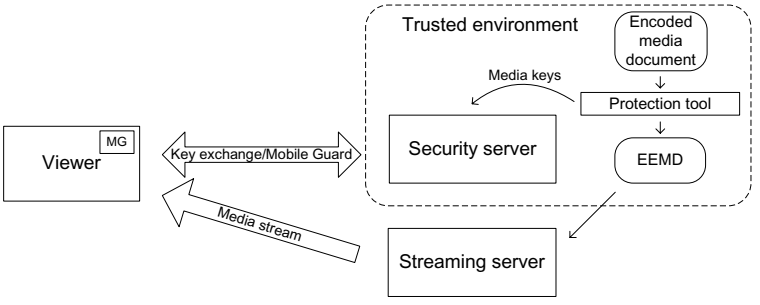


Fig. 6. System Architecture

The *trusted environment* is the only domain in which the encoded media document is available in plaintext as a single access unit. This environment is controlled by the instance that controls the access to the media document, i. e. by the content owner. Before a media document leaves the trusted environment, it is processed by the protection tool. The protection tool creates a large number of media keys and encrypts the encoded media document with the media keys, as described in the previous section, resulting in an EEMD.

Since the EEMD is protected by the media keys, it can be safely made available to third parties. In our architecture it is transfered to a streaming server, that is outside the trusted environment. The task of the streaming server is to distribute EEMDs to users, i. e. to viewers that are executed by users.

The security server has two main tasks. First, it creates mobile guards and delivers them to viewers that are presenting media documents. Second it fulfills requests for media keys from viewers. A requested media key is only sent from the security server to a viewer, if the mobile guard has successfully verified this viewer.

It should be noted, that our architecture allows to separate the security related aspects, i. e. handling of media keys and mobile guards, from the delivery related aspect, i. e. the transport of the EEMD to viewers. Since the encryption effectively protects the EEMD against unauthorized access, the employed distribution channel does not need to be trusted. In our current design, the EEMD is delivered by a streaming server, but it could as well be delivered through, for example, a peer-to-peer network or by the means of a tangible medium.

6.1 Scalability of the Security Server

The security server has three tasks that each requires considerable amount of computational resources. These tasks are the creation of mobile guards, verification of the checksum and decryption of the transport key. In order for our solution to be feasible these tasks need to be computed efficiently.

To minimize the computational load of creating mobile guards, mobile guards are not created for every individual viewer. Instead the security server creates mobile guards for individual trust-intervals. A mobile guard for the current trust interval is then distributed to all viewers that are connected to the security server. In this way the security server

only needs to create a few mobile guards per minute. Only minor changes have to be made to a mobile guard when it is sent to an individual viewer.

Although a common mobile guard is created for each trust-interval, each viewer will return a unique checksum as a result of the reverse authentication. By including the transport key in the last part of the checksum, most of the checksum can be pre-computed. Then only the received transport key needs to be added to the pre-computed checksum before the security server can compare the two checksums.

With these modifications of the creation and handling of the mobile guard, it is clear that the decryption of the transport key is by far the most resource intensive task performed by the security server. To estimate the required resources, we ran some performance measures on our prototype running on a standard 2.5 GHz Pentium4 system using the RSA algorithm with different key lengths. During this test the system performed 1250 decryption operation with a 512 bit keys, 500 decryption operations with a 768 bit keys, and 200 decryption operations with a 1024 bit key each second. From these results it is clear that, although the key exchange is resources intensive, it is in fact possible to serve a significant amount of users on standard hardware.

7 Related Work

Preventing unauthorized access to sensitive data is at the core of any DRM-system. Current systems employ encryption together with software-based protection mechanisms, to protect sensitive data. While little is known about the inner workings of these systems, the protection mechanisms are mainly based on two following techniques, tamper-proofing —sometimes also referred to as self-checking— and obfuscation.

Tamper-proofing aims at preventing modification attacks. It consists of a set of algorithms which verify that the program code in question has not been modified. Aucsmith [2] presents a self-checking method based on encryption and self-modification. In his approach, the program is divided into encrypted segments. During execution a segment is decrypted and later encrypted as the execution jumps to a new segment along with the execution of an accumulator function verifying the correct execution of the program. Other tamper-resistant solutions are described in [4, 14, 5]. All of them use various forms of checksum calculation to verify the integrity of the executing program.

Obfuscation attempts to make analyzation attacks difficult. Obfuscation is based on software transformations, aimed at increasing the time, skill and effort needed to successfully reverse engineer a program. The transformations are designed in such a way that the functionality of the original program is preserved, but the complexity of the program provided to the users is increased. For more details see some of the following articles [10, 9, 8, 21, 20, 6, 17, 7].

The basic techniques tamper-proofing and obfuscation are used in our approach as well as in currently existing protection systems. But currently existing solutions differ from our approach in one crucial point, they can not be dynamically updated. After a program is installed at a users computer the protection mechanisms are fixed and does not change. Even though some of the solutions are self-modifying during execution, the start-state is given. This enables an attacker to replay the execution arbitrarily often in

order to examine the mechanism. Consequently currently existing protection systems will eventually fall to a determined attacker given enough time and effort.

The use of dynamic code to protect media documents is discussed in [16]. Their idea behind using dynamic code is twofold. First, it enables the content owner to take responsibility for protecting their own documents. Secondly, it adds a dynamic dimension to protection mechanism as a whole, making it possible to update the protection mechanism if attacks are discovered. Although they use dynamic code to improve the protection mechanism they still require tamper-proof hardware to store keys and perform decryption. The need for special tamper-proof hardware to store keys and decrypt media documents is exactly what we avoid by combining the use of dynamic code containing randomly generated checksum algorithms and runtime obfuscation with our key exchange protocol.

8 Conclusion and Further Work

We presented a solution to protect encoded media documents during dissemination and playback. A cornerstone of our protection mechanism is to take advantage of the temporal property of audio and video documents. We introduced the concept of media keys, which give us fine grained control over the media document. This control is used to force the user to continuously update the protection mechanisms in the viewer during the presentation of a media document. The lifetime of the protection mechanisms is so short, e. g. 60 seconds, that an intellectual attack to circumvent the mechanisms is not feasible. In this way we solved a major problem that current software protection mechanisms suffer from.

The protection mechanisms enable the detection of modification attacks. Although they can not guarantee that no loss occurs, we achieve that only a small fraction of data is lost before counter measures can be initiated. The possibility of detection is a clear advantage of our system because it discourages modifications attacks, as it leaves traces to the identity of the attacker. The risk of being caught is a powerful deterrence.

Like all pure software solutions, our protection mechanism can not prevent spying attack. But we designed the key exchange protocol and the shielding mechanisms in the mobile guard in such a way, that this attack would require a huge amount of computational and intellectual resources. The media keys together with the key exchange protocol ensure that at no time the viewer's memory contains enough information to obtain the complete encoded media document. In addition the runtime obfuscation makes static spying attacks impossible and dynamic spying attacks difficult. The major reconfigurations force an attacker to continuously dump large parts of the viewers memory. This yields a huge amount of data that has to be intellectually processed to identify sensitive data.

Our approach requires an online communication channel between the viewer and the security server during presentation. This can be a drawback, for example, when considering music services because the user expects to be able to playback music in a mobile environment. For media documents like movies or TV-shows, which are usually consumed at home, the missing mobility is of no concern.

We do not strive to compete with hardware-based solution. In the contrary, our system can be used to protect hardware-based solutions, e. g. Set-top-boxes, against

hardware and software errors and catastrophic breaches by adding a dynamic dimension to the hardware-based mechanisms.

Further work includes the completion of our prototype system and the further investigation of additional runtime obfuscation methods. In addition we would like to optimize the mobile guard obfuscation by leveraging the fact that the randomized mobile guard creation adds another degree of freedom to the structure of the mobile guard.

References

1. Bertrand Anckaert, Bjorn De Sutter, Dominique Chanet, and Koen De Bosschere. Steganography for executables and code transformation signatures. In Choonsik Park and Seongtaek Chee, editors, *Information Security and Cryptology – ICISC 2004: 7th International Conference, Seoul, Korea, December 2-3, 2004, Revised Selected Papers*, volume 3506 of *LNCS*, pages 425–439. Springer, 2005.
2. David Aucsmith. Tamper Resistant Software: An Implementation. In *Proceedings of the First International Workshop on Information Hiding*, pages 317–333. Springer-Verlag, 1996.
3. L. Jean Camp. Drm: doesn't really mean digital copyright management. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 78–87, New York, NY, USA, 2002. ACM Press.
4. Hoi Chang and Mikhail J. Atallah. Protecting Software Code by Guards. In *DRM '01: Revised Papers from the ACM CCS-8 Workshop on Security and Privacy in Digital Rights Management*, pages 160–175. Springer-Verlag, 2002.
5. Yuqun Chen, Ramarathnam Venkatesan, Matthew Cary, Ruoming Pang, Saurabh Sinha, and Mariusz H. Jakubowski. Oblivious Hashing: A Stealthy Software Integrity Verification Primitive. In *IH '02: Revised Papers from the 5th International Workshop on Information Hiding*, pages 400–414. Springer-Verlag, 2003.
6. Stanley Chow, Yuan Gu, Harold Johnson, and Vladimir A. Zakharov. An Approach to the Obfuscation of Control-Flow of Sequential Computer Programs. In *ISC '01: Proceedings of the 4th International Conference on Information Security*, pages 144–155. Springer-Verlag, 2001.
7. Frederick B. Cohen. Operating system protection through program evolution. *Computers and Security*, 12(6):565–584, October 1993.
8. Christian Collberg and Clark Thomborson. Watermarking, tamper-proofing, and obfuscation: tools for software protection. *IEEE Trans. Softw. Eng.*, 28(8):735–746, 2002.
9. Christian Collberg, Clark Thomborson, and Douglas Low. Breaking Abstractions and Unstructuring Data Structures. In *ICCL '98: Proceedings of the 1998 International Conference on Computer Languages*, page 28, Washington, DC, USA, 1998. IEEE Computer Society.
10. Christian Collberg, Clark Thomborson, and Douglas Low. Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs. In *Principles of Programming Languages 1998, POPL'98*, San Diego, CA, January 1998.
11. National Research Council. The Digital Dilemma - Intellectual Property IN THE INFORMATION AGE. Technical report, National Reasearch Council, USA, January 2000.
12. Tobias Hauser and Christian Wenz. DRM Under Attack: Weaknesses in Existing Systems. In *Digital Rights Management - Technological, Economic, Legal and Political Aspects*, pages 206–223. Springer-Verlag, 2003.
13. Fritz Hohl. Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts. In *Mobile Agents and Security*, pages 92–113. Springer-Verlag, 1998.
14. Bill Horne, Lesley R. Matheson, Casey Sheehan, and Robert Endre Tarjan. Dynamic Self-Checking Techniques for Improved Tamper Resistance. In *DRM '01: Revised Papers from*

the ACM CCS-8 Workshop on Security and Privacy in Digital Rights Management, pages 141–159. Springer-Verlag, 2002.

15. Andrew Huang. Keeping Secrets in Hardware: The Microsoft XBox Case Study. In *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 213–227. Springer-Verlag, 2003.
16. Paul Kocher, Joshua Jaffe, Benjamin Jun, Carter Laren, and Nate Lawson. Self-Protecting Digital Content. Technical report, Cryptography Research Inc, April 2003.
17. Toshio Ogiso, Yusuke Sakabe, Masakazu Soshi, and Atsuko Miyaji. Software tamper resistance based on the difficulty of interprocedural analysis. In *The Third International Workshop on Information Security Applications (WISA 2002)*, pages 437–452, August 2002.
18. Andrew Orlowski. itunes drm cracked wide open for gnu/linux. seriously. *The Register*, Jan 2004.
19. Tomas Sander and Christian F. Tschudin. Protecting Mobile Agents Against Malicious Hosts. In *Mobile Agents and Security*, pages 44–60, London, UK, 1998. Springer-Verlag.
20. Chenxi Wang, Jonathan Hill, John Knight, and Jack Davidson. Software Tamper Resistance: Obstructing Static Analysis of Programs. Technical Report CS-2000-12, Dept. Computer science, University of Virginia, Charlottesville, VA, USA, 2000.
21. Chenxi Wang, Jonathan Hill, John C. Knight, and Jack W. Davidson. Protection of Software-Based Survivability Mechanisms. In *DSN '01: Proceedings of the 2001 International Conference on Dependable Systems and Networks (formerly: FTCS)*, pages 193–202. IEEE Computer Society, 2001.

Ambiguity Attacks on the Ganic-Eskicioglu Robust DWT-SVD Image Watermarking Scheme

Grace C.-W. Ting*

Information Security Research (iSECURES) Lab,
Swinburne University of Technology (Sarawak Campus) – SUTS,
93576 Kuching, Malaysia
gting@swinburne.edu.my

Abstract. We consider the security of a DWT-SVD robust watermarking scheme for images due to Ganic and Eskicioglu. We show that although the scheme has been proven to be robust, it falls to ambiguity attacks, namely that both the image owner and an attacker can extract their watermarks from what appear to be originally watermarked images, causing an unresolveable dispute on the ownership claims of the image. The underlying problem is that the designers concentrated solely on robustness but overlooked the equally important issue of rightful ownership first defined by Craver et al. Without providing rightful ownership, a watermarking scheme cannot be used for proofs of ownership applications. To the best of our knowledge, these are first known attacks on this scheme.

Keywords: Information hiding, watermarking, SVD, DWT, rightful ownership, ambiguity.

1 Introduction

As we move from paper to digital media, the ease with which people can generate identical duplicate copies of originals in digital form is partly contributing to the alarming rate of software piracy these days. Add to that the fact that these digital duplicates are exactly identical to their originals, thus once duplicated there is no way to differentiate a copy from its original.

One way to combat against this and so to protect the right of content owners is to use digital watermarking. A *digital watermark* [4] is commonly a message embedded into the digital (cover) content to prove who owns it. Another application of digital watermarks is in *digital fingerprinting* [4] where each copy of digital content sold to buyers are embedded with a unique digital fingerprint to enable enforcement authorities to trace buyers who distribute the contents illegally.

* Currently registered as a postgraduate student with the Centre for Cryptography and Information Security (CCIS), Faculty of Engineering, Multimedia University, Malaysia.

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 378–388, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Among the most popular contents where digital watermarking is applied on are images. An example scenario is where a reknowned photographer puts his photo collection online, i.e. creates an art gallery on his website to share with the rest of the world, but at the same time he would want to lay claim to those photos and not want anyone who downloads his photos to claim them for his own. Digital watermarking schemes typically require two phases: watermark embedding and watermark detection, and they are often designed to ensure that the scheme is robust against various types of attacks. The *robustness* [4] of a watermarking scheme is the ability of the watermarked content to survive signal processing operations and also intentional tampering. On the other hand, some watermarking schemes are *fragile* [4], in that the watermark easily becomes undetectable after even minor modifications of the watermarked content in which it is embedded. Though this is normally undesirable for most applications, this fragility property can be useful for content authentication, i.e. checking the content's integrity against unauthorized tampering.

Nowadays, the concept of wavelets [9] has become popular for image compression mainly because of its use in the JPEG2000 compression standard for images. Due to this fact, Discrete Wavelet Transform (DWT) is increasingly being commonly used for image watermarking. Meanwhile, the Singular Value Decomposition (SVD) [3] is a numerical tool applied in the field of digital signal and image processing, for example in image compression. In recent years [11, 8], it has also been applied to image watermarking. Quite recently, Ganic and Eskicioglu have also proposed a combination of DWT and SVD techniques for image watermarking [6, 7].

Our paper studies the Ganic-Eskicioglu DWT-SVD scheme, first proposed and later revised in [6, 7], respectively. We show that the scheme falls to two ambiguity attacks, and hence do not provide rightful ownership. Rightful ownership was first highlighted by Craver et al. [5] and deals with the problem of whether an embedded watermark in a content can unambiguously prove that the watermark owner is the only person laying rightful claims to the content, or are there other watermarks that can be extracted from the content leading to others laying equally rightful claims to the content. Clearly, this is of major concern in the case of using watermarking to provide proof of ownership.

The outline of this paper is as follows: In Section 2, we briefly review concepts of DWT and SVD for a better understanding of why they are used in watermarking and of how our attacks work. In Section 3, we describe the DWT-SVD watermarking scheme proposed by Ganic-Eskicioglu. In Section 4, we present two ambiguity attacks on the scheme. Section 5 concludes this paper.

2 Preliminaries

In Discrete Wavelet Transforms (DWT) [9], an image is decomposed into a set of basis functions (wavelets) namely low frequency band (LL), high-low frequency band (HL), low-high frequency (LH) and high frequency band (HH); of varying frequency and limitation duration, in contrast to other transforms such as the

Fourier Transform that only provides frequency information. DWT has applications in image processing, where typically the approach is to DWT an image, alter the transform coefficients (by thresholding or zeroing), and inverse DWT to regain an altered image that has denoised, or its edges either sharpened or blurred. DWT is also used for image compression since most DWT coefficients are very small thus can be zeroed nicely for compression. In view that DWT is well suited for image processing operations, it is natural to use DWT-based approaches in image watermarking such that when watermarked images are put through image processing transformations (as is common for a typical image), the embedded watermark remains robust (still detectable).

Meanwhile, every real matrix A (as is an image) can be decomposed into a product of three matrices, $A = U \cdot \Sigma \cdot V^T$ where U and V are orthogonal matrices, $U^T \cdot U = I$, $V^T \cdot V = I$, and $\Sigma = \texttt{diagonal}(\lambda_1, \lambda_2, \ldots)$; where U^T denotes the conjugate tranpose of U. This is known as singular value decomposition (SVD) [3], and the diagonal entries of Σ are called the singular values (SVs) of A, the columns of U (respectively V) are called the left (respectively right) singular vectors of A. Applying SVD in watermarking [11] takes advantage of the fact that large singular values do not vary much after going through common image processing transformations, thus are used to embed watermark information. This complicates the task of an attacker in trying to distort, modify or remove the watermark as it will affect the singular values which would in turn result in a seriously distorted image.

3 Ganic-Eskicioglu DWT-SVD Watermarking Scheme

The DWT-SVD based watermarking scheme by Ganic and Eskicioglu [6, 7] performs *watermark embedding* with following steps:

A1. Decompose the cover image A into four sub-bands (LL, HL, LH, and HH) by using DWT to obtain A^k, where $k = 1, 2, 3, 4$ denotes LL, HL, LH, HH sub-bands.

A2. Do SVD on each sub-band image:

$$A^k = U_A^k \cdot \Sigma_A^k \cdot V_A^{kT}; \quad k = 1, 2, 3, 4, \tag{1}$$

and let λ_i^k, $i = 1, \ldots, n$ denote the singular values of Σ_A^k.

A3. Do SVD on the watermark image:

$$W = U_W \cdot \Sigma_W \cdot V_W^T, \tag{2}$$

and let λ_{Wi}, $i = 1, \ldots, n$ denote the singular values of Σ_W.

A4. Modify the singular values λ_i^k in each sub-band with scaled singular values of the watermark λ_{Wi}:

$$\lambda_i^{*k} = \lambda_i^k + \alpha_k \lambda_{Wi}; \quad i = 1, \ldots, n; \quad k = 1, 2, 3, 4. \tag{3}$$

where α_k are the corresponding scaling factors. Let Σ_A^{*k} denote the diagonal matrix comprising the singular values λ_i^{*k}; $\quad i = 1, \ldots, n; \quad k = 1, 2, 3, 4$.

A5. Obtain the four sets of modified DWT coefficients:

$$A^{*k} = U_A^k \cdot \Sigma_A^{*k} \cdot V_A^{kT}; \;\; k = 1, 2, 3, 4. \tag{4}$$

A6. Obtain the watermarked image A_W^* by performing the inverse DWT using the four sets of modified DWT coefficients.

Meanwhile, the *watermark extraction* comprises the following steps:

B1. Use DWT to decompose the watermarked image A_W^* into four sub-bands: LL, HL, LH, and HH.

B2. Do SVD on each sub-band image:

$$A^{*k} = U_A^k \cdot \Sigma_A^{*k} \cdot V_A^{kT}; \;\; k = 1, 2, 3, 4. \tag{5}$$

B3. Extract the singular values of the watermark from each sub-band:

$$\lambda_{Wi}^k = (\lambda_i^{*k} - \lambda_i^k)/\alpha_k; \;\; i = 1, \ldots, n; \;\; k = 1, 2, 3, 4. \tag{6}$$

B4. Combine the four singular values with corresponding singular vectors U_W, V_W to obtain the embedded watermark:

$$W^k = U_W \cdot \Sigma_W^k \cdot V_W^T; \;\; k = 1, 2, 3, 4. \tag{7}$$

4 Ambiguity Attacks

An *ambiguity attack* [1] is caused by the *rightful ownership problem* [5], where an attacker causes the confusion on who the actual owner of the content (image) is. In particular, this means that besides the original owner being able to extract his watermark from the watermarked content, the attacker similarly is able to extract his own watermark too, and neither can prove who is more right than the other, thus the ambiguity situation and thus none is able to rightfully claim ownership. In this section, we describe two attacks on the Ganic-Eskicioglu scheme that cause ambiguity. Both our attacks have been implemented in Matlab.

4.1 Our First Attack

Consider that the owner has performed the embedding steps A1 to A6 (see Figure 1) to embed his original watermark W_O into his (cover) image A, obtaining the watermarked image A_{WO}. During this embedding process, U_{WO}, V_{WO} were used.

To lay ownership claims to A, the owner performs the extracting steps B1 to B4. If his watermark W_O is successfully extracted, he is deemed the rightful owner.

However, if we suppose that the attacker has access to the singular values λ_i^k in Σ_A^k (which is disclosed during normal watermark extraction - see equation (6)), then he can choose his own fake watermark, W_F instead of W_O and do embedding step A3, i.e. SVD on W_F, to get the singular values λ_{WFi}, $i = 1, \ldots, n$, and corresponding singular vectors U_{WF} and V_{WF}.

To prove ownership of the watermarked image A^*_{WO}, he performs extraction steps B1 to B4 (Figure 2) as follows:

C1. Steps B1 and B2 can be done easily.

C2. Step B3 requires λ_i^k (which is assumed as above to be known by the attacker), and λ_i^{*k} obtained from step [C1] above. This allows to compute:

$$\lambda_{Wi}^k = (\lambda_i^{*k} - \lambda_i^k)/\alpha_k; \;\; i = 1, \ldots, n; \;\; k = 1, 2, 3, 4. \tag{8}$$

C3. By supplying U_{WF} and V_{WF}, step B4 can be done:

$$W_F^k = U_{WF} \cdot \Sigma_W^k \cdot V_{WF}^T; \;\; k = 1, 2, 3, 4; \tag{9}$$

thus the attacker's watermark W_F is successfully extracted from the watermarked image A_{WO}, which is not supposed to contain it.

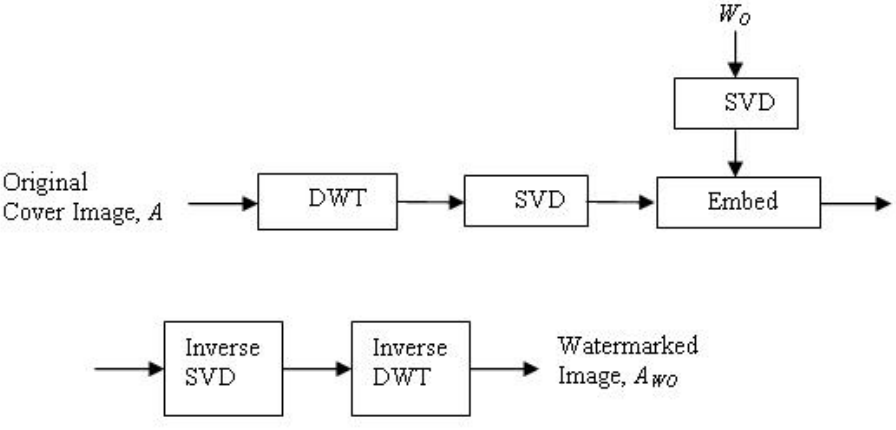


Fig. 1. Steps by the Owner (Watermark Embedding)

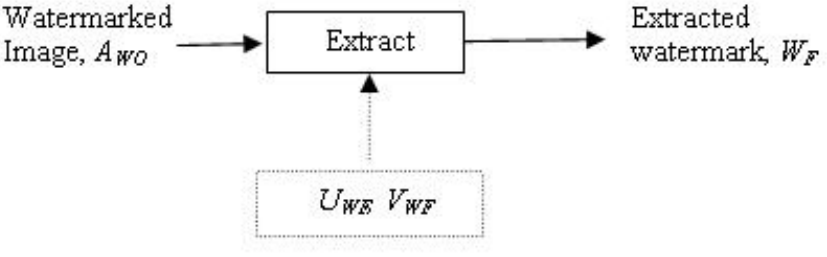


Fig. 2. Steps by the Attacker (Watermark Extraction)

This creates a false positive problem, since only the owner's watermark W_O was embedded but we have just shown that the attacker's watermark, W_F can be

successfully extracted. An ambiguity occurs as both the owner and attacker lay seemingly rightful claims to A_{WO}.

Comparison with Related Work. This attack is similar to [14], but the difference is that the attack in [14] assumed that the attacker is able to embed his watermark W_F directly into the original cover image A. Nevertheless, this assumption is impractical because SVD-based watermarking schemes are semi-blind [11], thus the original cover image is not made public during watermark extraction but only the singular values are.

In [13], an ambiguity attack was considered on the Liu-Tan SVD scheme [11] that requires creating a fake original cover image and a meaningful but very constrained watermark, in the form of an 'X'. The existence of such a small class of weak (constrained) watermarks out of the entire watermark space is akin to the existence of weak keys in ciphers [12] which can be easily avoided by requiring that this small class of watermarks not be used. In contrast, both our attacks do not require any infeasible constraints on the attacker's watermark, nor that watermark be specially computed.

4.2 Our Second Attack

Our first attack supposed that the attacker has access to singular values λ_i^k in Σ_A^k, which means for the attack to work, the owner must have previously made at least one ownership claim such that during watermark extraction Σ_A^k was disclosed.

We can do away with this requirement. Consider that the owner (as per Figure 1) has embedded his watermark W_O (Figure 3) into his cover image A (Figure 4) to obtain A_{WO} (Figure 5).

Fig. 3. Real Watermark W_O

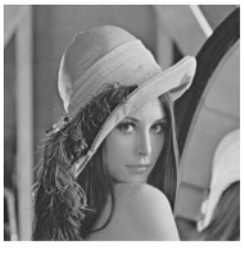


Fig. 4. Original Cover Image A

Fig. 5. Watermarked Image A_{WO}

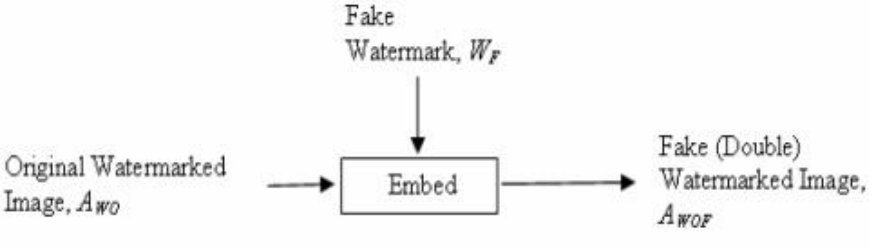


Fig. 6. Steps by the Attacker (Watermark Embedding)

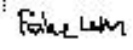


Fig. 7. Fake Watermark A_F

The attacker can directly take A_{WO}, and then performs embedding steps A1 to A6 (see Figure 6) to embed his watermark W_F (Figure 7) onto the original watermarked image A_{WO} to get the fake watermarked image, A_{WOF} (Figure 8):

D1. Decompose the original watermarked image A_{WO} into four sub-bands (LL, HL, LH, and HH) by using DWT to obtain A^k_{WO}, where $k = 1, 2, 3, 4$ denotes LL, HL, LH, HH sub-bands.
D2. Do SVD on each sub-band image:

$$A^k_{WO} = U^k_{A_{WO}} \cdot \Sigma^k_{A_{WO}} \cdot V^{kT}_{A_{WO}}; \;\; k = 1, 2, 3, 4, \tag{10}$$

and let λ^k_{WOi}, $i = 1, \ldots, n$ denote the singular values of $\Sigma^k_{A_{WO}}$.
D3. Do SVD on the fake watermark image:

$$W_F = U_{WF} \cdot \Sigma_{WF} \cdot V^T_{WF}, \tag{11}$$

and let λ_{WFi}, $i = 1, \ldots, n$ denote the singular values of Σ_{WF}.

Fig. 8. Fake (Double) Watermarked Image A_{WOF}

D4. Modify the singular values λ^k_{WOi} in each sub-band with scaled singular values of the watermark λ_{WFi}:

$$\lambda^{*k}_{Fi} = \lambda^k_{WOi} + \alpha_k \lambda_{WFi}; \quad i = 1, \ldots, n; \quad k = 1, 2, 3, 4. \tag{12}$$

where α_k are the corresponding scaling factors. Let $\Sigma^{*k}_{A_{WOF}}$ denote the diagonal matrix comprising the singular values λ^{*k}_{Fi}; $i = 1, \ldots, n$; $k = 1, 2, 3, 4$.

D5. Obtain the four sets of modified DWT coefficients:

$$A^{*k}_{WOF} = U^k_{A_{WO}} \cdot \Sigma^{*k}_{A_{WOF}} \cdot V^{kT}_{A_{WO}}; \quad k = 1, 2, 3, 4. \tag{13}$$

D6. Obtain the watermarked image A^*_{WOF} by performing the inverse DWT using the four sets of modified DWT coefficients.

Note that A^*_{WOF} is now embedded with two watermarks, first with W_O by the owner, and second with W_F by the attacker.

Ambiguity. If the owner wishes to claim ownership of A, he performs extraction steps B1 to B4 to extract the original watermark W_O from the original watermarked image A_{WO} (see Figure 9) by supplying Σ^k_A, U_{WO} and V_{WO}.

An attacker could equally lay claim to A by doing extraction steps B1 to B4 to extract his watermark W_F by claiming A_{WOF} to be the original watermarked image, instead of A_{WO} (see Figure 10). He supplies $\Sigma^k_{A_{WO}}$, U_{WF} and V_{WF}.

This creates ambiguity because both the owner and attacker can equally claim that the original cover image belong to each, respectively since each can extract his own watermark. In this situation, no one would know which party is telling the truth. One limitation of this attack compared to that in Section 4.1 is that it appears to be possible for the owner to recover W_O from A_{WOF} while an attacker does not seem to be able to recover W_F from A_{WO}. The attack in Section 4.1 does not exhibit this problem because there is only one copy of the watermarked image A_{WO}.

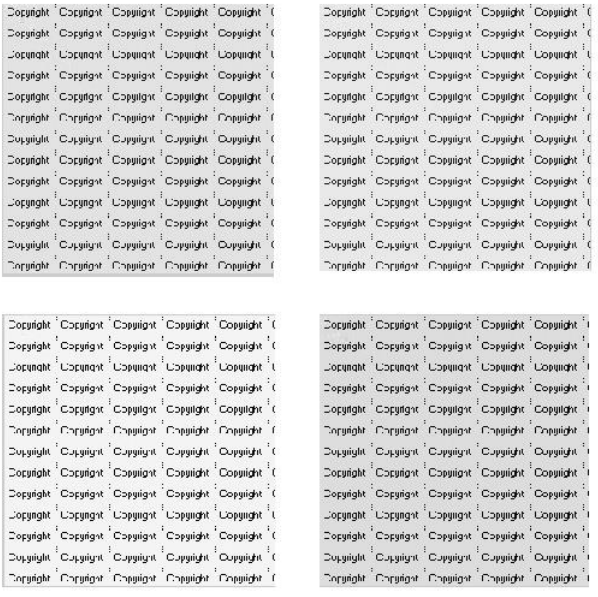


Fig. 9. Recovered Real Watermark W_O

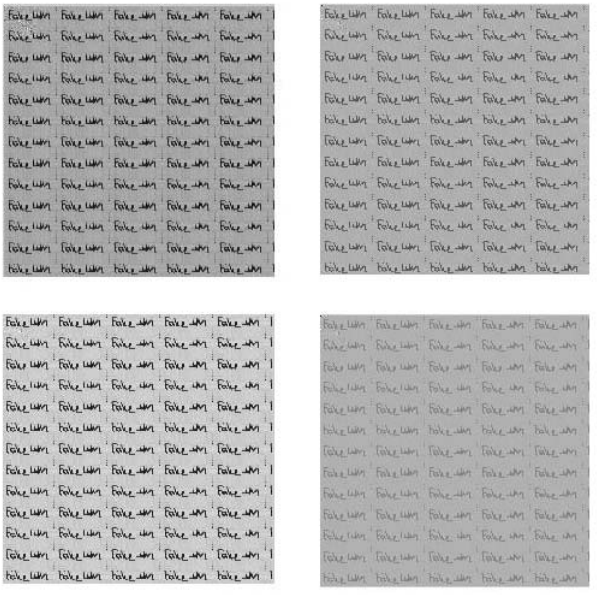


Fig. 10. Recovered Fake Watermark W_F

5 Concluding Remarks

We have shown that the Ganic-Eskicioglu DWT-SVD scheme falls to ambiguity attacks. The authors concentrated only in proving the robustness of their scheme, but neglected to consider the problem of rightful ownership which should be provided by any watermarking scheme since it is commonly used for proofs of ownership. We conclude that the scheme should not be used for proving ownership of content.

At the time of writing, we are unaware of any suitable countermeasure to protect SVD-based watermarking schemes against ambiguity attacks. We leave this as an open problem.

Acknowledgement

We thank Dennis ML Wong for his implementation of basic Ganic-Eskicioglu embedding/extracting upon which our attack codes are based, and for his advice, insight and suggestion to consider SVD-based watermarking schemes. We thank the Digital Watemarking groupmates of iSECURES Lab for inspiring the idea of double watermarking used in our attack in Section 4.2, and Enn Ong for constant encouragement. We are grateful to Mohammad Umar Siddiqi for his confidence and support. We thank Swee-Huay Heng for help in accessing the seminal reference [3]. Finally, we thank God for everything.

References

1. A. Adelsbach, S. Katzenbeisser and A.-R. Sadeghi, "On the Insecurity of Non-invertible Watermarking Schemes for Dispute Resolving", *Proceedings of the International Workshop on Digital Watermarking (IWDW '03)*, Lecture Notes in Computer Science, vol. 2939, Springer-Verlag, pp. 355-369, 2003.
2. A. Adelsbach, S. Katzenbeisser and H. Veith, "Watermarking Schemes Provably Secure Against Copy and Ambiguity Attacks", *Proceedings of the ACM Workshop on Digital Rights Management (DRM '03)*, pp. 111-119, 2003.
3. H.C. Andrews and C.L. Patterson, "Singular Value Decomposition (SVD) Image Coding", *IEEE Transactions on Communications*, pp. 425-432, 1976.
4. I.J. Cox, M.L. Miller and J.A. Bloom, *Digital Watermarking*, Morgan Kaufmann, 2002.
5. S. Craver, N. Memon, B.L. Yeo and M.M. Yeung, "Resolving Rightful Ownerships with Invisible Watermarking Techniques: Limitations, Attacks and Implications", *IEEE Journal of Selected Areas in Communication*, vol. 16, no. 4, pp. 573-586, 1998.
6. E. Ganic and A.M. Eskicioglu, "Robust DWT-SVD Domain Image Watermarking: Embedding Data in All Frequencies", *Proceedings of the ACM Multimedia and Security Workshop*, pp. 166-174, 2004.
7. E. Ganic and A.M. Eskicioglu, "Robust Embedding Of Visual Watermarks using DWT-SVD", *Journal of Electronic Imaging*, 2005.

8. E. Ganic, N. Zubair and A.M. Eskicioglu, "An Optimal Watermarking Scheme Based on Singular Value Decomposition", *Proceedings of the IASTED International Conference on Communication, Network, and Information Security (CNIS '03)*, ACTA Press, pp. 85-90, 2003.
9. R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Prentice Hall, 2002.
10. S. Katzenbeisser and H. Veith, "Securing Symmetric Watermarking Schemes Against Protocol Attacks", *Proceedings of the SPIE vol. 4675, Security and Watermarking of Multimedia Contents IV*, pp. 260-268, 2002.
11. R. Liu and T. Tan, "An SVD-Based Watermarking Scheme for Protecting Rightful Ownership", *IEEE Transactions on Multimedia*, vol. 4, no. 1, pp. 121-128, 2002.
12. A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
13. Y. Wu, "On the Security of an SVD-Based Ownership Watermarking", *IEEE Transactions on Multimedia*, vol. 7, no. 4, pp. 624-627, 2005.
14. X.-P. Zhang and K. Li, "Comments on 'An SVD-Based Watermarking Scheme for Protecting Rightful Ownership' ", *IEEE Transactions on Multimedia*, vol. 7, no. 3, pp. 593-594, 2005.

Universal Custodian-Hiding Verifiable Encryption for Discrete Logarithms

Joseph K. Liu[1], Patrick P. Tsang[2,⋆], Duncan S. Wong[3,⋆⋆], and Robert W. Zhu[3]

[1] Department of Computer Science,
University of Bristol,
Woodland Road, Bristol, BS8 1UB, UK
liu@cs.bris.ac.uk
[2] Department of Computer Science,
Dartmouth College,
Hanover NH 03755, USA
patrick@cs.dartmouth.edu
[3] Department of Computer Science,
City University of Hong Kong,
Hong Kong, China
{duncan, zhuwei}@cs.cityu.edu.hk

Abstract. We introduce the notion of Universal Custodian-Hiding Verifiable Encryption (UCH-VE) and propose a scheme of this type for discrete logarithms. A UCH-VE scheme allows an encryptor to designate t out of a group of n users and prepare a publicly verifiable ciphertext in such a way that any k of these t designated users can recover the message. The values of k and t are set arbitrarily by the encryptor. The anonymity of these t designated users will also be preserved. The UCH-VE scheme captures the notions of various types of verifiable encryption schemes that include conventional one-decryptor type, conventional threshold type, designated-1-out-of-n custodian-hiding type and designated group custodian-hiding type. On efficiency, the new scheme avoids using inefficient cut-and-choose proofs and compares favourably with the state-of-the-art verifiable encryption schemes for discrete logarithms.

1 Introduction

A verifiable encryption (VE) scheme [15, 1, 2, 6, 9] for a relation $\mathcal{R}$ is a protocol that allows a prover to convince a verifier that a ciphertext is an encryption of a value w under a given public key such that $(w, \delta) \in \mathcal{R}$ for a given δ, while no more information about w is leaked. We call this type of VE schemes as *conventional one-decryptor type*. For all the schemes of this type cited above, the verifier knows the identity of the receiver.

⋆ The work of the author was done when he was with the Chinese University of Hong Kong.

⋆⋆ The work of the author was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. 9040904 (RGC Ref. No. CityU 1161/04E)).

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 389–409, 2006.
© Springer-Verlag Berlin Heidelberg 2006

A threshold VE scheme [6] extends the conventional one-decryptor VE in such a way that, instead of having one receiver (or decryptor), it has a group of n receivers (or decryptors) and the ciphertext requires at least k of these n receivers to work together for decrypting it. We call this type of VE schemes as *conventional threshold type*.

A custodian-hiding VE scheme [12] also works on a group of n receivers. But unlike a threshold VE scheme, a custodian-hiding VE scheme only allows one designated receiver out of n receivers to recover the message from the ciphertext. No other receivers in the group, even they collaborate, can decrypt the ciphertext. Also the anonymity of the designated receiver is provided so that the public verifier does not know the identity of the designated receiver. We call this type of VE schemes as *designated-1-out-of-n custodian-hiding type*. Notice that this is the first time that receiver anonymity is considered in VE schemes.

In [12], there are two extensions to the designated-1-out-of-n custodian-hiding type. One extension replaces the designated receiver with a t-element set of designated receivers. It requires the t designated receivers to work jointly for decrypting a ciphertext. Anonymity requirement is also changed to maintain the anonymity of all the t designated receivers. By convention, we call it the designated-t-out-of-n custodian-hiding type. Another extension modifies the requirement of designated-t-out-of-n custodian-hiding type slightly so that anyone of the t designated receivers can decrypt a ciphertext. Anonymity requirement is the same as that of the first extension. These two extensions are closely interrelated. Consider the threshold setting of the t-element subgroup of designated receivers. The first extension is actually the (t, t)-threshold type while the second extension is the $(1, t)$-threshold type. Due to this inter-relationship, we put these two extensions into the same type of VE schemes and call the type as *designated group custodian-hiding type*.

Universal Custodian-Hiding Verifiable Encryption (UCH-VE). In this paper, we extend the notion of custodian-hiding VE and introduce the ***universal*** custodian-hiding VE (UCH-VE). A UCH-VE scheme allows an encryptor to designate t out of a group of n users and prepare a publicly verifiable ciphertext in such a way that any k of these t designated users can recover the message. The values of k and t are set arbitrarily by the encryptor. In addition, the anonymity of these t designated users will be preserved. Obviously, we require that $1 \leq k \leq t \leq n$. Our notion is *universal* as it includes all the current notions of VE reviewed above.

To see this, we have a UCH-VE scheme of conventional one-decryptor type when $n = 1$; a UCH-VE scheme of conventional threshold type when $t = n$; a UCH-VE scheme of designated-1-out-of-n custodian-hiding type when $t = 1$; a UCH-VE scheme of the first extension of designated group custodian-hiding type when $k = t$; and a UCH-VE scheme of the second extension of designated group custodian-hiding type when $k = 1$. Essentially, we would like to propose the idea of having one single scheme support all possible values of (k, t, n) provided that the condition $1 \leq k \leq t \leq n$ holds.

Our Contributions. We introduce the notion of (k, t, n) UCH-VE and propose a scheme of this type for discrete logarithms. The scheme is universal as it supports various types of VE that include conventional one-decryptor type, conventional threshold type, designated-1-out-of-n custodian-hiding type and designated group custodian-hiding type by adjusting the values of k, t and n.

Not only our scheme is the first one of this type, the efficiency of our scheme also compares favourably with the state-of-the-art VE schemes. In particular, it gives the same order of performance when compared with the conventional one-decryptor type VE scheme proposed by Camenisch and Shoup [9]. Our scheme can be considered as an extension of theirs to the UCH-VE setting. Our scheme also avoids using inefficient cut-and-choose proofs and is much more efficient than the designated-1-out-of-n custodian-hiding type scheme and the two extensions for designated group custodian-hiding type proposed by Liu et al. in [12]. Our scheme is also *non-interactive*. That is, the encryptor generates a ciphertext without the participation of the public verifier. In [12], all of their schemes require the verifier to get involved.

Applications. There are numerous applications of verifiable encryption. Examples include publicly verifiable secret sharing [15], optimistic fair exchange [1], revokable anonymous credential [7], and encrypted message gateway [12].

In a key escrow application, a user Alice encrypts her secret key under the public key of a custodian and sends to a verifier (e.g. an organization or a government) together with a proof that the ciphertext is indeed an encryption of her own secret key. In order to increase the level of trust, n custodians may be called in rather than only one single custodian. The key is then split and shared among a designated set of t out of these n custodians. For the interest of Alice, she may not want the verifier to find out which particular t custodians are holding shares of her secret key. To increase reliability of this key escrow system, we may also want Alice's secret key to be able to recover as long as there are at least k of these t designated custodians are in service. In other words, even some (at most $t - k$) of the designated custodians have gone offline, Alice's secret key can still be recovered.

1.1 Scheme Outline

To construct a UCH-VE scheme for a relation $\mathcal{R}$, we divide our work into two phases. In the first phase, we construct an encryption scheme called *Universal Custodian-Hiding Encryption* (UCH-Enc). In the second phase, we add in a proof system which allows the encryptor of the UCH-Enc scheme (now acts as a prover) to convince a verifier that a ciphertext can be decrypted jointly by any k out of t designated receivers of a group of n receivers (or decryptors) to a value w such that $(w, \delta) \in \mathcal{R}$ for a given δ. The (public) verifier is a party who only has the public system information such as the public keys of all the receivers.

For a group of n receivers, a UCH-Enc scheme allows anyone who has the public keys of the n receivers to arbitrarily choose and specify integers k, t and a t-element subset of the receiver group provided that $1 \leq k \leq t \leq n$, then generate

a ciphertext on a message. The t-element subset is called the *designated decryptor group*. To recover the message, at least k members of the designated decryptor group have to work jointly on the ciphertext using their private keys. Any $k-1$ receivers (or fewer) cannot recover the message. We define a secure UCH-Enc scheme to be semantically secure against adaptive chosen ciphertext and public key attacks. The scheme should also provide anonymity to the designated decryptor group. A secure UCH-VE scheme built on a secure UCH-Enc scheme should also be Sound and Special Honest-Verifier Zero Knowledge.

Notations. For positive real numbers a and b, let $\lfloor a \rfloor$ be the largest integer smaller or equal to a, and $\lceil a \rfloor$ be the largest integer smaller or equal to $a + 1/2$. Let $[a] = \{0, \cdots, \lfloor a \rfloor - 1\}$, $[a, b] = \{\lfloor a \rfloor, \cdots, \lfloor b \rfloor\}$, and $[-a, b] = \{-\lfloor a \rfloor, \cdots, \lfloor b \rfloor\}$.

Let $\wp_t(S)$ be the collection of all t-element subsets of the power set of S, i.e. $\wp_t(S) = \{\mathcal{T} \in 2^S \ : \ |\mathcal{T}| = t\}$. By $\wp_t(S)[j]$, we mean the j-th element of $\wp_t(S)$ for some arbitrary order.

By $|X|$, we denote the length of X in binary form; or the cardinality if X is a set. By $\mathrm{neg}(\ell)$, we denote a negligible function where $\mathrm{neg}(\ell) < 1/\mathrm{poly}(\ell)$ holds for all polynomial $\mathrm{poly}(\ell)$ and all sufficiently large ℓ.

Paper Organization. We define a secure UCH-Enc in Sec. 2 and propose a scheme of this type in Sec. 3. We then define a secure UCH-VE in Sec. 4 and describe our UCH-VE scheme in Sec. 5.

2 UCH-Enc: Definition and Security Model

A (k, t, n) UCH-Enc (*Universal Custodian-Hiding Encryption*) scheme, where $1 \leq k \leq t \leq n$, allows an encryptor to arbitrarily set up a group of n receivers (or decryptors), *designate* t (or more) members of the group and encrypt a message m to a ciphertext ψ under the public keys of the n group members. To recover m from ψ, at least k of the t designated group members have to cooperate. For non-designated group members, they cannot recover m even they collaborate with each other. In addition, we require that the public cannot tell who is a designated member or who is not. We call this additional property the *Designated Decryptor Anonymity*.

Note that when $t = n$, the UCH-Enc scheme becomes a conventional threshold encryption scheme [11, 14, 10]. However, when $k = t$, it is not alike a threshold scheme. Instead, in the special case when $k = t = 1$ and consider n to be the total number of public keys in the entire system, a UCH-Enc scheme becomes reminiscent to a public key encryption with key privacy scheme [3]. A public key encryption with key privacy scheme allows an encryptor to generate a ciphertext without leaking any information about the identity of the decryptor.

A UCH-Enc scheme is defined as a tuple of four probabilistic polynomial-time (PPT) algorithms which correspond to System Setup, Key Generation, Encryption and Decryption. For every PPT algorithm discussed in this paper, its running time is assumed to be some polynomial of a system-wide security parameter $\ell_0 \in \mathbb{N}$. For simplicity, we sometimes omit 1^{ℓ_0} from the description of the input specification of an algorithm.

System Setup: On input a security parameter $\ell_0 \in \mathbb{N}$ (in the form of 1^{ℓ_0}), the algorithm outputs the specification of a message space $\mathcal{M}$, a group size n, and some auxiliary security parameters ℓ_i $(1 \leq i \leq n)$ such that each of n and ℓ_i $(1 \leq i \leq n)$ is some polynomial in ℓ_0.

Key Generation: On input a security parameter $\ell_i \in \mathbb{N}$ and the specification of a message space $\mathcal{M}$, a public/private key pair $(\mathsf{PK}_i, \mathsf{SK}_i)$ is generated.

Encryption: On input a set $\{\mathsf{PK}_i\}_{1 \leq i \leq n}$ of n public keys, some integer k, t such that $1 \leq k \leq t \leq n$, some set $\mathcal{T} \in \wp_t([1, n])$ called *designated decryptor group*, and some message $m \in \mathcal{M}$ with label $L \in \{0,1\}^*$, the algorithm outputs a ciphertext ψ.

Decryption: On input a set $\{\mathsf{PK}_i\}_{1 \leq i \leq n}$ of n public keys, a k-element set $\mathcal{K} \in \wp_k([1, n])$, a set $\{\mathsf{SK}_i\}_{i \in \mathcal{K}}$ of private keys, the specification of a message space $\mathcal{M}$, and a ciphertext ψ with label $L \in \{0,1\}^*$, the algorithm outputs either reject for failure of decryption or some message $m' \in \mathcal{M}$.

Remark: The specification of the message space $\mathcal{M}$ is generated in the System Setup phase. An alternative approach of specifying $\mathcal{M}$ is to remove its specification from the System Setup and let it be determined by the set of public keys used in the Encryption algorithm. Each of these two approaches has its advantages and tradeoffs. Although we pick the current one in our paper, one can readily convert ours to the alternative approach if preferred.

2.1 Security Model

We require a secure UCH-Enc scheme to be *correct*, *indistinguishable against adaptive chosen ciphertext and public key attacks* and *designated decryptor anonymous*.

Decryption Correctness. For all n, ℓ_i $(1 \leq i \leq n)$ and the specification of $\mathcal{M}$ generated by System Setup with input ℓ_0; for all key pairs $(\mathsf{PK}_i, \mathsf{SK}_i)$, $1 \leq i \leq n$, generated by Key Generation with corresponding inputs; for any message $m \in \mathcal{M}$ with label $L \in \{0,1\}^*$; and for any designated decryptor group $\mathcal{T} \in \wp_t([1, n])$, we have the following holds with probability at least $1 - \mathrm{neg}(\ell_0)$.

> *If ψ is generated by Encryption with inputs $\{PK_i\}_{1 \leq i \leq n}$, k, t, $\mathcal{T}$, and (m, L), such that $1 \leq k \leq t \leq n$, then Decryption outputs m on input $\{PK_i\}_{1 \leq i \leq n}$, any $\mathcal{K}$ such that $|\mathcal{K} \cap \mathcal{T}| \geq k$, $\{SK_i\}_{i \in \mathcal{K}}$, the specification of $\mathcal{M}$, and (ψ, L).*

Indistinguishability Against Adaptive Chosen Ciphertext and Public Key Attacks. Different from a conventional threshold decryption scheme, a UCH-Enc scheme has the additional component called designated decryptor group. We require that at least k members in this designated decryptor group are working together before being able to recover the message. Also, any other receivers who do not belong to the designated decryptor group cannot obtain the message, even they collaborate. We follow the notion of IND-CCA2 security by Rackoff and Simon [13] to set up a game called Game Confidentiality. The

game captures both the threshold setting and the notion of designated decryptor group. It also captures the idea of adaptive chosen public key attack. Details of the formalization are given in Appendix A.

Designated Decryptor Anonymity. Along with the new notion of designated decryptor group, we also want to provide anonymity to the members of the group. In particular, we require that no one can tell from a ciphertext about who is a designated decryptor or who is not if he does not have corresponding private keys. In Appendix A, we describe an indistinguishability-based game called Game Anonymity for this.

3 A UCH-Enc Scheme

The main technique is based on Camenisch-Shoup encryption scheme [9]. Their scheme can be shown to be IND-CCA2 secure without the random oracle model [5]. The scheme below extends their scheme by adding the ingredients of designated decryptor group as well as threshold decryption.

System Setup: On input a system-wide security parameter, $\ell_0 \in \mathbb{N}$, a group size n, auxiliary security parameters ℓ_i $(1 \leq i \leq n)$ and the specification of a message space $\mathcal{M} = [N_0]$ are generated where N_0 is a prime and each of n, ℓ_i, $1 \leq i \leq n$, and $|N_0|$ is some polynomial in ℓ_0.

Key Generation: For $i = 1, \cdots, n$, a public key pair $(\mathsf{PK}_i, \mathsf{SK}_i)$ is generated as follows:

1. Select two random ℓ_i-bit Sophie Germain primes p'_i and q'_i, with $p'_i \neq q'_i$, and compute $p_i = 2p'_i + 1$, $q_i = 2q'_i + 1$, $N_i = p_i q_i$ and $N'_i = p'_i q'_i$.
2. Randomly choose $x_{i,1}, x_{i,2}, x_{i,3} \in_R [N_i^2/4]$, $g'_i \in_R \mathbb{Z}^*_{N_i^2}$, and compute $g_i = (g'_i)^{2N_i}$, $y_{i,1} = g_i^{x_{i,1}}$, $y_{i,2} = g_i^{x_{i,2}}$ and $y_{i,3} = g_i^{x_{i,3}}$.
3. Select a keyed hash function [4] H_i and randomly choose a key $\mathsf{hk}_i \in_R \{0,1\}^{\ell_i}$. The resulting keyed hash function $H_{i,\mathsf{hk}_i} : \{0,1\}^* \rightarrow [2^{\ell_i}]$ should be collision-resistant.
4. Compute $h_i = (1 + N_i \bmod N_i^2) \in \mathbb{Z}^*_{N_i^2}$. Set function $\mathrm{abs}_i : \mathbb{Z}^*_{N_i^2} \rightarrow \mathbb{Z}^*_{N_i^2}$ to map $(a \bmod N_i^2)$ for $0 < a < N_i^2$, to $(N_i^2 - a \bmod N_i^2)$ if $a > N_i^2/2$, and to $(a \bmod N_i^2)$, otherwise. Notice that the order of h_i is N_i and $v^2 = (\mathrm{abs}_i(v))^2$ for all $v \in \mathbb{Z}^*_{N_i^2}$.

The public key is $\mathsf{PK}_i = (N_i, h_i, g_i, y_{i,1}, y_{i,2}, y_{i,3}, H_i, \mathsf{hk}_i)$ and the private key is $\mathsf{SK}_i = (x_{i,1}, x_{i,2}, x_{i,3})$. We require that $N_0 < N_i$.

Encryption: For a set $\{\mathsf{PK}_i\}_{1 \leq i \leq n}$ of n public keys generated as above, given some integers k, t such that $1 \leq k \leq t \leq n$ and $t < N_0$, and some designated decryptor group $\mathcal{T} \in \wp_t([1,n])$, a message $m \in [N_0]$ with label $L \in \{0,1\}^*$ is encrypted as follows:

1. (*Secret Sharing*) Generate a random polynomial $f(x) = \sum_{j=0}^{k-1} a_j x^j$ of degree $(k-1)$ over $GF(N_0)$ such that $f(0) = m$ (i.e. $a_0 = m$). Compute $m_i = f(i)$ for all $i \in \mathcal{T}$.

2. (*Generating Ciphertext Components*) For each $i \in \mathcal{T}$, choose a random $r_i \in_R [N_i/4]$ and compute $\mathsf{ctxt}_i = (u_i, e_i, v_i)$ as

$$u_i = g_i^{r_i}, \quad e_i = y_{i,1}^{r_i} h_i^{m_i}, \quad \text{and} \quad v_i = \mathrm{abs}_i\left((y_{i,2} y_{i,3}^{H_{i,\mathsf{hk}_i}(u_i,e_i,L,k,t,n)})^{r_i}\right).$$

   Note that u_i, e_i and v_i are in $\mathbb{Z}_{N_i^2}^*$ but $v_i \leq N_i^2/2$. For each $i \in [1,n] \setminus \mathcal{T}$, generate $\mathsf{ctxt}_i = (u_i, e_i, v_i)$ by picking each of its components at random from its corresponding domain.
3. The ciphertext is $\psi = (k, t, \mathsf{ctxt}_1, \cdots, \mathsf{ctxt}_n)$.

Decryption: Let $\mathcal{K}$, with $|\mathcal{K}| = k$, be a subset of the designated decryptor group $\mathcal{T}$. To decrypt a ciphertext $\psi = (k, t, \mathsf{ctxt}_1, \cdots, \mathsf{ctxt}_n)$ with label L under the set $\{\mathsf{SK}_i\}_{i \in \mathcal{K}}$ of k private keys, the following steps are carried out:

1. (*Recovering Shares*) For each $i \in \mathcal{K}$, $\mathsf{ctxt}_i = (u_i, e_i, v_i)$ is decrypted under SK_i as follows:
   (a) Check if $\mathrm{abs}_i(v_i) = v_i$ and $u_i^{2(x_{i,2} + H_{i,\mathsf{hk}_i}(u_i,e_i,L,k,t,n)x_{i,3})} = v_i^2$. Output reject and halt if any of them does not hold.
   (b) Let $t_i = 2^{-1} \bmod N_i$. Compute $\hat{m}_i = (e_i / u_i^{x_{i,1}})^{2t_i}$. If $\hat{m}_i$ is of the form $h_i^{m_i}$ for some $m_i \in [N_i]$, output a share (i, m_i); otherwise, output reject and halt.
2. (*Recovering Message*) Check that k shares (i, m_i) have been collected. If this does not hold, output reject and halt. Otherwise, interpolate a polynomial over $GF(N_0)$ of degree no greater than $k-1$ by requiring f to pass through the points $(i, m_i \bmod N_0)$. Output $f(0)$ as the recovered message.

It is easy for a group member to tell whether he is designated or not as the non-designated members output reject during the *Recovering Shares* phase of decryption with an overwhelming probability.

In the *Recovering Message* phase of decryption, the k shares (i, m_i) have to be collected. To do this, the cooperating members can broadcast their shares securely to one another, or send their shares to a trusted party. Yet another way to do so is to compute $f(0)$ interactively over many rounds. To add robustness, one may require the members to submit their shares together with proofs that the shares are decrypted correctly. Details omitted.

Proposition 1. *The scheme above is indistinguishable against adaptive chosen ciphertext and public key attacks provided that the DCR assumption holds and all keyed hash functions H_i's are collision resistant.*

The DCR assumption is reviewed in Appendix B. This proposition follows directly from [8, Theorem 1]. To see this, notice that every ciphertext component ctxt_i, $i \in \mathcal{T}$ for some designated decryptor group, of a ciphertext ψ is the Camenisch-Shoup encryption [9] of a share of the message. In addition, for $i \in [1,n] \setminus \mathcal{T}$, the ciphertext component ctxt_i does not contain any information of the message.

Theorem 1. *The scheme above is designated decryptor anonymous provided that the DCR assumption holds.*

Proof is given in Appendix C.

4 UCH-VE: Definition and Security Model

Let $\mathcal{R}$ be a relation defined by a generation algorithm $\mathcal{G}'$ which on input 1^{ℓ_0} outputs a description $\Psi = \Psi[\mathcal{R}, W, \Delta]$ of a binary relation $\mathcal{R}$ on $W \times \Delta$. We require that the sets $\mathcal{R}$, W and Δ are easy to recognize by given Ψ. For $\delta \in \Delta$, an element $w \in W$ such that $(w, \delta) \in \mathcal{R}$ is called a *witness* for δ.

A *Universal Custodian-Hiding Verifiable Encryption* (UCH-VE) proceeds as follows. Given a value δ, a witness w for δ and a label L, an encryptor chooses a set $\{\mathsf{PK}_i\}_{1 \le i \le n}$ of n public keys, two integers k, t such that $1 \le k \le t \le n$, and a t-element *designated decryptor group* $\mathcal{T} \in \wp_t([1, n])$, and encrypts w with L to a ciphertext ψ. Similar to a conventional UCH-Enc scheme, ψ decrypts to w when k (or more) private keys corresponding to decryptors in $\mathcal{T}$ are given.

In addition to the encryption and decryption of a witness for an element $\delta \in \Delta$, the encryptor can also prove to a public verifier that the pair of ciphertext and label (ψ, L) decrypts to a witness for δ under k (or more) private keys of some designated t (or more) private keys corresponding to $\{\mathsf{PK}_i\}_{1 \le i \le n}$. For carrying out the proof, the verifier is given the value of δ, the label L, a set $\{\mathsf{PK}_i\}_{1 \le i \le n}$ of n public keys and a ciphertext ψ. The verifier will output accept or reject at the end of the proof. When the verifier outputs accept, we say that the verifier is convinced that the proofing statement made by the encryptor above (also known as the prover) is correct.

Formally, a UCH-VE consists of a tuple of four PPT algorithms and one protocol. The four PPT algorithms are $\mathsf{System\ Setup}$, $\mathsf{Key\ Generation}$, $\mathsf{Encryption}$ and $\mathsf{Decryption}$. They are similar to that of a UCH-Enc scheme specified in Sec. 2. There are only a few changes need to be made: First, in the $\mathsf{System\ Setup}$ phase, there may be some auxiliary parameters generated for carrying out the $\mathsf{Verification\ Protocol}$ described below. Note that auxiliary parameters can be null if they are not needed. Second, the message now becomes a witness in W. Therefore, it is required that $W \subseteq \mathcal{M}$ where $\mathcal{M}$ is the message space.

The protocol called $\mathsf{Verification\ Protocol}$ is carried out between two interactive PPT algorithms: a *prover* and a *verifier*.

Verification Protocol: The common inputs of the prover and the verifier are

1. the system-wide security parameter ℓ_0 and auxiliary parameters generated in the $\mathsf{System\ Setup}$ phase,
2. a relation description $\Psi = \Psi[\mathcal{R}, W, \Delta]$,
3. an element $\delta \in \Delta$,
4. the set $\{\mathsf{PK}_i\}_{1 \le i \le n}$ of n public keys,
5. two integers k, t such that $1 \le k \le t \le n$, and
6. a ciphertext ψ with label $L \in \{0,1\}^*$.

The prover has the following additional inputs:

1. a witness w for δ, such that $(w, \delta) \in \mathcal{R}$,
2. a designated decryptor group $\mathcal{T} \in \wp_t([1, n])$, and
3. the random coins *coins* that were used by the Encryption algorithm for generating ψ.

At the end of the protocol, the verifier outputs either accept or reject while the prover has no output.

The security requirements of the underlying encryption scheme composed by System Setup, Key Generation, Encryption and Decryption are the same as that of a secure UCH-Enc scheme specified in Sec. 2.1. For the Verification Protocol, the additional requirements are (1) Verification Correctness, (2) Soundness and (3) Special Honest-Verifier Zero Knowledge. Details are given in Appendix D.

5 A UCH-VE Scheme for Discrete Logarithms

We start with the UCH-Enc scheme described in Sec. 3 and design a suitable proof of knowledge system as the Verification Protocol. Let Γ be a cyclic group of prime order ρ generated by γ. We assume that γ and ρ are publicly known. Let $W = [\rho]$ and $\Delta = \Gamma$, and let $\mathcal{R} = \{(w, \delta) \in W \times \Delta : \gamma^w = \delta\}$.

We now slightly modify our UCH-Enc scheme given in Sec. 3 by adding some additional parameters. These parameters will solely be used in the Verification Protocol. They do not affect the encryption and decryption processes of the original scheme, and the security claims on the encryption namely correctness, indistinguishability against adaptive chosen ciphertext and public key attacks, and designated decryptor anonymity, still hold.

System Setup: Let ℓ_0 be the system-wide security parameter. There are two parts in this phase. The first part is the same as that of the UCH-Enc scheme. We run the SystemSetup algorithm described in Sec. 3 on input 1^{ℓ_0} to generate the group size n, auxiliary security parameters ℓ_i $(1 \leq i \leq n)$ and the specification of a message space $\mathcal{M}$. We require that $\mathcal{M} = [\rho]$ (i.e. set $N_0 = \rho$).

In the second part, two additional parameters, κ' and κ'', are generated from $\kappa'(\ell_0)$ and $\kappa''(\ell_0)$, where $2^{-\kappa'(\ell_0)}$ and $2^{-\kappa''(\ell_0)}$ are negligible functions. $\{0,1\}^{\kappa'}$ is going to be the "challenge space" of the verifier in the Verification Protocol and κ'' controls the quality of the zero-knowledge property.

Key Generation: In addition to the key generation steps described in the UCH-Enc scheme, the following steps are added for generating some auxiliary parameters: $\mathfrak{n}_i$, $\mathfrak{g}_i$ and $\mathfrak{h}_i$, for $i = 1, \cdots, n$.

1. Generate $\mathfrak{n}_i$ in the same way as generating N_i in Sec. 3. That is, randomly select two ℓ_0-bit Sophie Germain primes $\mathfrak{p}'_i$ and $\mathfrak{q}'_i$, with $\mathfrak{p}'_i \neq \mathfrak{q}'_i$, and compute $\mathfrak{p}_i = 2\mathfrak{p}'_i + 1$, $\mathfrak{q}_i = 2\mathfrak{q}'_i + 1$, $\mathfrak{n}_i = \mathfrak{p}_i\mathfrak{q}_i$ and $\mathfrak{n}'_i = \mathfrak{p}'_i\mathfrak{q}'_i$.
2. Set $\mathfrak{g}_i$ and $\mathfrak{h}_i$ to be two distinct generators of $\mathfrak{G}_{\mathfrak{n}'_i} \subset \mathbb{Z}^*_{\mathfrak{n}_i}$ where $\mathfrak{G}_{\mathfrak{n}'_i}$ is the subgroup of $\mathbb{Z}^*_{\mathfrak{n}_i}$ of order $\mathfrak{n}'_i$.

We may simply put $\mathfrak{n}_i = N_i$. In any event, it is required that the prover (in the Verification Protocol below) does not know the factorization of $\mathfrak{n}_i$. The public key of group member i now becomes $\mathsf{PK}_i = (\mathfrak{n}_i, \mathfrak{g}_i, \mathfrak{h}_i, N_i, h_i, g_i, y_{i,1}, y_{i,2}, y_{i,3}, H_i, \mathsf{hk}_i)$ and the private key remains the same, that is, $\mathsf{SK}_i = (x_{i,1}, x_{i,2}, x_{i,3})$.

We require that $2^{\kappa'} < \min(p'_i, q'_i, \mathfrak{p}'_i, \mathfrak{q}'_i, \rho)$. This means the value of the challenge (denoted by c_i in the Verification Protocol) will "fit" in all the possible orders of the involved elements (i.e. u_i, e_i, v_i, δ, δ_i and $\mathfrak{t}_i$ in the Verification Protocol) of their corresponding groups.

We also require that $\rho < \min(N_i 2^{-\kappa'-\kappa''-1}, \mathfrak{n}_i 2^{-\kappa'-\kappa''-3})$. It is obvious that we need $\rho < N_i$ in order to allow proper encryption of ρ (more accurately, encrypting its t shares) under the public keys of the group members. In order to make sure that ρ also fits in the auxiliary group $(\mathfrak{n}_i, \mathfrak{g}_i, \mathfrak{h}_i)$, we also need $\rho < \mathfrak{n}'_i$. Since the factorization of $\mathfrak{n}_i$ is unknown to the prover, the prover does not know the value of $\mathfrak{n}'_i$. The prover therefore makes an approximation of the group order by observing that $\mathfrak{n}'_i$ is always greater than $\mathfrak{n}_i/8$. Hence we need to restrict that $\rho < \mathfrak{n}_i 2^{-3}$ holds. Further note that in the Verification Protocol below, we require the value of m'_i, which is in the range of $[-\rho 2^{\kappa'+\kappa''}, \rho 2^{\kappa'+\kappa''}]$, to fit in the order of $\mathfrak{g}_i$. This implies that we need to further restrict that $\rho < \mathfrak{n}_i 2^{-\kappa'-\kappa''-3}$. Similarly, we should also restrict the value of ρ such that $\rho < N_i 2^{-\kappa'-\kappa''-1}$ for fitting m'_i into the order of h_i.

Encryption: For a message $m = w$ with label $L \in \{0,1\}^*$ where $w \in [\rho]$ is a witness of a group element $\delta \in \Gamma$, namely $\delta = \gamma^w$, the encryptor picks two integers k, t such that $1 \leq k \leq t \leq n$, and a t-element *designated decryptor group* $\mathcal{T} \in \wp_t([1,n])$. Then the encryption scheme of the UCH-Enc scheme described in Sec. 3 is invoked by ignoring the auxiliary parameters $(\mathfrak{n}_i, \mathfrak{g}_i, \mathfrak{h}_i)$ in each of the public keys PK_i's, $1 \leq i \leq n$. Notice that the polynomial $f(x)$ in the encryption scheme is now over $GF(\rho)$.

Decryption: This is same as that of the UCH-Enc scheme described in Sec. 3.

This completes the modification of the encryption scheme. Essentially, in the modification above, we have only added the auxiliary parameters $(\mathfrak{n}_i, \mathfrak{g}_i, \mathfrak{h}_i)$ to each of the public keys PK_i, $1 \leq i \leq n$, and made several checks of the relations among several domains. These changes are for carrying out the Verification Protocol and have no effect on the encryption scheme above in terms of its security.

We now describe the protocol which contributes to the verification part of the UCH-VE scheme. The protocol is carried out between a prover and a verifier.

Verification Protocol: The common inputs of the prover and the verifier are (1) the system-wide security parameter ℓ_0 and additional security parameters κ' and κ'', (2) group domain parameters (Γ, γ, ρ), (3) group element $\delta \in \Gamma$, (4) the set $\{\mathsf{PK}_i\}_{1 \leq i \leq n}$ of n public keys, and (5) ciphertext $\psi = (k, t, \mathsf{ctxt}_1, \cdots, \mathsf{ctxt}_n)$ with label L. Note that each $\mathsf{ctxt}_i = (u_i, e_i, v_i)$, $1 \leq i \leq n$.

The prover has the following additional inputs: (1) $m \in [\rho]$, (2) $\mathcal{T} \in \wp_t([1,n])$, (3) $r_i \in [N_i/4]$ for $i \in \mathcal{T}$, and (4) polynomial $f(x) = \sum_{j=0}^{k-1} a_j x^j$ of degree $(k-1)$ over $GF(\rho)$ such that $\delta = \gamma^m$, $f(0) = m$, and the following are satisfied for all $i \in \mathcal{T}$:

$$u_i = g_i^{r_i}, \quad e_i = y_{i,1}^{r_i} h_i^{f(i)}, \quad \text{and} \quad v_i = \mathrm{abs}_i \left((y_{i,2} y_{i,3}^{H_{i,\mathsf{hk}_i}(u_i, e_i, L, k, t, n)})^{r_i} \right).$$

The protocol proceeds as follows.

1. (*Auxiliary Commitment*) For $i = 1, \cdots, n$, the prover chooses at random $s_i \in_R [\mathfrak{n}_i/4]$, and computes $\mathfrak{t}_i = \mathfrak{g}_i^{f(i)} \mathfrak{h}_i^{s_i}$ and $\delta_i = \gamma^{f(i)}$. The prover further computes $A_j = \gamma^{a_j}$ for $1 \le j \le k-1$. The prover sends $\langle (\delta_1, \mathfrak{t}_1), \cdots, (\delta_n, \mathfrak{t}_n) \rangle$ and $\langle A_1, \cdots, A_{k-1} \rangle$ to the verifier.
2. (*Auxiliary Verification*) For $i = 1, \cdots, n$, the verifier checks if

$$\delta_i = \delta \prod_{j=1}^{k-1} A_j^{i^j}.$$

   If any of them does not hold, the verifier stops and outputs reject.
3. (*Commitment*)
   (a) For $i = 1, \cdots, n$, the prover chooses random

$$r_i' \in_R [-N_i 2^{\kappa' + \kappa'' - 2}, N_i 2^{\kappa' + \kappa'' - 2}], \quad s_i' \in_R [-\mathfrak{n}_i 2^{\kappa' + \kappa'' - 2}, \mathfrak{n}_i 2^{\kappa' + \kappa'' - 2}],$$

$$m_i' \in_R [-\rho 2^{\kappa' + \kappa''}, \rho 2^{\kappa' + \kappa''}].$$

   (b) For $i \in \mathcal{T}$, the prover computes

$$u_i' = g_i^{2r_i'}, \quad e_i' = y_{i,1}^{2r_i'} h_i^{2m_i'}, \quad v_i' = (y_{i,2} y_{i,3}^{H_{i,\mathsf{hk}_i}(u_i, e_i, L, k, t, n)})^{2r_i'},$$

$$\delta_i' = \gamma^{m_i'}, \quad \mathfrak{t}_i' = \mathfrak{g}_i^{m_i'} \mathfrak{h}_i^{s_i'}.$$

   (c) For $i \in [1, n] \backslash \mathcal{T}$, the prover chooses random $c_i \in_R \{0,1\}^{\kappa'}$, $r_i \in [N_i/4]$, computes $\tilde{r}_i = r_i' - c_i r_i$, $\tilde{s}_i = s_i' - c_i s_i$, $\tilde{m}_i = m_i' - c_i m_i$ in $\mathbb{Z}$, and computes

$$u_i' = u_i^{2c_i} g_i^{2\tilde{r}_i}, \quad e_i' = e_i^{2c_i} y_{i,1}^{2\tilde{r}_i} h_i^{2\tilde{m}_i}, \quad v_i' = v_i^{2c_i} (y_{i,2} y_{i,3}^{H_{i,\mathsf{hk}_i}(u_i, e_i, L, k, t, n)})^{2\tilde{r}_i},$$

$$\delta_i' = \delta_i^{c_i} \gamma^{\tilde{m}_i}, \quad \mathfrak{t}_i' = \mathfrak{t}_i^c \mathfrak{g}_i^{\tilde{m}_i} \mathfrak{h}_i^{\tilde{s}_i}.$$

   The prover sends $\langle (u_1', e_1', v_1', \delta_1', \mathfrak{t}_1'), \cdots, (u_n', e_n', v_n', \delta_n', \mathfrak{t}_n') \rangle$ to the verifier.
4. (*Challenge*) The verifier chooses a random challenge $c \in_R \{0,1\}^{\kappa'}$ and sends $\langle c \rangle$ to the prover.
5. (*Response*) The prover generates a polynomial $\hat{f}$ of degree at most $(n-t)$ over $GF(2^{\kappa'})$ such that $\hat{f}(0) = c$, $\hat{f}(i) = c_i$ for $i \in [1, n] \backslash \mathcal{T}$. For each $i \in \mathcal{T}$, the prover computes the following in $\mathbb{Z}$:

$$\tilde{r}_i = r_i' - \hat{f}(i) r_i, \quad \tilde{s}_i = s_i' - \hat{f}(i) s_i, \quad \tilde{m}_i = m_i' - \hat{f}(i) m_i \tag{1}$$

   The prover replies with $\langle \hat{f}, (\tilde{r}_1, \tilde{s}_1, \tilde{m}_1), \cdots, (\tilde{r}_n, \tilde{s}_n, \tilde{m}_n) \rangle$.
6. (*Verification*)
   (a) The verifier checks if $\hat{f}$ is a polynomial of degree at most $(n-t)$ over $GF(2^{\kappa'})$, $\hat{f}(0) = c$, and check, for all $i = 1, \cdots, n$, if $-N_i/4 < \tilde{m}_i < N_i/4$. If any of them does not hold, the verifier stops and outputs reject.

(b) For each $i = 1, \cdots, n$, the verifier checks whether the relations

$$u_i' = u_i^{2\hat{f}(i)} g_i^{2\tilde{r}_i}, \quad e_i' = e_i^{2\hat{f}(i)} y_{i,1}^{2\tilde{r}_i} h_i^{2\tilde{m}_i}, \quad v_i' = v_i^{2\hat{f}(i)} (y_{i,2} y_{i,3}^{H_{i,\mathsf{hk}_i}(u_i,e_i,L,k,t,n)})^{2\tilde{r}_i},$$

$$\delta_i' = \delta_i^{\hat{f}(i)} \gamma^{\tilde{m}_i}, \quad \mathfrak{t}_i' = \mathfrak{t}_i^{\hat{f}(i)} \mathfrak{g}_i^{\tilde{m}_i} \mathfrak{h}_i^{\tilde{s}_i}.$$

hold. If any of them does not hold, the verifier stops and outputs reject. If all the checks above go through, the verifier outputs accept.

Discussions. In the Verification Protocol above, Steps (1) and (2) establish the evidence of $f(i) = \log_\gamma \delta_i$, $1 \le i \le n$, and $f(0) = \log_\gamma \delta$ to the verifier. Since the degree of f is $k-1$, one can find the value of $f(0)$, that is a witness for δ, once at least k out of the n shares $(f(1), \cdots, f(n))$ are known. Also note that these two steps can be combined with Step (3) and (4), respectively, for making the proof system a truly three-move one.

Steps (3) to (6) contribute to the construction of the following proof system:

$$\begin{aligned} PK[\, ((r_i, m_i, s_i)_{,i\in[1,n]}) : \textstyle\bigvee_{\mathcal{T}\in\wp_t([1,n])} (\bigwedge_{i\in\mathcal{T}} (\\ u_i^2 = g_i^{2r_i} \ \wedge\ e_i^2 = y_{i,1}^{2r_i} h_i^{2m_i} \ \wedge\ v_i^2 = (y_{i,2} y_{i,3}^{H_{i,\mathsf{hk}_i}(u_i,e_i,L,k,t,n)})^{2r_i} \ \wedge \\ \delta_i = \gamma^{m_i} \ \wedge\ \mathfrak{t}_i = \mathfrak{g}^{m_i} \mathfrak{h}^{s_i} \ \wedge\ -N_i/2 < m_i < N_i/2))]. \end{aligned}$$

This protocol can also be thought as the generalization of the following:

$$\begin{aligned} PK[\, (r, m, s) : u^2 = g^{2r} \ \wedge\ e^2 = y_1^{2r} h^{2m} \ \wedge\ v^2 = (y_2 y_3^{\mathcal{H}_{\mathsf{hk}}(u,e,L)})^{2r} \ \wedge \\ \delta = \gamma^m \ \wedge\ \mathfrak{t} = \mathfrak{g}^m \mathfrak{h}^s \ \wedge\ -n/2 < m_i < n/2))], \end{aligned}$$

which appears as Step 2 of the protocol for verifiable encryption of discrete logarithms in [8].

These four steps allow the prover to convince the verifier that there is at least one designated t-element set $\mathcal{T} \in \wp_t([1,n])$ whose corresponding ciphertext components $\{\mathsf{ctxt}_i\}_{i\in\mathcal{T}}$ decrypt to $\{f(i) = \log_\gamma \delta_i\}_{i\in\mathcal{T}}$. Combining with the evidence established in Steps (1) and (2), the verifier is convinced that from the ciphertext $\psi = (k, t, \mathsf{ctxt}_1, \cdots, \mathsf{ctxt}_n)$, any k (or more) out of some designated t members of a group of n members decrypt their corresponding ciphertext components to get k (or more) (k,t)-shares of a witness of δ.

The entire Verification Protocol can be represented by:

$$\begin{aligned} PK[\, ((r_i, m_i, s_i)_{,i\in[1,n]}) : \textstyle\bigvee_{\mathcal{T}\in\wp_t([1,n])} (\\ \textstyle\bigwedge_{i\in\mathcal{T}} (\, u_i^2 = g_i^{2r_i} \ \wedge\ e_i^2 = y_{i,1}^{2r_i} h_i^{2m_i} \ \wedge\ v_i^2 = (y_{i,2} y_{i,3}^{H_{i,\mathsf{hk}_i}(u_i,e_i,L,k,t,n)})^{2r_i} \ \wedge \\ \delta_i = \gamma^{m_i} \ \wedge\ \mathfrak{t}_i = \mathfrak{g}^{m_i} \mathfrak{h}^{s_i} \ \wedge\ -N_i/2 < m_i < N_i/2) \\ \textstyle\bigwedge_{\mathcal{K}\in\wp_k\mathcal{T}} (\delta = \gamma^{\sum_{i\in\mathcal{K}} m_i \prod_{j\in\mathcal{K}, j\neq i} \frac{m_j}{m_j - m_i}}))]. \end{aligned}$$

To implement the UCH-VE scheme, one can set $\mathfrak{n}_i = N_i$ for all $1 \le i \le n$, set $|N_i| = 2048$, $|\rho| = 1024$, $\kappa' = 512$ and $\kappa'' = 80$.

Theorem 2. *The UCH-VE scheme above satisfies the soundness definition if the Strong RSA problem is hard.*

The Strong RSA problem is reviewed in Appendix B and the proof is given in Appendix E.

Theorem 3. *The UCH-VE scheme above is Special HVZK.*

Proof is given in Appendix F.

References

1. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In *Proc. EUROCRYPT 98*, pages 591–606. Springer-Verlag, 1998. LNCS Vol. 1403.
2. F. Bao. An efficient verifiable encryption scheme for encryption of discrete logarithms. In *Proc. Smart Card Research and Applications (CARDIS) 1998*, pages 213–220. Springer-Verlag, 2000. LNCS Vol. 1820.
3. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *Proc. ASIACRYPT 2001*, pages 566–582. Springer-Verlag, 2001. LNCS Vol. 2248.
4. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Proc. CRYPTO 96*, pages 1–15. Springer-Verlag, 1996. LNCS Vol. 1109.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.
6. J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *Proc. ASIACRYPT 2000*, pages 331–345. Springer-Verlag, 2000. LNCS Vol. 1976.
7. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocations. In *Proc. EUROCRYPT 2001*, pages 93–118. Springer-Verlag, 2001. LNCS Vol. 2045.
8. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. http://eprint.iacr.org/2002/161/, 2002.
9. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Proc. CRYPTO 2003*, pages 126–144. Springer-Verlag, 2003. LNCS Vol. 2729.
10. R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In *Proc. EUROCRYPT 99*, pages 90–106. Springer-Verlag, 1999. LNCS Vol. 1592.
11. Y. Desmedt and Y. Frankel. Threshold cryptosystem. In *Proc. CRYPTO 89*, pages 307–315. Springer-Verlag, 1989. LNCS Vol. 435.
12. J. Liu, V. Wei, and D. Wong. Custodian-hiding verifiable encryption. In *WISA 2004*, pages 54–67. Springer-Verlag, 2004. LNCS Vol. 3325.
13. C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Proc. CRYPTO 91*, pages 433–444. Springer, 1992. LNCS Vol. 576.
14. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *Proc. EUROCRYPT 98*, pages 1–16. Springer-Verlag, 1998. LNCS Vol. 1403.
15. M. Stadler. Publicly verifiable secret sharing. In *Proc. EUROCRYPT 96*, pages 191–199. Springer-Verlag, 1996. LNCS Vol. 1070.

A Games for Defining a Secure UCH-Enc Scheme

Game Confidentiality: "For a UCH-Enc scheme, consider the following game which is played by a simulator $\mathcal{S}_{\mathcal{A}}$ against a PPT adversary $\mathcal{A}$.

1. (*System Setup and Key Generation*) Let $\ell_0 \in \mathbb{N}$ be the system-wide security parameter. $\mathcal{S}_{\mathcal{A}}$ implements the System Setup algorithm and the Key Generation algorithm accordingly, and generates $\mathcal{N}$ public key pairs $(\mathsf{PK}_i, \mathsf{SK}_i)$, $1 \leq i \leq \mathcal{N}$, and the specification of a message space $\mathcal{M}$, where $\mathcal{N}$ is some polynomial in ℓ_0. In the rest of the game, we consider that all involved keys belong to this $\mathcal{N}$ pairs of keys.
2. (*Oracle Simulation*) $\mathcal{S}_{\mathcal{A}}$ equips two oracles which behave as follows.
   - *Decryption Oracle* takes a set $\{\mathsf{PK}_{i_1}, \cdots, \mathsf{PK}_{i_n}\}$ of n public keys where $\{i_1, \cdots, i_n\} \subseteq [1, \mathcal{N}]$, a set $\mathcal{K} \in \wp_k(\{i_1, \cdots, i_n\})$, a ciphertext ψ with label $L \in \{0,1\}^*$ and returns either reject for failure of decryption or a message $m \in \mathcal{M}$ by decrypting ψ under the set $\{\mathsf{SK}_i\}_{i \in \mathcal{K}}$ of private keys,
   - *Corruption Oracle* takes an integer i, $1 \leq i \leq \mathcal{N}$ and returns SK_i.
3. (*Probing Phase I*) $\mathcal{A}$ interacts with the decryption oracle and the corruption oracle in an arbitrary, adaptive fashion. This phase goes on for a polynomial amount of time, specified by $\mathcal{A}$.
4. (*Target-Selection Phase*) $\mathcal{A}$ selects two messages m_0 and m_1 from $\mathcal{M}$, along with a label $L^* \in \{0,1\}^*$, a set $\mathcal{PK}_{1..n} = \{\mathsf{PK}_{i_1}, \cdots, \mathsf{PK}_{i_n}\}$ of n public keys where $\{i_1, \cdots, i_n\} \subseteq [1, \mathcal{N}]$, two integers k, t such that $1 \leq k \leq t \leq n$ and a set $\mathcal{T} \in \wp_t(\{i_1, \cdots, i_n\})$, and sends $(m_0, m_1, L^*, \mathcal{PK}_{1..n}, k, t, \mathcal{T})$ to $\mathcal{S}_{\mathcal{A}}$. $\mathcal{S}_{\mathcal{A}}$ selects a random $b \in_R \{0,1\}$ and encrypts m_b with label L^*. The resulting ciphertext ψ^* is presented to $\mathcal{A}$.
5. (*Probing Phase II*) This phase is similar to *Probing Phase I*. The only difference is that the decryption oracle only responds to queries of which the pair of ciphertext and label is not (ψ^*, L^*) and the public key set is not $\mathcal{PK}_{1..n}$.
6. (*Guessing Phase*) $\mathcal{A}$ outputs a bit $\hat{b}$."

Let $\mathcal{C}_{\mathcal{T},k} = \{i \mid i \in \mathcal{T} \wedge \mathsf{SK}_i \text{ has been corrupted}\}$. We say that "$\mathsf{SK}_i$ has been corrupted" if $\mathcal{A}$ has queried the corruption oracle with i before. $\mathcal{A}$ *wins* the game if $\hat{b} = b$ and $|\mathcal{C}_{\mathcal{T},k}| < k$. Define

$$\mathbf{Adv}_{\mathcal{A}}(\ell_0) = |\Pr[\mathcal{A} \text{ wins the game}] - \frac{1}{2}|$$

to be the advantage of $\mathcal{A}$ over random guessing. A UCH-Enc scheme is indistinguishable against adaptive chosen ciphertext and public key attacks if for all PPT adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}(\ell_0) = \text{neg}(\ell_0)$.

Game Anonymity: "For a UCH-Enc scheme, consider the following game which is played by a simulator $\mathcal{S}_{\mathcal{B}}$ against a PPT adversary $\mathcal{B}$.

1. (*System Setup and Key Generation*)
2. (*Oracle Simulation*)
3. (*Probing Phase I*)
   The above three phases are the same as that of Game Confidentiality.
4. (*Target-Selection Phase*) $\mathcal{B}$ picks the following:
   - a set $\mathcal{PK}_{1..n} = \{\mathsf{PK}_{i_1}, \cdots, \mathsf{PK}_{i_n}\}$ of n public keys where $\{i_1, \cdots, i_n\} \subseteq [1, \mathcal{N}]$,
   - two integers k, t such that $1 \leq k \leq t \leq n$,
   - two sets $\mathcal{T}_0, \mathcal{T}_1 \in \wp_t(\{i_1, \cdots, i_n\})$ such that $|\mathcal{T}_0 \cap \mathcal{T}_1| < t$,
   - a message m from $\mathcal{M}$, along with a label $L^* \in \{0,1\}^*$,

   and sends $(m, L^*, \mathcal{PK}_{1..n}, k, t, \mathcal{T}_0, \mathcal{T}_1)$ to $\mathcal{S}_\mathcal{B}$. $\mathcal{S}_\mathcal{B}$ selects a random $b \in_R \{0,1\}$ and encrypts m with label L^* with respect to the designated decryptor group $\mathcal{T}_b$. The resulting ciphertext ψ^* is presented to $\mathcal{B}$.
5. (*Probing Phase II*) This phase is as that of Game Confidentiality. Essentially, $\mathcal{B}$ cannot query the decryption oracle with the pair of ciphertext and label (ψ^*, L^*) along with the public key set $\mathcal{PK}_{1..n}$.
6. (*Guessing Phase*) $\mathcal{A}$ outputs a bit $\hat{b}$."

Let $\mathcal{C}_{i_1..i_n} = \{j \mid j \in \{i_1, \cdots, i_n \wedge \mathsf{SK}_j \text{ has been corrupted}\}$. $\mathcal{B}$ *wins* the game if $\hat{b} = b$ and $\mathcal{C}_{i_1..i_n} \cap (\mathcal{T}_0 \cup \mathcal{T}_1) = \emptyset$ where $\emptyset$ denotes the empty set. Define

$$\mathbf{Adv}_\mathcal{B}(\ell_0) = |\Pr[\mathcal{B} \text{ wins the game}] - \frac{1}{2}|$$

to be the advantage of $\mathcal{B}$ over random guessing. A UCH-Enc scheme is designated decryptor anonymous if for all PPT adversary $\mathcal{B}$, $\mathbf{Adv}_\mathcal{B}(\ell_0) = \text{neg}(\ell_0)$.

B Assumptions

Paillier's Decision Composite Residuosity (DCR) Problem. Randomly select two ℓ_0-bit Sophie Germain primes p' and q', with $p' \neq q'$, and set $N = (2p'+1)(2q'+1)$. Consider the group $\mathbb{Z}^*_{N^2}$ and the subgroup $\mathbf{P}$ of $\mathbb{Z}^*_{N^2}$ consisting of all N-th powers of elements in $\mathbb{Z}^*_{N^2}$. Given N and a random element $u \in \mathbb{Z}^*_{N^2}$, determine if $u \in \mathbf{P}$.

Strong RSA Problem. Randomly select two ℓ_0-bit Sophie Germain primes p' and q', with $p' \neq q'$, and set $N = (2p'+1)(2q'+1)$. Given N and a random element $g \in \mathbb{Z}^*_N$, find an element $h \in \mathbb{Z}^*_n$ and an integer $e > 1$ such that $h^e = g$.

Fact 1. If the Strong RSA Problem is hard, then given N with random elements $g, h \in_R (\mathbb{Z}^*_N)^2$, it is hard to compute $w \in \mathbb{Z}^*_N$ and integers a, b, c such that

$$w^c = g^a h^b \quad \text{and } (c \not| a \text{ or } c \not| b).$$

The proof of Fact 1 can be found in the proof of [8, Theorem 3].

C Proof of Theorem 1 (Designated Decryptor Anonymity)

Proof. For contradiction, suppose an adversary $\mathcal{B}$ has non-negligible advantage of winning Game Anonymity. We show that the simulator $\mathcal{S}_{\mathcal{B}}$ of Game Anonymity can be used to solve the DCR problem.

Given a DCR problem instance $u^* \in \mathbb{Z}^*_{N^2}$, $\mathcal{S}_{\mathcal{B}}$ sets $N_i = N$ for all $i = 1, \cdots, \mathcal{N}$. Then all public key pairs are generated according to the scheme and all oracles are simulated properly according to Game Anonymity.

During the *Target-Selection Phase*, the ciphertext ψ^* is prepared such that u^* is used as the first component of $\mathsf{ctxt}_{j'}$ for some j' randomly chosen from $\mathcal{T}^*_{1-b} = \mathcal{T}_{1-b} - \mathcal{T}_b$. The remaining two components of $\mathsf{ctxt}_{j'}$ are generated just like as j' is also in the designated decryptor group $\mathcal{T}_b$. For other $j \in \mathcal{T}^*_{1-b} \setminus \{j'\}$, the first component of ctxt_j is generated just like as j is also in the designated decryptor group $\mathcal{T}_b$.

As we can see, all ciphertext components corresponding to receivers indexed by $\mathcal{T}_b$ and $\mathcal{T}_{1-b}$, except the one containing u^* (i.e. $\mathsf{ctxt}_{j'}$), are generated in identically the same way. If $u^* \in \mathbf{P}$, the adversary $\mathcal{B}$ has no advantage in guessing b as there is actually no difference between the ciphertext ensembles generated with respect to $\mathcal{T}_0$ or $\mathcal{T}_1$. The difference would only exist if $u^* \notin \mathbf{P}$. In this case, the ciphertext component with u^* as the first component in $\mathcal{T}_{1-b}$ is the only difference from other ciphertext components as the first components of all other ciphertext components are in $\mathbf{P}$. Hence if the adversary has non-negligible advantage, $\mathcal{S}_{\mathcal{B}}$ has non-negligible advantage of telling that u^* is not in $\mathbf{P}$. □

D The Security Models of a UCH-VE Scheme

Verification Correctness. For all n, ℓ_i $(1 \leq i \leq n)$, auxiliary parameters (if any), and the specification of $\mathcal{M}$ generated by System Setup with input ℓ_0; $\Psi[\mathcal{R}, W, \Delta]$ generated by $\mathcal{G}'(1^{\ell_0})$; $(w, \delta) \in \mathcal{R}$; key pairs $(\mathsf{PK}_i, \mathsf{SK}_i)$, $1 \leq i \leq n$, generated by the Key Generation; integers k, t such that $1 \leq k \leq t \leq n$; $\mathcal{T} \in \wp_t([1, n])$; random coins *coins*; and ciphertext ψ generated by Encryption with respect to w as the message and label $L \in \{0,1\}^*$ with the random coins *coins* during the encryption process, we have the following holds with probability at least $1 - \mathrm{neg}(\ell_0)$.

> "Given common inputs of ℓ_0, auxiliary parameters (if any), $\Psi[\mathcal{R}, W, \Delta]$, δ, $\{\mathsf{PK}_i\}_{1 \leq i \leq n}$, k, t and ψ to the prover and the verifier; given secret inputs of w, $\mathcal{T}$ and random coins *coins* to the prover, if the prover and the verifier interacts honestly, the verifier outputs accept and the prover outputs nothing at the end of the Verification Protocol."

Soundness. If a ciphertext is verified to be valid by an honest verifier, we require that there exists a designated decryptor group of t members, in which any k members can jointly recover a witness for δ from the ciphertext. On the

other hand, any $k-1$ (or less) members of the group cannot recover the witness. For ensuring that the prover cannot cheat, we define the following game and give the definition of soundness.

Game Soundness: "For a UCH-VE scheme, consider the following game, played by a simulator $\mathcal{S}_{\mathcal{P}}$ against two PPT adversaries $\mathcal{A}^*$ and $\mathcal{P}^*$.

1. (*System Setup and Key Generation*) Let $\ell_0 \in \mathbb{N}$ be the system-wide security parameter. $\mathcal{S}_{\mathcal{P}}$ invokes $\mathcal{G}'(1^{\ell_0})$ for a relation description $\Psi[\mathcal{R}, W, \Delta]$. Then, $\mathcal{S}_{\mathcal{P}}$ implements the System Setup algorithm and the Key Generation algorithm accordingly, and generates n public key pairs $(\mathsf{PK}_i, \mathsf{SK}_i)$, $1 \leq i \leq n$, auxiliary parameters (if any), and the specification of a message space $\mathcal{M}$, where n is some polynomial number of ℓ_0. In the remaining of the game, we consider that all involved keys belong to this n pairs of keys.
2. (*Target Generation Phase*) $\mathcal{A}^*$ takes all key pairs $\{(\mathsf{PK}_i, \mathsf{SK}_i)\}_{1 \leq i \leq n}$ and relation description Ψ, and produces an element $\delta \in \Delta$, a ciphertext ψ, a label L, two integers k, t such that $1 \leq k \leq t \leq n$ and some auxiliary string aux.
3. (*Interact With Malicious Prover*) $\mathcal{P}^*$ takes the auxiliary string aux as an input and interacts with the game simulator $\mathcal{S}_{\mathcal{P}}$ as in one run of the Verification Protocol. In this phase, $\mathcal{S}_{\mathcal{P}}$ simulates the verifier accordingly.
4. (*Verifier Output*) After completing the run of Verification Protocol, $\mathcal{S}_{\mathcal{P}}$ outputs what a honest verifier should output."

The adversaries $\mathcal{A}^*$ and $\mathcal{P}^*$ *win* the game if the game output is accept but

- $(w', \delta) \notin \mathcal{R}$ for all w' that is the decryption of ψ with respect to L under a set $\{\mathsf{SK}_i\}_{i \in \mathcal{K}_{i,j}}$ of private keys where $\mathcal{K}_{i,j} = \wp_k(\mathcal{T}_i)[j]$, for all $i \in [1, \binom{n}{t}]$ and $j \in [1, \binom{t}{k}]$, **or**
- $(w'', \delta) \in \mathcal{R}$ where w'' is the decryption of ψ with respect to L under the set $\{\mathsf{SK}_i\}_{i \in \overline{\mathcal{K}}}$ where $\overline{\mathcal{K}} \subseteq [1, n]$ and $|\overline{\mathcal{K}}| < k$.

A UCH-VE scheme is Sound if for all PPT adversaries $\mathcal{A}^*$ and $\mathcal{P}^*$, the probability that the adversaries win the game is equal to $\mathrm{neg}(\ell_0)$.

The model also captures the scenario that the prover and all decryptors are colluding. This is captured using the parameter aux as it can be used to pass all the secrets of the decryptors obtained in phase 2 (*Target Generation Phase*) to adversary $\mathcal{P}^*$ in phase 3 (*Interact With Malicious Prover*).

Special Honest-Verifier Zero Knowledge. For preventing the verifier from getting any useful information from the Verification Protocol, we require the protocol to be special honest-verifier zero knowledge (Special HVZK).

A Special HVZK protocol has two parties, a prover and a verifier. They have a common input y and the prover has an additional secret input x. The protocol is restricted to three moves. In the first move, the prover sends a 'commitment' t to the verifier. In the second move, the verifier sends a 'challenge' c back to the prover. In the third move, the prover sends a 'response' s to the prover. Also,

there must exist a simulator Sim that on input y and any 'challenge' $\tilde{c}$, outputs a 'commitment' $\tilde{t}$ and a 'response' $\tilde{s}$ such that the distribution of the triple $(\tilde{t}, \tilde{c}, \tilde{s})$ is indistinguishable from a triple (t, c, s) obtained from a real interaction between the prover and the verifier for which $c = \tilde{c}$.

We define the following game to capture the corresponding security requirement of UCH-VE schemes.

Game SHVZK: "For a UCH-VE scheme, consider the following game, played by a simulator $\mathcal{S}_{\mathcal{V}}$ against three PPT adversaries $\mathcal{A}^*$, $\mathcal{C}^*$ and $\mathcal{D}^*$.

1. (*System Setup and Key Generation*) This phase is the same as that of Game Soundness.
2. (*Target Selection Phase*) $\mathcal{A}^*$ takes all key pairs $\{(\mathsf{PK}_i, \mathsf{SK}_i)\}_{1 \leq i \leq n}$ and relation description Ψ, and produces a pair $(w, \delta) \in \mathcal{R}$, a label L, two integers k, t such that $1 \leq k \leq t \leq n$ and some auxiliary string aux.
3. (*Target Generation Phase*) $\mathcal{S}_{\mathcal{V}}$ selects random coins $coins$ and a t-element designated decryptor group $\mathcal{T} \in \wp_t([1, n])$, and prepares a ciphertext ψ under $\{\mathsf{PK}_i\}_{1 \leq i \leq n}$, $k, t, \mathcal{T}$ and w with L using $coins$ during the encryption process.
4. (*Challenge Phase*) $\mathcal{C}^*$ takes aux and ψ, and outputs a 'challenge' c.
5. (*Simulation Phase*) $\mathcal{S}_{\mathcal{V}}$ picks a random bit $b \in_R \{0, 1\}$.
   - If $b = 0$, then set $\alpha \leftarrow Trans(\langle \text{common inputs} \rangle, w, coins)$ where $Trans$ denotes the transcript seen by the verifier in a real interaction when c is used as the 'challenge'. $\langle \text{common inputs} \rangle$ denotes a list of public keys $\mathsf{PK}_1, \cdots, \mathsf{PK}_n$, auxiliary parameters (if any), Ψ, δ, k, t, ψ, L and c.
   - If $b = 1$, then set $\alpha \leftarrow Sim(\langle \text{common inputs} \rangle)$ when the 'challenge' is c.
6. (*Guessing Phase*) Given aux, ψ and α, $\mathcal{D}^*$ is to output a bit $\hat{b}$.

The adversaries $\mathcal{A}^*$, $\mathcal{C}^*$ and $\mathcal{D}^*$ *win* the game if $b = \hat{b}$. A UCH-VE scheme is Special HVZK if for all PPT adversaries $\mathcal{A}^*$, $\mathcal{C}^*$ and $\mathcal{D}^*$, the probability that the adversaries win the game is equal to $\frac{1}{2} + \text{neg}(\ell_0)$.

E Proof of Theorem 2 (Soundness)

Proof. If there exists some PPT adversaries $\mathcal{A}^*$ and $\mathcal{P}^*$ who win the Game Soundness with non-negligible probability, then there exists a *Knowledge Extractor* which produces (with overwhelming probability) two responses:

$$(\hat{f}^{(1)}, (\tilde{r}_1^{(1)}, \tilde{s}_1^{(1)}, \tilde{m}_1^{(1)}), \ldots, (\tilde{r}_n^{(1)}, \tilde{s}_n^{(1)}, \tilde{m}_n^{(1)})),$$

$$(\hat{f}^{(2)}, (\tilde{r}_1^{(2)}, \tilde{s}_1^{(2)}, \tilde{m}_1^{(2)}), \ldots, (\tilde{r}_n^{(2)}, \tilde{s}_n^{(2)}, \tilde{m}_n^{(2)}))$$

on two different challenges: $c^{(1)}$, $c^{(2)}$ with respect to the same commitment:

$$\langle (\delta_1, \mathfrak{t}_1), \cdots, (\delta_n, \mathfrak{t}_n), A_1, \cdots, A_{k-1}, (u'_1, e'_1, v'_1, \delta'_1, \mathfrak{t}'_1), \cdots, (u'_n, e'_n, v'_n, \delta'_n, \mathfrak{t}'_n) \rangle.$$

If $\hat{f}^{(1)}$ and $\hat{f}^{(2)}$ have more than $(n-t)$ points in common, then they must be equal as they are of degree at most $(n-t)$. Therefore $c^{(1)} \neq c^{(2)}$ implies $\hat{f}^{(1)} \neq \hat{f}^{(2)}$

which in turn implies that there are at least t distinct indices $\pi_1, \cdots, \pi_t \in [1, n]$ for which $\hat{f}^{(1)}(\pi_i) \neq \hat{f}^{(2)}(\pi_i)$ for all $i \in [1, t]$.

For $i = 1, \cdots, t$, let $\Delta r_{\pi_i} = \tilde{r}^{(1)}_{\pi_i} - \tilde{r}^{(2)}_{\pi_i}$, $\Delta s_{\pi_i} = \tilde{s}^{(1)}_{\pi_i} - \tilde{s}^{(2)}_{\pi_i}$, $\Delta m_{\pi_i} = \tilde{m}^{(1)}_{\pi_i} - \tilde{m}^{(2)}_{\pi_i}$, and $\Delta c_{\pi_i} = \hat{f}^{(1)}(\pi_i) - \hat{f}^{(2)}(\pi_i)$. From the verification equations, one can derive the following equations.

$$u_{\pi_i}^{2\Delta c_{\pi_i}} = g_{\pi_i}^{2\Delta r_{\pi_i}}, \quad e_{\pi_i}^{2\Delta c_{\pi_i}} = y_{\pi_i,1}^{2\Delta r_{\pi_i}} h_{\pi_i}^{2\Delta m_{\pi_i}},$$

$$v_{\pi_i}^{2\Delta c_{\pi_i}} = (y_{\pi_i,2} y_{\pi_i,3}^{H_{\pi_i,\mathsf{hk}_{\pi_i}}(u_{\pi_i}, e_{\pi_i}, L, k, t, n)})^{2\Delta r_{\pi_i}},$$

$$\delta_{\pi_i}^{\Delta c_{\pi_i}} = \gamma^{\Delta m_{\pi_i}}, \quad \mathfrak{t}_{\pi_i}^{\Delta c_{\pi_i}} = \mathfrak{g}_{\pi_i}^{\Delta m_{\pi_i}} \mathfrak{h}_{\pi_i}^{\Delta s_{\pi_i}}.$$

By **Fact 1** (Appendix B), since the Knowledge Extractor has computed $\mathfrak{t}_{\pi_i}$, Δc_{π_i}, Δm_{π_i}, and Δs_{π_i} such that $\mathfrak{t}_{\pi_i}^{\Delta c_{\pi_i}} = \mathfrak{g}_{\pi_i}^{\Delta m_{\pi_i}} \mathfrak{h}_{\pi_i}^{\Delta s_{\pi_i}}$, then $\Delta c_{\pi_i} | \Delta m_{\pi_i}$ and $\Delta c_{\pi_i} | \Delta s_{\pi_i}$.

By construction, we have the length of Δc_{π_i} to be at most κ' and $2^{\kappa'} < \min(p'_{\pi_i}, q'_{\pi_i}, \mathfrak{p}'_{\pi_i}, \mathfrak{q}'_{\pi_i}, \rho)$. This implies that Δc_{π_i} is relatively prime to $N_{\pi_i} N'_{\pi_i}$. Let $\hat{c}_{\pi_i} = \Delta c_{\pi_i}^{-1} \bmod N_{\pi_i} N'_{\pi_i}$. Since $\phi(N^2_{\pi_i}) = 4 N_{\pi_i} N'_{\pi_i}$ and $u^2_{\pi_i}$ has order dividing $N_{\pi_i} N'_{\pi_i}$, we have $u^2_{\pi_i} = g_{\pi_i}^{2\Delta r_{\pi_i} \hat{c}_{\pi_i}}$. That is,

$$u_{\pi_i} = w_{\pi_i,1} g_{\pi_i}^{\Delta r_{\pi_i} \hat{c}_{\pi_i}} \tag{2}$$

for some w_1 of order 2. Similarly, we get

$$e_{\pi_i} = w_2 y_{\pi_i,1}^{\Delta r_{\pi_i} \hat{c}_{\pi_i}} h_{\pi_i}^{\Delta m_{\pi_i} / \Delta c_{\pi_i}} \tag{3}$$

$$v_{\pi_i} = w_3 (y_{\pi_i,2} y_{\pi_i,3}^{H_{\pi_i,\mathsf{hk}_{\pi_i}}(u_{\pi_i}, e_{\pi_i}, L, k, t, n)})^{\Delta r_{\pi_i} \hat{c}_{\pi_i}} \tag{4}$$

$$\delta_{\pi_i} = \gamma^{\Delta m_{\pi_i} / \Delta c_{\pi_i}} \tag{5}$$

for some w_2 and w_3 of order 2.

Notice that from $v_{\pi_i} = \mathrm{abs}_{\pi_i}(v_{\pi_i})$ and Eqns. (2) - (4), we can see that the decryption of $(u_{\pi_i}, e_{\pi_i}, v_{\pi_i})$ will provide the integer $\bar{m}_{\pi_i} \in [N_{\pi_i}]$, which is equal to $\Delta m_{\pi_i} / \Delta c_{\pi_i} \bmod N_{\pi_i}$. The terms of w_1, w_2 and w_3 are gone due to the squarings in the decryption algorithm.

As $-N_{\pi_i}/4 < \tilde{m}^{(1)}_{\pi_i}, \tilde{m}^{(2)}_{\pi_i} < N_{\pi_i}/4$ and $\Delta c_{\pi_i} | \Delta m_{\pi_i}$, we must have $-N_{\pi_i}/2 < \Delta m_{\pi_i} / \Delta c_{\pi_i} < N_{\pi_i}/2$. Note that $\log_\gamma \delta_{\pi_i} = \Delta m_{\pi_i} / \Delta c_{\pi_i}$. Hence the relation between $\log_\gamma \delta_{\pi_i}$ and the plaintext $\bar{m}_{\pi_i}$ encrypted in $(u_{\pi_i}, e_{\pi_i}, v_{\pi_i})$ is that $\log_\gamma \delta_{\pi_i} = \bar{m}_{\pi_i} \bmod N_{\pi_i}$, where $\bar{m}_{\pi_i} \bmod N_{\pi_i} = \bar{m}_{\pi_i} - \lceil \bar{m}_{\pi_i} / N_{\pi_i} \rfloor N_{\pi_i}$, which is called the balanced remainder.

In other words, based on **Fact 1** in Appendix B (due to the assumption that the Strong RSA Problem is hard), there exists a Knowledge Extractor which can compute, the plaintext of the encryption $(u_{\pi_i}, e_{\pi_i}, v_{\pi_i})$ and ensure that its balanced remainder is equal to $m_{\pi_i} = \log_\gamma \delta_{\pi_i}$, for all $i \in [1, t]$, provided that $\mathcal{A}^*$ and $\mathcal{P}^*$ win Game Soundness.

We have only shown that the adversaries have to know at least t discrete logarithms of the n committed δ_i's and these values are encrypted in the ciphertext

components properly. Next, we show that any k of these t values can be used to reconstruct the discrete logarithm of δ.

Now $\delta_{\pi_i} = \prod_{j=0}^{k-1} A_j^{\pi_i^j}$ implies that $\log_\gamma \delta_{\pi_i} = \sum_{j=0}^{k-1} \pi_i^j \log_\gamma A_j$. Hence for $i = 1, \cdots, t$, (π_i, m_{π_i}) are points of some polynomial $\bar{f}(x) = \sum_{j=0}^{k-1} x^j \log_\gamma A_j$ of degree $(k-1)$ over $GF(\rho)$. Therefore, any k of the t m_{π_i}'s can be used to reconstruct $\bar{f}(0) = \log_\gamma A_0 = \log_\gamma \delta \in [\rho]$. This completes the proof. □

F Proof of Theorem 3 (Special Honest-Verifier Zero-Knowledge)

Proof. In the following, we consider that Step (1) and (2) of our Verification Protocol are combined with Step (3) and (4), respectively. This follows that the protocol is three-move. In the first move, the prover sends the commitment

$$\langle (\delta_1, \mathfrak{t}_1), \cdots, (\delta_n, \mathfrak{t}_n), A_1, \cdots, A_{k-1}, (u'_1, e'_1, v'_1, \delta'_1, \mathfrak{t}'_1), \cdots, (u'_n, e'_n, v'_n, \delta'_n, \mathfrak{t}'_n) \rangle$$

to the verifier. In the second move, the verifier sends the challenge, $\langle c \rangle$, back to the prover. In the third move, the prover sends the response

$$\langle \hat{f}, (\tilde{r}_1, \tilde{s}_1, \tilde{m}_1), \cdots, (\tilde{r}_n, \tilde{s}_n, \tilde{m}_n) \rangle$$

to the prover.

We now construct a simulator Sim which simulates an equated protocol transcript as follows.

The input of Sim is the set of common inputs specified in our Verification Protocol on page 398. For a 'fixed' challenge $c \in \{0,1\}^{\kappa'}$, Sim randomly picks $A_j \in_R \Gamma$ for $1 \leq j \leq k-1$, computes $\delta_i = \delta \prod_{j=1}^{k-1} A_j^{i^j}$ for $i = 1, \cdots, n$, randomly picks $\mathfrak{t}_i$ from its corresponding domain for $i = 1, \cdots, n$, randomly generates a polynomial $\hat{f}$ of degree at most $(n-t)$ over $GF(2^{\kappa'})$ provided that $\hat{f}(0) = c$ (that is, sets $a_0 = c$ and randomly picks $a_j \in_R [\rho]$ for $j = 1, \cdots, k-1$), and computes $c_i = \hat{f}(i)$ for all $i = 1, \cdots, n$. For all $i = 1, \cdots, n$, randomly picks

$$r'_i \in_R [-N_i 2^{\kappa'+\kappa''-2}, N_i 2^{\kappa'+\kappa''-2}], \quad s'_i \in_R [-\mathfrak{n}_i 2^{\kappa'+\kappa''-2}, \mathfrak{n}_i 2^{\kappa'+\kappa''-2}],$$

$$m'_i \in_R [-\rho 2^{\kappa'+\kappa''}, \rho 2^{\kappa'+\kappa''}], \quad s_i \in_R [\mathfrak{n}_i/4], \quad m_i \in_R [\rho],$$

computes $\tilde{r}_i = r'_i - c_i r_i$, $\tilde{s}_i = s'_i - c_i s_i$, $\tilde{m}_i = m'_i - c_i m_i$, and computes

$$u'_i := u_i^{2c_i} g_i^{2\tilde{r}_i}, \quad e'_i := e_i^{2c_i} y_{i,1}^{2\tilde{r}_i} h_i^{2\tilde{m}_i}, \quad v'_i := v_i^{2c_i} (y_{i,2} y_{i,3}^{H_{i,\mathsf{hk}_i}(u_i, e_i, L, k, t, n)})^{2\tilde{r}_i},$$

$$\delta'_i := \delta_i^{c_i} \gamma^{\tilde{m}_i}, \quad \mathfrak{t}'_i := \mathfrak{t}_i^{c_i} \mathfrak{g}_i^{\tilde{m}_i} \mathfrak{h}_i^{\tilde{s}_i}.$$

The simulated transcript is:

$$\langle (\delta_1, \mathfrak{t}_1), \cdots, (\delta_n, \mathfrak{t}_n), A_1, \cdots, A_{k-1}, (u'_1, e'_1, v'_1, \delta'_1, \mathfrak{t}'_1), \cdots, (u'_n, e'_n, v'_n, \delta'_n, \mathfrak{t}'_n), c, \hat{f}, (\tilde{r}_1, \tilde{s}_1, \tilde{m}_1), \cdots, (\tilde{r}_n, \tilde{s}_n, \tilde{m}_n) \rangle.$$

We finally proceed to show that the simulated transcript is indistinguishable from the real protocol transcript. We compare the distribution of the simulated transcript with that of the transcript of a real interaction.

Note that the coefficients a_i, $1 \leq i \leq k-1$, of f are randomly chosen over $[\rho]$. A_i, $1 \leq i \leq k-1$, are uniformly distributed over Γ in a real transcript. The distributions of A_i's in a simulated transcript are therefore identical to that in a real transcript. Since the generations of δ_i's in the simulated transcript are the same as that in the real transcript and they are functions of A_i's, the distributions of δ_i's in both simulated and real transcripts are also identical. Similarly, the generations of $\mathfrak{t}_i$'s in the simulated transcript can also be an identical copycat of that in the real transcript. Hence the distributions are also identical.

For each $i = 1, \cdots, n$, $(u'_i, e'_i, v'_i, \delta'_i, \mathfrak{t}'_i)$ are all distributed over their corresponding domains with negligible derivation from uniform distribution in a real transcript. Next, we show that $\hat{f}$ is a random polynomial of degree at most $(n-t)$ over $GF(2^{\kappa'})$ with $\hat{f}(0) = c$. Note that $\hat{f}$ is uniquely determined by the fixed challenge c and $n-t$ values of c_i's. As each c_i is randomly chosen from $\{0,1\}^{\kappa'}$, for $i \in [1,n] \backslash \mathcal{T}$, $\hat{f}$ is a random polynomial of degree at most $(n-t)$ over $GF(2^{\kappa'})$ provided that $\hat{f}(0) = c$. Finally, we can see that $(\tilde{r}_i, \tilde{s}_i, \tilde{m}_i)$ are also identically simulated in the simulated transcript.

Since the simulated transcript has identical distribution to that of a real transcript, we can easily implement Game SHVZK with a PPT which completes the game with non-negligible probability in such a way that for any PPT adversaries $\mathcal{A}^*$, $\mathcal{C}^*$ and $\mathcal{D}^*$, the chance of winning Game SHVZK is $\frac{1}{2}$. □

An Efficient Static Blind Ring Signature Scheme*

Qianhong Wu[1], Fanguo Zhang[2], Willy Susilo[1], and Yi Mu[1]

[1] Center for Information Security Research,
School of Information Technology and Computer Science,
University of Wollongong,
Wollongong NSW 2522, Australia
{qhw, wsusilo, ymu}@uow.edu.au
[2] School of Information Science and Technology,
Sun Yat-sen University, Guangzhou 510275,
Guangdong Province, P.R. China
isdzhfg@zsu.edu.cn

Abstract. Blind group/ring signatures are useful for applications such as e-cash and e-voting systems. In this paper, we show that the blindness of some existing blind group/ring signature schemes is easy to break by a malicious anonymous signer of dynamic groups. However, this risk has not been pointed out in these proposals, which may cause misuse of the schemes. Fortunately, for static groups, it is possible to integrate the blindness of message into group/ring signatures. An efficient static blind ring signature is proposed with its security provable under the extended ROS assumptions in the random oracle model plus the generic group model. After the group public key is generated, the space, time, and communication complexities of the relevant parameters and operations are constant.

1 Introduction

1.1 Group Signatures

In a group signature scheme, the members of a given group are allowed to sign on behalf of the entire group. The signatures can be verified using a single group public key. Once a document is signed, no one, except a designated group manager, can determine which member of the group signed it.

Group signatures were first introduced and implemented by Chaum and van Heyst [8]. Camenisch and Stadler [12] presented the first group signature scheme in which the size of the group public key remains independent of the group size, as do the time, space, and, communication complexities of the necessary operations. Some recent proposals support revocation ([11], [24]) where group membership can be selectively disabled without affecting the signing ability of unrevoked

* This work is supported by ARC Discovery Grant DP0557493 and the National Natural Science Foundation of China (No. 60403007).

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 410–423, 2006.
© Springer-Verlag Berlin Heidelberg 2006

members. Currently, the most efficient constructions ([1], [11]) are based on the Strong-RSA assumption introduced by Baric and Pfitzman [4], except that the schemes in [3] are based on the newly introduced DLP-related assumptions from bilinear pairings.

1.2 Ring Signatures

Compared with group signatures, in a ring signature scheme, there exists no third trusted party such as a group manager to setup the system or trace the identity of a group signer, but it is assumed that every user has a PKI-based public key correctly generated following the specifications of the system. A ring signature convinces one that the signature is anonymously generated by at least one member in an ad hoc group.

The notion of ring signatures was introduced by Rivest, *et al.* [21] in 2001. It has attracted a lot of attentions since its inception([2], [5], [17], [27]). The ring signatures in [13] and [26] are the first constant-sized ring signature schemes independently proposed by Dodis *et al* and Wu *et al*, respectively. The threshold ring signature is a natural extension of ring signatures [5]. In a (t, n)-threshold ring signature scheme, the generation of a ring signature for a group of n members requires the involvement of at least t members/signers, and yet the signature reveals nothing about the identities of the signers. For special applications, linkable ring signatures are proposed [17] so that anyone can determine whether two anonymous ring signatures are signed by the same group member (in which case the two signatures are said to be *linked*). If a user signs only once on behalf of a group, the user still enjoys anonymity similar to that in the conventional ring signature schemes. If the user signs multiple times, anyone can tell that these signatures have been generated by the same group member.

1.3 Blind Signatures

Blind digital signatures were introduced by Chaum [9]. It has been used in numerous applications, most prominently in anonymous voting schemes and anonymous e-cash systems. In the electronic cash scenario, a document corresponds to an electronic coin, and the signer represents a bank. The spender retains anonymity in any transaction that involves electronic coins since they are blindly signed.

Informally speaking, blind signature schemes allow a user to obtain signatures from an authority on any document, in such a way that the authority learns nothing about the message that is being signed. A bit more formally, a signer S with public key pk and secret key sk interacts with user U having m as its private input. At the end of the interaction, the user obtains a signature σ on m. Two seemingly contradictory properties must be satisfied. The first property, termed blindness, requires that after interacting with various users, the signer S is not able to link a valid message-signature pair (m, σ) obtained by some user, with the protocol session during which σ was created. The second security property, termed unforgeability, requires that it be impossible for any malicious

user that engages in k runs of the protocol with the signer, to obtain strictly more than k valid message-signature pairs. These security notions were formalized in [22] building on previous work [31], [20]. Most of the blind signatures base their security on the random oracle model. The scheme in [10] is the provably secure blind signature scheme in the standard model which is also efficient.

1.4 Comment on Dynamic Blind Group/Ring Signatures

Several blind group signatures [16], [19] and blind ring signatures [6] have been proposed. They have been shown as a powerful tool for applications such as multi-bank electronic cash or e-voting systems. However, we point out all these schemes are subject to the following chosen group-public-key attack if they are implemented for dynamic groups. The blindness can be easily broken and the user who submits blind messages to be anonymously signed can be traced.

Our comment is based on the following simple observation. In the group/ring signature schemes applying to dynamic groups, the group public keys, i.e., the verification keys, contain some dynamic information related to the change of the group members. When innocent users broadcast blind messages to be signed to dynamic groups, a malicious anonymous group/ring signer can interact with the users and generate valid blind group/ring signatures from different groups, in the meanwhile, the malicious signer creates a secret local database of the corresponding pairs (*group-public-key, user, blind-message*) to link the users who submitted the messages. Finally, the users remove the blind factors and obtain their resulting group/ring signatures. Notice that the verification of these signatures requires some dynamic information (It is often a part of the group public key) related to the change of the group. Hence, by checking whether or not this information is in its local database of pairs (*group-public-key, user, blind-message*), the malicious signer can now trace its signed message and the user who submitted the blind message.

In the schemes in [16] and [19], the public key of the underlying group signature keeps unchanged when a member joins the group while the public key will change when some members leave. Hence, the implementation of their schemes will be insecure if the members frequently leave. The case is more serious for the schemes in [6] designed for blind spontaneous anonymous group signatures for ad hoc groups since the final verification key changes if any member joins or leaves the group. Indeed, an anonymous signer can freely determine the verification key so that it can link it with the signed message and the user more easily. However, the above attack and risks from malicious signers have not been pointed out in these schemes, which may cause misuse of the schemes and serious security concerns in practice.

1.5 Our Contributions

Consider the following scenario. Many banks compete for their services in one country. Some of the banks tend to ally to improve their competition capacity, and they will share their responsibility and interests from the ally. These allied

banks now want to anonymously issue e-cash so that the competitors out of the ally cannot trace them. Meanwhile, the privacy of spender who has e-cash from the allied banks should be protected. What can they do in this case?

Fortunately, the above attack does not apply to static groups. To address the problems such as those of the allied banks, an efficient static blind ring signature is proposed, in which the message to be signed is blind and the signer is anonymous. Furthermore, it is infeasible to determine whether two blind ring signatures are generated by a same signer. The security of the proposed schemes is provable under the extended ROS assumptions in the random oracle model and the generic group model. After the group public key is generated, the space, time, and communication complexities of the relevant parameters and operations are constant, while the complexity of the schemes in [6] linearly increases regarding the group size. Our scheme appears to be the first blind ring signature enjoying constant complexity.

The rest of the paper is organized as follows. In Section 2, we provide some preliminaries of our proposals. Section 3 defines the security model of static blind ring signatures. We propose our static blind ring signature and detail its performance in Section 4, followed by concluding remarks in the last section.

2 Preliminaries

2.1 Generic Group Model

Generic algorithms for group G do not use the binary encodings of the group elements, as they access group elements only for group operations and equality tests. The *data of a generic algorithm* is partitioned into group elements in G and non-group data. The *generic steps* for group elements are multivariate exponentiations. The *restriction of the generic model* is that A can use group elements only for generic group operations, equality tests and for queries to the hash oracle, whereas non-group data can be arbitrarily used without charge. The computed group elements are given as explicit multiplicative combinations of given group elements. Nechaev [18] proved that the discrete logarithm problem is hard in such a model. The generic model of algorithms was further elaborated on by Shoup [23].

2.2 Computational Assumptions

In this section we review the strong RSA assumption [4] and ROS assumption [22].

A number N is an RSA integer if $N{=}PQ$ where P and Q are safe primes: $P{=}2P'{+}1$, $Q{=}2Q'{+}1$, where both P' and Q' are also primes. Let RSA_λ be the set of RSA integers of size λ.

Strong RSA Assumption. Given λ, for any polynomial time adversary **Ad:**

$$\Pr[N \leftarrow \mathrm{RSA}_\lambda;\, y \leftarrow \mathbb{Z}_N^*;\, (u,\, x) \leftarrow \mathbf{Ad}(1^\lambda,\, N,\, y) | x \neq \pm 1 \wedge u^x = y] < \varepsilon(\lambda)$$

the probability being over the random choice of N and y, and **Ad**'s random coin flips, where $\varepsilon(\lambda)$ is a function such that for all polynomials $p(\lambda)$, $1/\varepsilon(\lambda) < 1/p(\lambda)$ holds for all sufficient large λ.

ROS-problem. Given a prime q and an oracle random function F: $\mathbb{Z}_q^l \to \mathbb{Z}_q$, find coefficients $a_{i,j} \in \mathbb{Z}_q$ for $i = 1, \cdots, t, j = 1, ..., l$, and a solvable system of $l+1$ distinct equations in the unknowns $c_1, \cdots, c_l$ over $\mathbb{Z}_q$:

$$a_{i,1}c_1 + ... + a_{i,l}c_l = F(a_{i,1}, ..., a_{i,l}) \bmod q \ .$$

The ROS-assumption in the general group model means that any PPT adversary can solve ROS-problem with only negligible probability if the adversary can access only group operations as specified in Section 2.1.

Extended ROS-problem. Given an oracle random function $F : \times_{j=1}^{l}\{0,1\}^* \to \{0,1\}^{\lambda_1}$, find coefficients $a_{i,j} \in \{0,1\}^*$ for $i = 1, \cdots, t, j = 1, ..., l$, and a solvable system of $l+1$ distinct equations in the unknowns $c_1, \cdots, c_l$ over $\{0,1\}^{\lambda_1}$:

$$a_{i,1}c_1 + \ldots + a_{i,l}c_l = F(a_{i,1}, \ldots, a_{i,l}) \bmod \tilde{q},$$

where $\tilde{q}$ is unknown.

The Extended ROS-assumption in the general group model means that any PPT adversary can solve Extended ROS-problem with only negligible probability if the adversary can access only group operations as specified in Section 2.1.

Schnorr evaluated the expected time $O(\sqrt{q})$ to solve ROS problem and assumed that ROS problem is infeasible in the generic group model [22] when $q > 2^{160}$. However, the further analysis in [25] shew that it only requires subexponential running time $O(2^{2\sqrt{\log q}})$ to break ROS problem with the generalized birthday attack. Hence, to achieve a popular level of security $O(2^{80})$, one should set $q > 2^{1600}$. So the existing generalized birthday attack on Schnorr's ROS problem makes the Schnorr and Okamoto-Schnorr blind signatures over elliptic curve groups disadvantaged.

Clearly, for any solution of the Extended ROS-problem, it is also a solution to the corresponding ROS-problem. Hence, the Extended ROS-problem is at least as difficult as the the original ROS-problem. For the the generalized birthday attack, the Extended ROS-problem makes it more difficult as the modular reduction is not available. Hence, we assume that the Extended ROS-assumption holds if appropriately setting the security parameters.

2.3 Proof of Range of Discrete Logarithm

In this section, we recall the zero-knowledge proof of discrete logarithm lying in a specific interval [7]. For a general proof of discrete logarithm relation in a specific interval, see [15].

Let $\lambda_1, \lambda_2, \lambda_3$ be security parameters and a hash function output λ_1-bit strings. The protocol proves that a committed number $x \in I$ belongs to J, where the expansion rate $\sharp J/\sharp I$ is equal to $2^{\lambda_1+\lambda_2+1}$. Let n be a large composite number whose factorization is unknown by Alice and Bob, g an element of large order

in $\mathbb{Z}_n$ and h be an element of the group generated by g such that the discrete logarithm of h in base g is unknown by Alice. $y = g^x h^r \bmod n$ is a commitment to $x \in [0, b]$, where r is randomly selected over $[-2^{\lambda_3} n + 1, 2^{\lambda_3} n - 1]$. Denote the following protocol by $ZK\{x, r | y = g^x h^r \bmod n \wedge x \in [-2^{\lambda_1+\lambda_2} b, 2^{\lambda_1+\lambda_2} b]\}$.

- Alice picks at random $\omega \in (0, 2^{\lambda_1+\lambda_2} b - 1)$ and $\eta \in (-2^{\lambda_1+\lambda_2+\lambda_3} n + 1, 2^{\lambda_1+\lambda_2+\lambda_3} n - 1)$, and then computes $W = g^\omega h^\eta \bmod n$, $c = H(W)$. Finally, she computes $D_1 = \omega + cx$ and $D_2 = \eta + cr$ (in $\mathbb{Z}$). If $D_1 \in [cb, 2^{\lambda_1+\lambda_2} b - 1]$, she sends ($c$, D_1, D_2) to Bob, otherwise she starts again the protocol.
- Bob checks that $D_1 \in [cb, 2^{\lambda_1+\lambda_2} b - 1]$ and $c = H(g^{D_1} h^{D_2} y^{-c})$. This convinces Bob that $x \in [-2^{\lambda_1+\lambda_2} b, 2^{\lambda_1+\lambda_2} b]$.

The proof succeeds with probability greater than $1 - 2^{-\lambda_2}$ if $x \in [0, b]$. The verifier is convinced that $x \in [-2^{\lambda_1+\lambda_2} b, 2^{\lambda_1+\lambda_2} b]$. A cheating prover can succeed with probability less than $2^{-\lambda_1+1}$.

3 Definitions of Security

3.1 Static Blind Ring Signatures

In this paper, we only consider the static blind ring signatures, that is, although the willing members can form a group freely, it will keep unchanged for a long term once the ad hoc group forms, for instance, the application of some banks to ally.

A static blind ring signature is a tuple $BRS = (\boldsymbol{MKG}^{BRS}(\mathbf{1}^\lambda), \boldsymbol{GKG}^{BRS}(\cdot), \boldsymbol{S}^{BRS}(\cdot), \boldsymbol{V}^{BRS}(\cdot, \cdot, \cdot))$.

- $(sk_i; pk_i) \leftarrow \boldsymbol{MKG}^{BRS}(\mathbf{1}^\lambda)$ is a PPT algorithm which, on input a security parameter λ, outputs the ring members' private/public key pairs $(sk_i; pk_i)$. We denote by PK the list of the public keys forming the static group and SK the corresponding private keys.
- $pk \leftarrow \boldsymbol{GKG}^{BRS}(PK)$ is a PPT algorithm which, on input all the possible public keys PK of the group members, outputs pk as the group public key.
- $\sigma \leftarrow \boldsymbol{S}^{BRS}_{sk_i;m}(pk)$ is a two-party protocol between an anonymous ring $Signer$ and a $User$. They have common input pk, which is the group public key. The anonymous ring $Signer$ has private input sk_i, which is its secret key, and its inner random coin flips. The $User$ has private input m, which is the message to be signed, and its inner random coin flips. At the end of the interaction between the two parties, the $Signer$ outputs one of the two messages: $completed$, $not - completed$, and the $User$ outputs either $fail$ or a signature σ on m.
- $0/1 \leftarrow \boldsymbol{V}^{BRS}(m, \sigma, pk)$ is a polynomial-time deterministic algorithm which, on input a message-signature pair (m, σ) and the group public key pk returns 1 or 0 for *accept* or *reject*, respectively. If *accept*, the message-signature pair is valid.

The security of static blind ring signature schemes has four critical aspects: completeness, anonymity, blindness, $(l, l+1)$-Unforgeability as well as an optional aspect: linkability. The are formally defined as follows.

Definition 1. (Completeness.) *If the ring Signer and the User honestly follow the protocol* $\boldsymbol{S}^{BRS}(\cdot)$*, the User will output a signature* σ *on* m *accepted by* $\boldsymbol{V}^{BRS}(\cdot,\cdot,\cdot)$ *with a dominant probability.*

Definition 2. (Signer Anonymity.) *Let* $PK = \{pk_1, \cdots, pk_I\}$*, where each key is generated as* $(sk_i, pk_i) \leftarrow MKG^{BRS}(1^\lambda)$*. A BRS scheme is anonymous if, for any message* m*, and the transcript* $\bowtie$ *of interaction due to* $\sigma \leftarrow \mathbf{S}^{BRS}_{sk_i;m}(pk)$*, where* $i \leftarrow \{1, \ldots, I\}$*, given* $(PK, m, \bowtie)$*, any PPT adversary* $\boldsymbol{Ad}$ *outputs* i' *such that* $i = i'$ *with probability* $Adv^{Anon}_{Ad}(\lambda)$*, where* $|Adv^{Anon}_{Ad}(\lambda) - 1/I|$ *is negligible in* λ*.*

Definition 3. (Linkability/Uninkability.) *A BRS scheme is linkable if there exists a PPT algorithm* $Link(\cdot,\cdot)$*, for any two transcripts* $\bowtie_1, \bowtie_2$*, which outputs 1 or 0 with non-negligible probability, representing that the same ring Signer produces* $\bowtie_1$ *and* $\bowtie_2$ *or not, no matter whether or not the User is the same one. Else if there exists no such PPT algorithm, the BRS scheme is unlinkable.*

Definition 4. (Blindness.) *The blindness is defined via an experiment involving an adversarial ring signer* $\boldsymbol{Ad}$*. The experiment is parameterized by a bit* b *and security parameter* λ*. First* $MKG^{BRS}(1^\lambda)$ *and* $GKG^{BRS}(PK)$ *are correctly run to generate the adversarial ring signer* $\boldsymbol{Ad}$*'s secret key/public key pair* (sk_i, pk_i) *and the group public key* pk*. Then,* $\boldsymbol{Ad}$ *outputs a pair of messages* (m_0, m_1) *lexicographically ordered. In the next stage of the experiment* $\boldsymbol{Ad}$ *engages in two (possibly correlated and interleaved) runs with two honest users, with inputs* m_b *and* $m_{b'}$*, respectively. If both users obtain valid signatures, on their respective message,* $\boldsymbol{Ad}$ *is also given these two signatures; otherwise there is no extra input to* $\boldsymbol{Ad}$*; in either case,* $\boldsymbol{Ad}$ *is required to output a bit* b''*. The advantage of* $\boldsymbol{Ad}$ *is defined by:*

$$Adv^{Bld}_{BRS,Ad}(\lambda) = 2\Pr[b = b''] - 1$$

A BRS scheme satisfies the blindness property, if for any PPT $\boldsymbol{Ad}$*, the function* $Adv^{Bld}_{BRS,Ad}(\lambda)$ *is negligible in* λ*.*

Definition 5. ($(l,l+1)$-Unforgeability.) *The* $(l,l+1)$*-Unforgeability of a BRS scheme is defined via an experiment parameterized by security parameters* l *and* λ*. The experiment involves an adversarial user* $\boldsymbol{Ad}$ *and is as follows. First* $MKG^{BRS}(1^\lambda)$ *and* $GKG^{BRS}(1^\lambda)$ *are run to generate the adversarial ring signer's secret/public key pair* (sk_i, pk_i) *and the group public key* pk*. Then,* $\boldsymbol{Ad}$ *engages in polynomially many (parallel) runs of the protocol with a ring Signer. Finally,* $\boldsymbol{Ad}$ *outputs a list of message-signature pairs* $\{(m_1, \sigma_1), (m_2, \sigma_2), \ldots, (m_t, \sigma_t)\}$ *with* $m_i \neq m_j$*. Let* l *be the number of runs successfully completed by the signer. Define the advantage of* $\boldsymbol{Ad}$

$$Adv^{Unf}_{BRS,Ad}(\lambda) = \Pr[\forall 1 \leq i \leq t, Ver^{BRS}(m_i, \sigma_i, pk) = 1 \wedge (l < t)]$$

and say that the BRS scheme is $(l, l+1)$*-unforgeable if* $Adv^{Unf}_{BRS,Ad}(\lambda)$ *is negligible for any PPT adversary user* ***Ad****.*

4 Proposed Blind Ring Signature

4.1 Basic Ideas

We provide here our basic ideas to make the following scheme more readable. The underlying is a knowledge proof that the signer knows x in a *certain range* and $v \in \mathbb{Z}_N^*$ such that $y = v^x \bmod N$, where N is an RSA modulus with unknown factorization and y is the public key. To achieve a ring signature, each ring member contributes to the public key $y = g^{n_1 n_2 \cdots n_I} \bmod N$, where g is an agreed quadratic residue modulo N and member i knows only the factorization of its RSA public key n_i. Finally, the blindness is achieved by exploiting the classical Schnorr-transformation with additional attention to blind the component v. The rest is to carefully set the security parameters so that the required security properties can be achieved.

4.2 The Scheme

Let λ, λ_1, λ_2, λ_3, λ_4, λ_5, λ_6, λ_7, λ_8 be security parameters. They are all polynomials of λ. $H(\cdot)$ is a hash function which outputs λ_1-bit strings. In the following, we detail the blind unlinkable ring signature for static groups.

[Member key generation]: Primes p_i, q_i satisfy $2^{\lambda_4} < p_i < 2^{\lambda_4} + 2^{\lambda_5}, 2^{\lambda_6} < q_i$. Set $PK = \{n_i\}_{i=1}^{I}$, where $n_i = p_i q_i$ is signer i's $(i = 1, 2, \cdots, I)$ public key and p_i, q_i its private key. The security parameters satisfy that $2^{\lambda_1+\lambda_2+\lambda_7+\lambda_8} < 2^{\lambda_4} - 2^{\lambda_1+\lambda_2+\lambda_5}$, $2^{\lambda_4} + 2^{\lambda_1+\lambda_2+\lambda_5} < 2^{\lambda_6}$, $\lambda_4 \leq 2\lambda_5$. Notice that $2^{\lambda_4} < p_i < 2^{\lambda_4} + 2^{\lambda_5}$ implies that $\lambda_4 - \lambda_5 + 1 (> 0)$ bits of p_i are leaked. Since efficient factorization techniques are known when at least $\log n_i/3$ bits of the prime factors of n are known [14], we need set the security parameters such that the leaked bits of p_i are less than $\lambda_4/2 < \lg n_i/4$. As most ring signatures, we assume that all members' pulbic/private key pairs are PKI-based and correctly generated as the specification.

[Group key generation]: Let N be randomly from RSA_λ with its factorization unknown, and a generator $g \in (\mathbb{Z}_N^*)^2$ with unknown order. Anyone can compute $y = g^{n_1 n_2 \cdots n_I} \bmod N$. Hereafter in this paper, the modular exponentiations are computed with modulo N. We omit it without confusion. The group public key is $\{\lambda, H(\cdot), N, g, y\}$.

[Signing procedure]: A ring member computes $u = g^{q_i \prod_{j \neq i}^{I} n_j}$ in advance (In the implementation, the ring member can compute and secretly save u before y so that less computation is required). It produces a blind ring signature σ on a blind message M with a user as follows.

- The signer randomly selects $\delta \leftarrow (0, 2^{\lambda_7})$, and computes $z = y^{\delta}$, $v = u^{\delta}$. The signer randomly selects $\beta \leftarrow (0, 2^{\lambda_1+\lambda_2+\lambda_7})$, $\omega \leftarrow (2^{\lambda_1+\lambda_2+\lambda_5-1} + 2^{\lambda_1+\lambda_5}, 2^{\lambda_1+\lambda_2+\lambda_5})$, computes $a = y^{\beta}$, $b = v^{\omega}$, and anonymously sends (z,v,a,b) to the user.
- The user randomly selects $\xi \leftarrow \pm(0, 2^{\lambda_1+\lambda_2+\lambda_7+\lambda_8})$, $\eta \leftarrow \pm(0, 2^{\lambda_1+\lambda_2+\lambda_5})$, $\rho \leftarrow \pm(0, 2^{\lambda_1})$, $\phi \leftarrow (2^{\lambda_8-1}, 2^{\lambda_8})$, $\gamma \leftarrow (0, 2^{\lambda_1})$, where $2^{\lambda_1+\lambda_2+\lambda_7+\lambda_8} < 2^{\lambda_4} - 2^{\lambda_1+\lambda_2+\lambda_5}$ and $2^{\lambda_4} + 2^{\lambda_1+\lambda_2+\lambda_5} < 2^{\lambda_6}$, and computes $m = H(M||\gamma)$, $Z = z^{\phi}$, $V = v^{\phi}$, $c = H(m||a^{\phi}y^{-\xi}Z^{\rho}||\ b^{\phi}Z^{\rho}V^{-\eta-\rho 2^{\lambda_4}})$, and checks whether or not $0 < c + \rho < 2^{\lambda_1}$. If not, it repeats this step. It broadcasts $c' = c + \rho$ to the group.
- The signer computes $r' = \beta + c'\delta$, $s' = \omega + c'(p_i - 2^{\lambda_4})$. It checks whether or not $2^{\lambda_1+\lambda_2+\lambda_7-1} + 2^{\lambda_1+\lambda_7} < r' < 2^{\lambda_1+\lambda_2+\lambda_7}$ and $2^{\lambda_1+\lambda_2+\lambda_5-1} + 2^{\lambda_1+\lambda_5} < s' < 2^{\lambda_1+\lambda_2+\lambda_5}$. If not, it and the user repeat the above steps. It anonymously sends r', s' to the user.
- The user checks whether or not $2^{\lambda_1+\lambda_2+\lambda_7-1} + 2^{\lambda_1+\lambda_7+1} < r' < 2^{\lambda_1+\lambda_2+\lambda_7}$, $2^{\lambda_1+\lambda_2+\lambda_5-1} + 2^{\lambda_1+\lambda_5} < s' < 2^{\lambda_1+\lambda_2+\lambda_5}$, $a = y^{r'}z^{-c'}$, $b = v^{s'+c'2^{\lambda_4}}z^{-c'}$. If not, the user outputs *failure* and aborts the protocol. Else it computes $r = \phi r' - \xi$, $s = s' - \eta$, and outputs (Z, V, c, r, s). It checks whether $c2^{\lambda_7+\lambda_8} < r < 2^{\lambda_1+\lambda_2+\lambda_7+\lambda_8}$, $c2^{\lambda_5} < s < 2^{\lambda_1+\lambda_2+\lambda_5}$. If not, it invalidates and broadcasts (m, Z, V, c, r, s). The anonymous signer repeats the above protocol with the user if $c = H(m||y^r Z^{-c}||V^{s+c2^{\lambda_4}}Z^{-c})$ and r, s are in incorrect ranges. Finally, the user outputs (γ, Z, V, c, r, s) as the resulting ring signature on blind message M.

[Verification procedure]: Compute $m = H(M||\gamma)$. Check that

$$c2^{\lambda_7+\lambda_8} < r < 2^{\lambda_1+\lambda_2+\lambda_7+\lambda_8},\ c2^{\lambda_5} < s < 2^{\lambda_1+\lambda_2+\lambda_5},$$
$$c = H(m||y^r Z^{-c}||V^{s+c2^{\lambda_4}}Z^{-c}).$$

Output *accept* if and only if all the checks hold.

4.3 Security Analysis of the Scheme

In this section, we analyze the security of the scheme following the definitions. Notice that $m = H(M||\gamma)$ is used to (unconditionally) mask the real message M in the last step of signing procedure if the output r or s of the user is not in the right range. In this case, the user can publish (m, Z, V, c, r, s) to show that this is really a part of the invalid resulting signatures due to incorrect ranges of r or/and s. The anonymous signer can validate the honesty of the user by checking $c = H(m||y^r Z^{-c}||V^{s+c2^{\lambda_4}}Z^{-c})$ but r and/or s are in incorrect ranges. The public (m, Z, V, c, r, s) will not leak any information of the involved blind message M in case of unsuccessful interactions, where γ is unknown and $m = H(M||\gamma)$. Hence, a successful interaction is the interaction in which a valid resulting ring signature (γ, Z, V, c, r, s) on blind message M is output. As M is masked by $m = H(M||\gamma)$ and the invalid pairs (m, Z, V, c, r, s) will be published, a successful interaction is also the interaction in which a valid pair

(m, Z, V, c, r, s) is output. So in the following analysis, we will view (m, Z, V, c, r, s) as the resulting blind ring message-signature pairs without confusion in case of successful interactions.

Theorem 1. (***Completeness.***) *If the signer and user follow the protocol, the output will be accepted by the verification procedure.*

Proof. From the protocol, we have that

$$a^{\phi}y^{-\xi}Z^{\rho} = (y^{r'}z^{-c'})^{\phi}y^{-\xi}Z^{\rho} = y^{\phi r'-\xi}z^{-c'\phi}Z^{\rho}$$
$$= y^{r}Z^{\rho-c'} = y^{r}Z^{-c},$$
$$b^{\phi}Z^{\rho}V^{-\eta-\rho 2^{\lambda_4}} = (v^{s'+c'2^{\lambda_4}}z^{-c'})^{\phi}V^{-\eta-\rho 2^{\lambda_4}}Z^{\rho}$$
$$= (V^{s+\eta+c'2^{\lambda_4}}z^{-c'\phi})V^{-\eta-\rho 2^{\lambda_4}}(Z^{-c'}Z^{\rho})$$
$$= V^{s+(c'-\rho)2^{\lambda_4}}Z^{\rho-c'} = V^{s+c2^{\lambda_4}}Z^{-c}.$$

It follows that

$$c = H(m||a^{\phi}y^{-\xi}Z^{\rho}\ ||b^{\phi}Z^{\rho}V^{-\eta-\rho 2^{\lambda_4}})$$
$$= H(m||y^{r}Z^{-c}||\ V^{s+c2\lambda_4}Z^{-c}).$$

Theorem 2. (***Unforgeability.***)*In the random oracle model plus the generic group model, the above BRS scheme is (l,l+1)-unforgeable if the extended ROS problem is infeasible.*

Proof. Appendix A.

Theorem 3. *The above BRS scheme is anonymous, unlinkable and blind.*

Proof. For the anonymity, it is sufficient to consider that the involved user as the adversary. Note that the transcript from the ring signer (z, v, a, b, r', s') is a knowledge proof $ZK\{u_i, p_i | y = u_i^{p_i} \wedge -2^{\lambda_1+\lambda_2+\lambda_5} < p_i - 2^{\lambda_4} < 2^{\lambda_1+\lambda_2+\lambda_5}\}$, where $p_i | n_1 n_2 \cdots n_I$ and $u_i = g^{q_i \prod_{j\neq i}^{I} n_j}$. Under the Strong RSA assumption and DLP assumption over $(\mathbb{Z}_N^*)^2$, the witnesses (u_i, p_i) and (u_k, p_k) are indistinguishable, where $p_k | n_1 n_2 \cdots n_I$ and $u_k = g^{q_k \prod_{j\neq k}^{I} n_j}$. Hence, the transcripts (z, v, a, b, r', s') from different ring members are indistinguishable if the Strong RSA assumption and the DLP problem over $(\mathbb{Z}_N^*)^2$ are infeasible and the anonymity follows. Furthermore, due the witness indistinguishability, an adversary user cannot determine whether or not the witnesses (u_i, p_i) and (u_k, p_k) involved different interactions are the same. Therefore, the blind ring signature is unlinkable.

For the blindness, for any view $(\delta, z, v; \beta, a; \omega, b; c', r', s')$ and any valid message-signature pair (m, Z, V, c, r, s), a tuple $(\tilde{\rho}, \tilde{\phi}, \tilde{\xi}, \tilde{\eta})$ in the corresponding ranges is a valid blind factor if and only if $c = H(m||y^{r}Z^{-c}||V^{s+c2\lambda_4}Z^{-c}) = H(m||a^{\tilde{\phi}}y^{-\tilde{\xi}}Z^{\tilde{\rho}}$ $||b^{\phi}Z^{\tilde{\rho}}V^{-\tilde{\eta}-\tilde{\rho}2^{\lambda_4}})$, $\tilde{\phi} \in (2^{\lambda_8-1}, 2^{\lambda_8})$, $Z = z^{\tilde{\phi}}$ and $V = v^{\tilde{\phi}}$.

Let $\tilde{\rho} = c' - c$. Randomly select $\tilde{\phi} \leftarrow (2^{\lambda_8-1}, 2^{\lambda_8})$. Set $\tilde{\xi} = \tilde{\phi}r' - r, \tilde{\eta} = s' - s$. It follows that

$$c = H(m||y^{r}Z^{-c}||\ V^{s+c2\lambda_4}Z^{-c})$$
$$= H(m||y^{r}Z^{\tilde{\rho}-c'}||\ V^{s+(c'-\tilde{\rho})2^{\lambda_4}}Z^{\tilde{\rho}-c'})$$
$$= H(m||y^{\tilde{\phi}r'-\tilde{\xi}}z^{-c'\tilde{\phi}}Z^{\tilde{\rho}}||\ (V^{s+\tilde{\eta}+c'2^{\lambda_4}}z^{-c'\tilde{\phi}})V^{-\tilde{\eta}-\tilde{\rho}2^{\lambda_4}}(Z^{-c'}Z^{\tilde{\rho}}))$$
$$= H(m||y^{\tilde{\phi}r'-\tilde{\xi}}z^{-c'\tilde{\phi}}Z^{\tilde{\rho}}||\ (V^{s+\tilde{\eta}+c'2^{\lambda_4}}z^{-c'\tilde{\phi}})V^{-\tilde{\eta}-\tilde{\rho}2^{\lambda_4}}(Z^{-c'}Z^{\tilde{\rho}}))$$

$$= H(m||(y^{r'}z^{-c'})^{\tilde{\phi}}y^{-\xi}Z^{\tilde{\rho}}||\ (v^{s'+c'2^{\lambda_4}}z^{-c'})^{\tilde{\phi}}V^{-\tilde{\eta}-\tilde{\rho}2^{\lambda_4}}Z^{\tilde{\rho}})$$
$$= H(m||a^{\tilde{\phi}}y^{-\tilde{\xi}}Z^{\tilde{\rho}}\ ||b^{\phi}Z^{\tilde{\rho}}V^{-\tilde{\eta}-\tilde{\rho}2^{\lambda_4}})$$

For the range checks, they will clearly hold. Hence, these equations show that $(\tilde{\rho}, \tilde{\phi}, \tilde{\xi}, \tilde{\eta})$ is a valid blind factor tuple if and only if there exists $\tilde{\phi} \in (2^{\lambda_8-1}, 2^{\lambda_8})$ such that $Z = z^{\tilde{\phi}}$, $V = v^{\tilde{\phi}}$. If λ_8 is sufficiently large, it is difficult to determine whether there exists such a solution under the DLP assumption over $(\mathbb{Z}_N^*)^2$, where the factorization of N is unknown [15]. Hence, the BRS scheme is blind.

4.4 Performance Analysis and Comparison

In the above blind unlinkable ring signature, the signer, the user and the verifier respectively requires 4, 10 and 4 modular exponentiations. Here and hereafter in this paper, the modular exponentiation with t bases are calculated as t modular exponentiations for a coarse efficiency analysis. But one should notice that the speeding-up algorithm to the computation of exponentiations with multiple bases is widely known. It requires $2\lambda+4\lambda_1+2\lambda_2+\lambda_5+\lambda_7+\lambda_8$ bits. Set $\lambda = 1600, \lambda_1 = 160, \lambda_2 = 80, \lambda_5 = 600, \lambda_7 = 160, \lambda_8 = 160$. The resulting signature needs 0.597 KB, independent of the group size. In the state-of-the-art blind ring signature schemes [6], the signer, the user and the verifier respectively requires I, $2I$ and $2I$ modular exponentiations, where I is the number of the ring members. Its resulting blind ring signature requires at least $0.4I$KB. Hence, our scheme is more efficient.

5 Conclusion

We commented that it is impossible to construct blind group/ring signatures for dynamic groups if the the group public key changes when a member joins or leaves the group. The blindness of such schemes will be easily broken by a malicious anonymous signer of the dynamic groups. We also pointed out that it is possible to integrate the blindness of message into group/ring signatures for static groups and give some potential applications of such schemes. Then an efficient static blind ring signature was proposed with their security provable under the extended ROS assumption in the random oracle model and the generic group model. After the group public key is generated, the space, time, and communication complexities of the relevant parameters and operations keep independent of the group size. It is also the first scheme satisfying such properties, and hence very practical.

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. Proc. Crypto'00, LNCS 1880, pp. 255-270, Springer-Verlag, 2000.
2. M. Abe, M. Ohkubo, and K. Suzuki. 1-out -of-n signatures from a variety of keys. Proc. Asiacrypt'02, LNCS 2501, pp. 415-432, Springer-verlag, 2002. .

3. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. Proc. Crypto'04, LNCS 3152, pp. 41-55, Springer-Verlag, 2004.
4. N. Baric, B. Pfitzman. Collision-free accumulators and fail-stop signature schemes without trees. Proc. Eurocrypt'97, LNCS 1233, pp. 480-494, Springer-Verlag, 1997.
5. E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. Proc. Crypto'02, LNCS 2442, pp. 465-480, Springer-Verlag, 2002.
6. T. K. Chan, K. Fung, J. K. Liu, and V. K. Wei. Blind Spontaneous Anonymous Group Signatures for Ad Hoc Groups. Proc. ESAS 2004, LNCS 3313, pp. 82-94, Springer-Verlag, 2005.
7. A. Chan, Y. Frankel, and Y. Tsiounis. Easy Come - Easy Go Divisible Cash. Updated version with corrections, GTE Tech. Rep. (1998), available at http://www.ccs.neu.edu/ home/ yiannis/.
8. D. Chaum, E. van Heyst. Group signatures. Proc. Eurocrypt'91, LNCS 547, pp. 257-265, Springer-Verlag, 1991.
9. D. Chaum. Blind signatures for untraceable payments. Proc. Crypto'82, pp. 199-203, New York, Plenum Press, 1983.
10. J. Camenisch, M. Koprowski, and B. Warinschi. Efficient Blind Signatures Without Random Oracles. Proc. SCN'04, LNCS 3352, pp. 134-148,Springer-Verlag, 2005.
11. J. Camenisch, A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. Proc. Crypto'02, LNCS 2442, pp. 61-76, Springer-Verlag, 2002.
12. J. Camenisch, M. Stadler. Efficient group signatures for large groups, Proc. Crypto'97, LNCS 1294, pp. 465-479, Springer-Verlag, 1997.
13. Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous Identification in Ad Hoc Groups. Proc. Eurocrypt'04, LNCS 3027, pp. 609-626. Springer-Verlag, 2004.
14. J. Hastad, A. W. Schrift, and A. Shamir. The discrete logarithm modulo a composite hides O(n) bits. JCSS: Journal of Computer and System Sciences, 47(3): 376-404, 1993.
15. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. Proc. Eurocrypt'04, LNCS 3027, pp. 571-589, Springer-Verlag, 2004. http://eprint.iacr.org/2004/007.pdf
16. A. Lysyanskaya, Z. Ramzan. Group Blind Digital Signatures: A Scalable Solution to Electronic Cash. Proc. FC'98, LNCS 1465, pp. 184-197, Springer-Verlag, 1998.
17. J. K. Liu, V. K. Wei, and D. S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). Proc. ACISP'04, LNCS 3108, pp. 325-335. Springer-Verlag, 2004.
18. V.I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. Mathematical Notes 55, pp. 165-172, 1994.
19. K. Q. Nguyen, Y. Mu, and V. Varadharajan. Divertible Zero-Knowledge Proof of Polynomial Relations and Blind Group Signature. Proc. ACISP'99, LNCS 1587, pp. 117-128, 1999. Springer-Verlag, 1999.
20. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. Journal of Cryptology, 13(3):361-396, 2000.
21. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In Proc. Asiacrypt'01, LNCS 2248, pp. 552-565, Springer-Verlag, 2001.
22. C. P. Schnorr. Security of blind discrete log signatures against interactive attacks. Proc. ICICS'01, LNCS 2229, pp. 1-12. Springer-Verlag, 2001.
23. V. Shoup. Lower bounds for discrete logarithms and related problems. Proc. Eurocrypt'97, LNCS 1233, Springer-Verlag, pp. 256-266, 1997.
24. G. Tsudik, S. Xu. Accumulating composites and improved group signing. Proc. Asiacrypt'03, LNCS 2894, pp. 269-86. Springer-Verlag, 2003.

25. D. Wagner. A Generalized Birthday Problem (Extended Abstract). Proc. Crypto'02, LNCS 2442, pp. 288-304, 2002. Springer-Verlag, 2002.
26. Q. Wu, X. Chen, C. Wang, Y. Wang. Shared-Key Signature and Its Application to Anonymous Authentication in Ad Hoc Group. Proc. ISC'04, LNCS 3225, pp. 330-341. Springer-Verlag, 2004.
27. F. Zhang and K. Kim. ID-Based blind signature and ring signature from pairings. Proc. Asiacrypt'02, LNCS 2501, pp. 533-547, Springer-Verlag, 2002.

Appendix A: Proof of Theorem 2

Sketch of Proof. Since a solution of the extended ROS-problem is not related to the public key and thus it cannot be used to solve the signer's secrete key. So the signing oracle is allowed to know the secret key and it can work as a true ring signer. The random oracle is not controlled by the signer in a blind ring signature scheme. After an $(l, l+1)$-forger **Ad** submits a query Q_j to the random oracle, it will respond with a random λ_1-bit string $c_j = H(Q_j)$. In the general group model, **Ad** is allowed to execute t general steps, including l interactions with the signing oracle and generating t_s group elements and t_h interactions with the random oracles, where $0 < t_s + t_h < t$.

Now we describe the game between the signing oracle (signer for short) and the forger **Ad**.

First, the signer and **Ad** are given the public parameters correctly generated. The signer is additionally allowed to get a secret p, which is a factor of n_i, so that the signer can complete the signing procedure.

Then the signer computes u as the protocol, and randomly selects $\delta_\ell \leftarrow (0, 2^{\lambda_7})$ for $\ell = 1, \cdots, l$, and computes $z_\ell = y^{\delta_\ell}$, $v_\ell = u^{\delta_\ell}$. The signer randomly selects $\beta_\ell \leftarrow (0, 2^{\lambda_1+\lambda_2+\lambda_7})$, $\omega_\ell \leftarrow (2^{\lambda_1+\lambda_2+\lambda_5-1} + 2^{\lambda_1+\lambda_5}, 2^{\lambda_1+\lambda_2+\lambda_5})$, computes $a_\ell = y^\beta$, $b_\ell = v^\omega$, and anonymously sends $(z_\ell, v_\ell, a_\ell, b_\ell)$ to **Ad**.

Then **Ad** selects integers $d_{i,-1}, d_{i,0}$, and $d_{i,1,1}, d_{i,1,2}, d_{i,1,3}, d_{i,1,4}, \cdots, d_{i,l,1}$, $d_{i,l,2}, d_{i,l,3}, d_{i,l,4}$, and $e_{i,1,1}, e_{i,1,2}, e_{i,1,3}, e_{i,1,4}, \cdots, e_{i,l,1}, e_{i,l,2}, e_{i,l,3}, e_{i,l,4}$, and messages m_i for $i = 1, \cdots, t_h$. **Ad** computes

$\phi_i = g^{d_{i,-1}} y^{d_{i,0}} z_1^{d_{i,1,1}} v_1^{d_{i,1,2}} a_1^{d_{i,1,3}} b_1^{d_{i,1,4}} \cdots z_l^{d_{i,l,1}} v_l^{d_{i,l,2}} a_l^{d_{i,l,3}} b_l^{d_{i,l,4}}$

and

$$\psi_i = g^{e_{i,-1}} y^{e_{i,0}} z_1^{e_{i,1,1}} v_1^{e_{i,1,2}} a_1^{e_{i,1,3}} b_1^{e_{i,1,4}} \cdots z_l^{e_{i,l,1}} v_l^{e_{i,l,2}} e_l^{e_{i,l,3}} b_l^{e_{i,l,4}}.$$

Notice that group elements ϕ_i, ψ_i can always be written in this form. **Ad** asks the random oracle with its query $Q_i = (m_i||\phi_i||\psi_i)$ and the random oracle responds with random values $H_i = H(m_i||\phi_i||\psi_i)$. Now **Ad** computes its challenges $c_1, \cdots, c_l$ in arbitrary way from H_i and broadcasts them to the signer.

Then, the signer honestly and anonymously responds with $r'_\ell = \beta_\ell + c'_\ell \delta_\ell$, $s'_\ell = \omega_\ell + c'_\ell x$, where $x = p - 2^{\lambda_4}$. r'_ℓ and s'_ℓ will be in the right range.

Finally, **Ad** is required to forge $l+1$ valid message signature pairs $(m_k, Z_k, V_k, c_k, r_k, s_k)$ in the corresponding right ranges for $k = 1, \cdots, l+1$.

From the verification equation $c_k = H(m_k||y^{r_k}Z_k^{-c_k}||V_k^{s_k+c_k2^{\lambda_4}}Z_k^{-c_k})$ required for message signature pairs, in the ROM this equation necessitates that **Ad** selects c_k from the given hash values $H_{\pi_k} = H(m_k||\phi_{\pi_k}||\psi_{\pi_k})$, where $\pi_k \in \{1, \cdots, t_h\}$ is an arbitrary function $k \mapsto \pi_k$ that selects $\phi_{\pi_k}, \psi_{\pi_k}$ from the computed group elements ϕ_i, ψ_i. Otherwise, the equality only holds with probability $1/2^{\lambda_1}$ as the hash value is random.

Let $c_j = H(m_j||\phi_{\pi_j}||\psi_{\pi_j})$ where $\phi_{\pi_j} = y^{r_j}Z_j^{-c_j}$ and $\psi_{\pi_j} = V_j^{s_j+c_j2^{\lambda_4}}Z_j^{-c_j}$ holds for the output $(m_j, Z_j, V_j, c_j, r_j, s_j)$, which determines π_j except that there is a collision $H(m_i||\phi_{\pi_i}||\psi_{\pi_i}) = H(m_j||\phi_{\pi_j}||\psi_{\pi_j})$ with $m_i = m_j$. However, the collision probability is at most $C_{t_h}^2/2^{\lambda_1} + 4C_{t_s}^2/\Phi(N)$ and negligible.

Let $n = n_1 \cdots n_I, \log_{z_j} Z_j = \theta_j, \log_{v_j} V_j = \vartheta_j$, where θ_j and ϑ_j are controlled by **Ad**. The equations $r'_\ell = \beta_\ell + c'_\ell\delta_\ell$, $s'_\ell = \omega_\ell + c'_\ell(p - 2^{\lambda_4})$, $y^{r_j}Z_j^{-c_j} = \phi_{\pi_j}$ and $V_j^{s_j+c_j2^{\lambda_4}}Z_j^{-c_j} = \psi_{\pi_j}$ imply

$$(1)\ nr_j = d_{\pi_j,-1} + nd_{\pi_j,0} + n\sum_{\ell=1}^{l} d_{\pi_j,\ell,3}r'_\ell + n\sum_{\ell=1}^{l}(d_{\pi_j,\ell,1} - d_{\pi_j,\ell,3}c'_\ell + \theta_j c_j)\delta + n\sum_{\ell=1}^{l}(d_{\pi_j,\ell,2}\delta + d_{\pi_j,\ell,4}s'_\ell - d_{\pi_j,\ell,4}(p - 2^{\lambda_4}))/p \bmod \Phi(N)/4;$$

$$(2)\ n\vartheta_j(s_j + c_j2^{\lambda_4}) = (e_{\pi_j,-1} + ne_{\pi_j,0} + n\sum_{\ell=1}^{l} e_{\pi_j,\ell,3}r'_\ell + n\sum_{\ell=1}^{l}(e_{\pi_j,\ell,1} - e_{\pi_j,\ell,3}c'_\ell + \theta_j c_j)\delta - e_{\pi_j,\ell,4})p + n\sum_{\ell=1}^{l} e_{\pi_j,\ell,2}\delta + n\sum_{\ell=1}^{l}(e_{\pi_j,\ell,4}s'_\ell + e_{\pi_j,\ell,4}2^{\lambda_4})) \bmod \Phi(N)/4.$$

If $\sum_{\ell=1}^{l}(d_{\pi_j,\ell,1} - d_{\pi_j,\ell,3}c'_\ell + \theta_j c_j)\delta + \sum_{\ell=1}^{l}(d_{\pi_j,\ell,2}\delta + d_{\pi_j,\ell,4}s'_\ell - d_{\pi_j,\ell,4}(p - 2^{\lambda_4}))/p = 0$, then **Ad** can easily compute the correct r_j be setting $n|d_{\pi_j,-1}$. In this case, the equation (1) does not depend on the secret values δ, p and we have $d_{\pi_j,-1} + nd_{\pi_j,0} + n\sum_{\ell=1}^{l} d_{\pi_j,\ell,3}r'_\ell$, where the signers' responses $r'_1, \cdots, r'_l$ are known and the coefficients and $d_{\pi_j,-1}, d_{\pi_j,0}, d_{\pi_j,\ell,3}$ are controlled by **Ad**. Conversely, **Ad** must select $c'_1, \cdots, c'_l, \theta_j$ to respectively zero the values $\sum_{\ell=1}^{l}(d_{\pi_j,\ell,1} - d_{\pi_j,\ell,3}c'_\ell + \theta_j c_j)\delta$ and $\sum_{\ell=1}^{l}(d_{\pi_j,\ell,2}\delta + d_{\pi_j,\ell,4}s'_\ell - d_{\pi_j,\ell,4}(p - 2^{\lambda_4}))/p$ in (1). Otherwise, the equations hold with negligible probability, as δ, p are statistically independent of the non-group data $d_{\pi_j,\ell,1}, d_{\pi_j,\ell,2}, d_{\pi_j,\ell,3}, d_{\pi_j,\ell,4}$, and thus **Ad**'s probability of success is also negligible. This shows that **Ad** must solve the equation (1) for each valid message signature pair, which implies a solution $c'_1, \cdots, c'_l$ of an extended ROS problem $\sum_{\ell=1}^{l}(d_{\pi_j,\ell,1} - d_{\pi_j,\ell,3}c'_\ell + \theta_j c_j) = 0$. Similarly, to make equations (2) to hold with a non-negligible probability, the solution of the extended ROS problems in (1) must be a solution of the extend ROS problem $\sum_{\ell=1}^{l}(e_{\pi_j,\ell,1} - e_{\pi_j,\ell,3}c'_\ell + \theta_j c_j) = 0$ in (2). This completes the proof.

Trading Time for Space: Towards an Efficient IBE Scheme with Short(er) Public Parameters in the Standard Model

Sanjit Chatterjee and Palash Sarkar

Applied Statistics Unit,
Indian Statistical Institute,
203, B.T. Road, Kolkata
700108, India
{sanjit_t, palash}@isical.ac.in

Abstract. At Eurocrypt 2005, Brent Waters proposed an efficient Identity Based Encryption scheme which is secure in the standard model. One drawback of this scheme is that the number of elements in the public parameter is rather large. Here we propose a generalisation of Waters scheme. In particular, we show that there is an interesting trade-off between the tightness of the security reduction and smallness of the public parameter. For a given security level, this implies that if one reduces the number of elements in public parameter then there is a corresponding increase in the computational cost due to the increase in group size. This introduces a flexibility in choosing the public parameter size without compromising in security. In concrete terms, to achieve 80-bit security for 160-bit identities we show that compared to Waters protocol the public parameter size can be reduced by almost 90% while increasing the computation cost by 30%. Our construction is proven secure in the standard model without random oracles. Additionally, we show that CCA security can also be achieved through the reduction to oracle decision bilinear Diffie-Hellman problem (OBDH).

Keywords: identity based encryption, standard model, security, parameter size.

1 Introduction

The area of public key cryptography called Identity Based Encryption (IBE) has witnessed a rapid progress in recent times. Initially proposed by Shamir [23], it was as well a challenge to the crypto community to come out with a practical IBE scheme. Boneh and Franklin [6, 7] were first to define a security model for IBE and gave an implementable solution based on the Bilinear Diffie-Hellman (BDH) problem. There is another construction due to Cocks [13] that uses quadratic residues modulo a composite. The security of these encryption schemes were proved in the random oracle model [11], i.e., the security of these schemes requires cryptographic hash functions that are modelled as random oracles. However,

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 424–440, 2006.
© Springer-Verlag Berlin Heidelberg 2006

such hash functions do not exist in reality. Consequently, there were several works such as [14, 2] to construct IBE schemes secure without the random oracle model. They used a weaker notion of security called *selective*-ID model in which an adversary has to commit in advance which identity it wants to attack.

Finally, Boneh and Boyen came out with a scheme for IBE [3] that is secure in the standard model without random oracles. Their work was more of a feasibility study. It solved the open problem but was not practical to be implemented. This work was soon supplemented by that of Waters [24]. Using a method from [2] and introducing a new trick, it provided an improved IBE scheme that is secure in the standard model without random oracle.

However, one disadvantage of the scheme in [24] is the requirement of a rather large public parameter file. If identities are represented by a bit string of length n, then the scheme requires a vector of length n to be maintained as part of public parameter, where each element of the vector is a point on a suitable elliptic curve group.

OUR CONTRIBUTION: We provide a generalisation of the identity based encryption scheme of Waters [24]. This generalisation shows that if one tries to reduce the number of elements in the public parameter then there is a corresponding degradation in the security reduction. In other words, a trade-off is involved in the tightness of security reduction and smallness of public parameter. The trade-off between tightness and smallness can be converted to a trade-off between group size and smallness of public parameter. When desiring a specific security level, the loss of security due to loss of tightness in the security reduction can be compensated by working in a larger group. This increses the bit length of representation of the elements in the public parameter but the number of elements in the public parameters decreases so drastically that there is a significant reduction in the overall size of the public parameter. The increse in group size in turn affects the efficiency of the protocol. Thus, the trade-off is actually between the space required to store the parameters and the time required to execute the protocol. For example, if identities are represented by 160-bit strings, then Waters protocol require to store 160 extra elements (EC points) as part of the public parameter. Alternatively, using our generalisation if one wants to store 16 elements, then to achieve 80-bit security, compared to Waters protocol the space requirement reduces by around 90% while the computation cost increases by around 30%.

– Like Waters, applying Naor's technique, our scheme can also be easily converted to a signature scheme where the underlying security assumption is the computational Diffie-Hellman problem.
– Our construction resembles closely the construction of Waters [24] and security against the chosen ciphertext attack (i.e., the CCA security) of the former follows from that of the later by constructing a 2 level hierarchical identity based encryption scheme (HIBE) and applying the technique of [15]. As an alternative, we show that CCA security can also be achieved by assuming the hardness of the oracle bilinear decision Diffie-Hellman assumption (OBDH).

2 Waters Construction

Waters has recently proposed an efficient identity based encryption scheme without random oracle [24]. We first briefly describe his construction. The relevant definitions of bilinear map, IBE protocol, its security model and hardness assumption are given in Appendix A

Waters IBE: Let $G_1 = \langle P \rangle$, G_2 and $e()$ be as defined in Section A.1. Here, identities are represented as bitstrings of length n.

Setup: Randomly choose a secret $x \in Z_p$. Set $P_1 = xP$, then choose $P_2 \in G_1$ at random. Further, choose a random element $U' \in G_1$ and a random n-length vector $\overrightarrow{U} = \{U_1, \ldots, U_n\}$, whose elements are from G_1. The master secret is xP_2 whereas the public parameters are $\langle P, P_1, P_2, U', \overrightarrow{U} \rangle$. Also $e()$ is publicly known.

Key Generation: Let $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_n) \in \{0,1\}^n$ be any identity. A secret key for v is generated as follows. Choose a random $r \in Z_p^*$, then the private key for v is

$$D_{\mathsf{v}} = (xP_2 + rV, rP).$$

where

$$V = U' + \sum_{\{i:\mathsf{v}_i=1\}} U_i.$$

Encryption: Any message $M \in G_2$ is encrypted for an identity v as

$$C = (e(P_1, P_2)^t M, tP, tV),$$

where t is a random element of Z_p and V is as defined in key generation algorithm.

Decryption: Let $C = (C_1, C_2, C_3)$ be a ciphertext and v be the corresponding identity. Then we decrypt C using secret key $D_{\mathsf{v}} = (D_1, D_2)$ by computing $C_1 e(D_2, C_3)/e(D_1, C_2)$.

3 Our Generalisation

Here we describe our generalisation of Waters scheme. The groups $G_1 = \langle P \rangle$, G_2 and the map $e()$ are as already defined in Section A.1. In the following, we assume the message space $\mathcal{M}$ is G_2, the cipher space $\mathcal{C}$ is $G_2 \times G_1 \times G_1$.

Note that, in Waters scheme identities are represented as n-bit strings. Because of this representation, Waters requires to store n elements of G_1 i.e., $\overrightarrow{U}$ in the public parameter. Depending upon the choice of representation of the identities we can change the size of the public parameter.

Let $N = 2^n$, then we can consider the identities as elements of Z_N and one extreme case would be to consider the identities just as elements of Z_N. A more moderate approach, however, is to fix *a-priori* a size parameter ℓ, where

$1 < \ell \leq n$. In this case, an identity v is represented as $\mathsf{v} = (\mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_\ell)$, where $\mathsf{v}_i \in Z_{N^{1/\ell}}$ i.e., each v_i is an n/ℓ bit string. (If identities are considered to be bit strings of arbitrary length, then as in Waters protocol we hash them into Z_N using a collision resistant hash function.)

In this case the protocol is changed to the following, which we call IBE-SPP(ℓ).

IBE-SPP(ℓ) **with** $1 < \ell \leq n$

Setup: Randomly choose a secret $x \in Z_p$. Set $P_1 = xP$, then choose $P_2 \in G_1$ at random. Further, choose random elements $U', U_1, U_2, \ldots, U_\ell \in G_1$. The master secret is xP_2 whereas the public parameters are $\langle P, P_1, P_2, U', U_1, U_2, \ldots, U_\ell \rangle$. Also $e()$ is publicly known.

Key Generation: Let v be any identity, a secret key for v is generated as follows. Choose a random $r \in Z_p^*$, then the private key for v is

$$D_{\mathsf{v}} = (xP_2 + rV, rP).$$

where $V = U' + \sum_{i=1}^{\ell} \mathsf{v}_i U_i$.

Encryption, Decryption: As in Waters IBE with the modified definition of V.

Note that, for $\ell = n$ this is exactly Waters protocol. For $\ell = 1$, some minor modifications in the above scheme give a protocol where the additional requirement in the public parameter is just a single element of G_1 as described below.

IBE-SPP*(1)*

Setup: Randomly choose a secret $x \in Z_N$. Set $P_1 = xP$, then choose $P_2 \in G_1$ at random. Further, choose a random element $U' \in G_1$. The master secret is xP_2 whereas the public parameters are $\langle P, P_1, P_2, U' \rangle$. Also $e()$ is publicly known.

Key Generation: Let v be any identity. A secret key for v is generated as follows. Choose a random $r \in Z_p^*$, then the private key for v is

$$D_{\mathsf{v}} = (xP_2 + rV, rP).$$

where $V = U' + \mathsf{v}P_2$.

Here also the Encryption and Decryption algorithms remain unaltered and this is essentially the Boneh-Boyen scheme of [2] in the *adaptive*-ID model.

Efficiency: Consider IBE-SPP(ℓ) with $1 < \ell \leq n$. Let $\mathsf{cost}(V)$ be the cost of computing V. The cost of key generation is two scalar multiplications over G_1 plus $\mathsf{cost}(V)$. By including $e(P_1, P_2)$ instead of P_1, P_2 in the public parameter, we can avoid the pairing computation during encryption. So the cost of encryption is one exponentiation over G_2, two scalar multiplications over G_1 plus $\mathsf{cost}(V)$. The cost of decryption is two pairings, one multiplication and one inversion over

G_2. The effect of ℓ is in $\mathsf{cost}(V)$ and affects key generation and encryption costs but does not affect decryption cost.

We first consider the costs of scalar multiplication over G_1 and exponentiation over G_2. As mentioned earlier, G_1 is an elliptic curve group. Let $\mathbb{F}_a$ denote the base field over which G_1 is defined. Then G_2 is a subgroup of $\mathbb{F}_a^k$, where k is the MOV degree. Additions and doublings over G_1 translate into a constant number of multiplications over $\mathbb{F}_a$. The actual number is slightly different for addition and doubling, but we will ignore this difference. Let $|\mathbb{F}_a|$ be the size of the representation of an element of $\mathbb{F}_a$. Assuming the cost of multiplication over G_1 is approximately equal to $|\mathbb{F}_a|^2$, the cost of a scalar multiplication over G_1 is equal to $c_1|\mathbb{F}_a|^3$ for some constant c_1. One can also show that the cost of exponentiation over G_2 is equal to $c_2|\mathbb{F}_a|^3$. Thus, the total cost of scalar multiplication and exponentiation is equal to $c|\mathbb{F}_a|^3$.

The cost of computing V amounts to computing ℓ scalar multiplications where each multiplier is an (n/ℓ)-bit string. On an average, the cost of each such multiplication will be $n/2\ell$ additions and $(n/\ell - 1)$ doublings over G_1. Hence, the total cost of computing V is $n/2$ additions and $(n - \ell)$ doublings over G_1. This cost is equal to $d(3/2 - \ell/n)n|\mathbb{F}_a|^2$ for some constant d.

We consider the cost of encryption. The total cost is

$$c|\mathbb{F}_a|^3 + d(3/2 - \ell/n)n|\mathbb{F}_a|^2 = \left(c + d \times \frac{n}{|\mathbb{F}_a|}\left(\frac{3}{2} - \frac{\ell}{n}\right)\right)|\mathbb{F}_a|^3. \qquad (1)$$

This cost is minimum when $\ell = n$ (as in Waters protocol). The maximum value of the coefficient of $|\mathbb{F}_a|^3$ is $(c + (3nd)/(2|\mathbb{F}_a|))$ whereas the minimum value is $(c + (nd)/(2|\mathbb{F}_a|))$. The value of $|\mathbb{F}_a|$ is usually greater than n and hence the value of $(nd)/(2|\mathbb{F}_a|)$ will be a small constant and hence there is not much effect of ℓ on the total cost of encryption. A similar analysis shows that the effect of ℓ is also not very significant on the cost of key generation. We note, however, that key generation is essentially a one-time offline activity.

3.1 Security Reduction

In this section, we only consider the security of $\mathsf{IBE-SPP}(\ell)$ against chosen plaintext attacks ($\mathsf{IND\text{-}ID\text{-}CPA}$). (The extension to chosen ciphertext attack is considered later.) The security (in the sense of $\mathsf{IND\text{-}ID\text{-}CPA}$) of the identity based encryption scheme ($\mathsf{IBE\text{-}SPP}(\ell)$) developed above can be reduced from the hardness of the DBDH problem as stated in the following theorem.

Theorem 1. *For $t \geq 1$, $q \geq 1$ let $\epsilon = \mathsf{Adv}^{\mathsf{IBE-SPP}}(\ell)(t, q)$. Then,*

$$\epsilon \leq 16q(\mu_\ell + 1)\mathsf{Adv}^{\mathsf{DBDH}}(t + O(\tau q) + \chi),$$

where identities are chosen from Z_N; ℓ is a size parameter with $1 < \ell \leq \lg N$; $\mu_\ell = \ell(N^{1/\ell} - 1)$; $\chi = O(\epsilon^{-2}\ln(\epsilon^{-1})\lambda^{-1}\ln(\lambda^{-1}))$; $\lambda = \frac{1}{4q(\mu_\ell+1)}$; and τ is the time for a scalar multiplication in G_1.

Note that, for $\ell = n$ we have $\mu_n = n$ and one gets the corresponding relationship for Waters protocol. The component χ in the time comes due to the

so-called "artificial abort" technique. The proof of Theorem 1 essentially follows the technique already developed by Boneh-Boyen [3] and Waters [24] and we defer it to Section 5.

3.2 Signature

It is an observation of Naor that any identity based encryption scheme can be converted to a signature scheme. Waters in his paper [24] has given a construction of a signature scheme based on his IBE scheme. A similar construction is possible for the generalised scheme IBE-SPP(ℓ) which we detail here. The sketch of the security reduction is provided in Appendix C.

Let $G_1 = \langle P \rangle$, G_2 and $e()$ be as defined in Section A.1. Messages are assumed to be elements of Z_N. Alternatively, if messages are assumed to be bit strings of arbitrary length, then we use a collision resistant hash function to map the messages into Z_N.

Setup: Choose a random x in Z_p. Let $P_1 = xP$. Next, choose random points $P_2, U', U_1, \ldots, U_l$ from G_1. The public key is $\langle P, P_1, P_2, U', U_1, \ldots, U_\ell \rangle$ and the secret key is xP_2.

Signing: Let $M = (m_1, m_2, \ldots, m_\ell)$ is the message to be signed, where each $m_i, 1 \leq i \leq \ell$ belongs to $Z_{N^{1/\ell}}$. To generate a signature on M, first choose a random $r \in Z_P^*$. Then the signature is

$$\sigma_M = (xP_2 + rV, rP),$$

where $V = U' + \sum_{i=1}^{\ell} m_i U_i$

Verification: Given a message $M = (m_1, m_2, \ldots, m_\ell)$ and a signature $\sigma = (\sigma_1, \sigma_2)$ on M, one accepts σ as a valid siganture on M if

$$e(\sigma_1, P) = e(P_1, P_2)e(\sigma_2, V)$$

where $V = U' + \sum_{i=1}^{\ell} m_i U_i$.

4 Concrete Security

From the security reduction of previous section we observe that any (t, q, ϵ) adversary $\mathcal{A}$ against IBE-SPP(ℓ) can actually be used to build an algorithm $\mathcal{B}$ to solve the DBDH problem over (G_1, G_2) which runs in time t' and has a probability of success ϵ'. Then $t' = t + O(\tau q) + \chi \approx t + c\tau q + \chi$ for some constant c and $\epsilon' \approx \epsilon/\delta$ where τ is the time for a group operation in G_1 and δ is the corresponding degradation in the security reduction. Resistance of IBE-SPP(ℓ) against $\mathcal{A}$ can be quantified as $\rho^{(\ell)}_{|\mathcal{A}} = \lg(t/\epsilon)$. To assert that IBE-SPP(ℓ) has at least 80-bit security, we must have $\rho^{(\ell)}_{|\mathcal{A}} \geq 80$ for all possible $\mathcal{A}$. Similarly, the resistance of DBDH against $\mathcal{B}$ can be quantified as

$$\rho_{|_{\mathcal{B}}} = \lg\left(\frac{t'}{\epsilon'}\right) \approx \lg\left(\delta \times \frac{t + c\tau q + \chi}{\epsilon}\right) = \lg(\delta(A_1 + A_2))$$

where $A_1 = t/\epsilon$ and $A_2 = (c\tau q + \chi)/\epsilon$. We now use $\max(A_1, A_2) \leq A_1 + A_2 \leq 2\max(A_1, A_2)$. Since a factor of two does not significantly affect the analysis we put $\rho_{|_{\mathcal{B}}} = \lg(\delta \times \max(A_1, A_2))$. By our assumption, $A_1 = t/\epsilon \geq 2^{80}$ and hence $\max(A_1, A_2) \geq A_1 \geq 2^{80}$. This results in the condition $\rho_{|_{\mathcal{B}}} \geq 80 + \lg\delta$.

Thus, if we want IBE-SPP(ℓ) to have 80-bit security, then we must choose the group sizes of G_1, G_2 in such a way that the best possible algorithm for solving DBDH in these groups takes time at least $2^{80+\lg\delta}$. Hence, in particular the currently best known algorithm for solving the DBDH should also take this time. Currently the only method to solve the DBDH problem over (G_1, G_2) is to solve the discrete log problem (DLP) over either G_1 or G_2. The best known algorithm for the former is the Pollard's rho method while that for the later is number/function field sieve. Thus, if we want IBE-SPP(ℓ) to have 80-bit security, then we must choose the group sizes such that, $2^{80+\lg\delta} \leq \mathsf{min}(t_{G_1}, t_{G_2})$, where t_{G_i} stands for the time to solve DLP in G_i for $i \in \{1, 2\}$.

We have assumed that G_1 is a group of elliptic curve points of order p defined over a finite field $\mathbb{F}_a$ (a is a prime power). Suppose G_2 is a subgroup of order p of the finite field $\mathbb{F}_{a^k}$ where k is the MOV degree. The Pollard's rho algorithm to solve ECDLP takes time $t_{G_1} = O(\sqrt{p})$, while the number/function field seive method to solve the DLP in $\mathbb{F}_{a^k}$ takes time $t_{G_2} = O(e^{c^{1/3}\ln^{1/3}a^k \ln^{2/3}(\ln a^k)})$ where $c = 64/9$ (resp. $32/9$) in large characteristic fields (resp. small characteristic fields).

4.1 Space/Time Trade-Off

In this section we parametrize the quantities by ℓ wherever necessary. Let, $\delta^{(\ell)}$ denote the degradation factor in IBE-SPP(ℓ). We have already noted in Section 3 that $\ell = n$ stands for Waters protcol. $\delta^{(\ell)}$ and hence $\rho^{(\ell)}$ is minimum when $\ell = n$ and we use this as a bench mark to compare with other values of ℓ. Suppose $\Delta\rho^{(\ell)} = \rho^{(\ell)} - \rho^{(n)} = \lg(\delta^{(\ell)}/\delta^{(n)}) = (n/\ell) - \lg(n/\ell)$. This parameter $\Delta\rho^{(\ell)}$ gives us an estimate of the extra bits required in case of IBE-SPP(ℓ), to achieve the same security level as that of IBE-SPP(n) i.e., Waters protocol.

Suppose, $|p^{(\ell)}|$ (resp. $|G_2^{(\ell)}|$) denotes the bit length of representation of $p^{(\ell)}$ (resp. an element of $G_2^{(\ell)}$). Like [16], we assume that the adversary $\mathcal{A}$ is allowed to make a maximum of $q = 2^{30}$ number of queries. For a given security level, we can now find the values of $|p^{(\ell)}|$ and $|G_2^{(\ell)}|$ for IBE-SPP(ℓ) based on the bit length of the identities (i.e., n), q and ℓ. Note that, the value of $|p^{(\ell)}|$ (resp. $|G_2^{(\ell)}|$) thus obtained is the *minimum* required to avoid the Pollards rho (resp. number/function field seive) attack. In our comparison, the MOV degree k is taken to be same for different values of ℓ and $|G_2^{(\ell)}| = k\lg a$ ($G_2^{(\ell)}$ is a multiplicative subgroup of order $p^{(\ell)}$ of the finite field $\mathbb{F}_a^k$). As already noted, the value of $p^{(\ell)}$ is given by Pollard's rho. On the other hand, the logarithm of the size of $G_1^{(\ell)}$ is equal to

Table 1. *Approximate group sizes for attaining 80-bit security for* IBE-SPP(ℓ) *for different values of* ℓ *and relative space and time requirement. The first part corresponds to* $n = 160$ *and the second to* $n = 256$.

ℓ	$\Delta\rho^{(\ell)}$	$\lvert p^{(\ell)}\rvert$	$\lvert G_2^{(\ell)}\rvert$		$\alpha^{(\ell)}$		$\beta^{(\ell)}$	
			(a)	(b)	(a)	(b)	(a)	(b)
160	–	246	1891(2225)	3284(3872)	–	–	–	–
4	34	314	3269(3730)	5721(6538)	4.3(4.2)	4.4(4.2)	5.17(4.71)	5.46(4.81)
8	15	276	2443(2831)	4258(4944)	6.5(6.4)	6.5(6.4)	2.16(2.06)	2.18(2.08)
16	6	258	2102(2457)	3655(4288)	11.1(11.0)	11.1(11.1)	1.37(1.35)	1.38(1.35)
32	2	250	1960(2300)	3405(4006)	20.7(20.7)	20.7(20.7)	1.11(1.11)	1.12(1.11)
80	1	248	1924(2262)	3344(3939)	50.9(50.8)	50.9(50.9)	1.05(1.05)	1.06(1.05)
256	–	246	1891(2225)	3284(3872)	–	–	-	–
4	58	362	4530(5090)	7959(8954)	3.7(3.6)	3.8(3.6)	13.75(11.97)	14.24(12.37)
8	27	300	2948(3381)	5151(5919)	4.9(4.7)	4.9(4.8)	3.79(3.51)	3.86(3.57)
16	12	270	2326(2703)	4051(4717)	7.7(7.6)	7.7(7.6)	1.86(1.79)	1.88(1.81)
32	5	256	2066(2417)	3592(4212)	13.7(13.6)	13.7(13.6)	1.30(1.28)	1.31(1.29)
64	2	250	1960(2300)	3405(4006)	25.9(25.8)	25.9(25.9)	1.11(1.11)	1.11(1.11)
128	1	248	1924(2262)	3344(3939)	50.9(50.8)	50.9(50.9)	1.05(1.05)	1.06(1.05)

$\mathsf{max}(p^{(\ell)}, |G_2^{(\ell)}|/k)$. For relatively small MOV degree (i.e., $k \leq 6$), $|G_2^{(\ell)}|/k > |p^{(\ell)}|$ and so the logarithm of the size of $G_1^{(\ell)}$ is equal to $|G_2^{(\ell)}|/k = |\mathbb{F}_a^{(\ell)}|$. For a given ℓ, we have to store ℓ elements of $G_1^{(\ell)}$ in the public parameter file and a scalar multiplication in $G_1^{(\ell)}$ takes time proportional to $(|\mathbb{F}_a^{(\ell)}|)^3$.

Now, we are in a position to compare the space requirement in the public parameter file and the time requirement for a scalar multiplication in $G_1^{(\ell)}$ for different values of ℓ. Let $\alpha^{(\ell)} = \frac{\ell \times |G_1^{(\ell)}|}{n \times |G_1^{(n)}|} \times 100$ i.e., the relative amount of space (expressed in percentage) required to store the public parameters in case of IBE-SPP(ℓ) with respect to IBE-SPP(n) and $\beta^{(\ell)} = |\mathbb{F}_a^{(\ell)}|^3/|\mathbb{F}_a^{(n)}|^3$, i.e., the relative increase in time for scalar multiplication in $G_1^{(\ell)}$in the case of IBE-SPP(ℓ) with respect to IBE-SPP(n). Note that, $\beta^{(\ell)}$ can be computed from $|G_2^{(\ell)}|$ and $|G_2^{(n)}|$ since k cancels out from both numerator and denominator. An analysis similar to the efficiency consideration in Section 3 shows that pairing computation is also of order $|\mathbb{F}_a^{(\ell)}|^3$ (but with a larger constant factor). So, the ratio $\beta^{(\ell)}$ also holds for pairing computation and exponentiation in case of IBE-SPP(ℓ) with respect to Waters protocol.

In Table 1 we sum-up these results for $n = 160$ and 256 for different values of ℓ ranging from 4 to n for 80-bit security. The subcolumns (a) and (b) under $\alpha^{(\ell)}$ and $\beta^{(\ell)}$ stand for the values obtained for general characteristic field and field of characteristic three respectively. The values of $|G_2^{(\ell)}|, \alpha^{(\ell)}, \beta^{(\ell)}$ are computed using the formula as suggested in [16] (see Section 3); while in parenthesis we give the corresponding values as computed from the formula obtained from [21] (as given in Section 3 of [16]). Note that, the values of $\alpha^{(\ell)}$ and $\beta^{(\ell)}$ being the

ratio of two quantities remain more or less invariant whether the underlying field is a general characteristic field or a field of characteristic three or which formula (of [16] or of [21]) is used.

Public parameter consists of $(\ell + 4)$ elements of G_1. From Table 1, for 80-bit security in general characteristic fields using EC with MOV degree 2, the public parameter size for Waters protocol will be around 37 kilobyte (kb) for 160-bit identities and 59 kb for 256-bit identities. The corresponding values in case of $\mathsf{IBE\text{-}SPP}(\ell)$ with $\ell = 16$ will be around 4 kb and 4.5 kb respectively. Similarly, in characteristic three field EC with MOV degree 6, the corresponding values are respectively 21.5 kb and 34.2 kb and for $\mathsf{IBE\text{-}SPP}(\ell)$ with $\ell = 16$ these are respectively 2.4 kb and 2.64 kb. There is an associated increase in computation cost by 30%. In typical applications, the protocol will be used in a key encapsulation mechanism (KEM). Thus the encryption and decryption algorithms will be invoked once for a message irrespective of its length. Also the key generation procedure is essentially an one-time offline activity. In view of this, the increase in computation cost will not substantially affect the throughput. On the other hand, the significant reduction in space requirement will be an advantage in implementing the protocol and also in reducing the time for downloading or transmitting the public parameter file over the net. Overall, we suggest $\ell = 16$ to be a good choice for implementing the protocol.

5 Security Proof

We prove Theorem 1 through a reductionist security argument. The proof is very similar to that of Waters and we describe only the essential features.

Proof : Suppose $\mathcal{A}$ is a (t, q)-CPA adversary for $\mathsf{IBE} - \mathsf{SPP}(\ell)$. Then we construct an algorithm $\mathcal{S}$ for DBDH running in time $(t + O(\tau q + \chi)$ such that, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IBE}-\mathsf{SPP}(\ell)} \leq 16q(\mu_\ell + 1)\mathsf{Adv}_{\mathcal{S}}^{\mathsf{DBDH}}$, where $\mu_\ell = \ell(N^{1/\ell} - 1)$. $\mathcal{S}$ will take as input a 5-tuple $\langle P, aP, bP, cP, Z\rangle$ where P is a generator of G_1, $aP, bP, cP \in G_1$ and $Z \in G_2$. We define the following game between $\mathcal{S}$ and $\mathcal{A}$.

Setup: $\mathcal{S}$ first chooses random $x, x_1, \ldots, x_\ell \in Z_m$ where $m = 4q$ (justified later); random
$y, y_1, \ldots, y_\ell \in Z_p$ and a random $k \in \{0, \ldots, \mu_\ell\}$. It then defines three functions: $F(\mathsf{v}) = p - mk + x + \sum_{i=1}^{\ell} x_i \mathsf{v}_i$, $J(\mathsf{v}) = y + \sum_{i=1}^{\ell} y_i \mathsf{v}_i$ and

$$K(\mathsf{v}) = \begin{cases} 0 \text{ if } x + \sum_{i=1}^{\ell} x_i \mathsf{v}_i \equiv 0 \bmod m \\ 1 \text{ otherwise} \end{cases}$$

Here, $F(\mathsf{v})$ and $K(\mathsf{v})$ are defined in such a way that $K(\mathsf{v}) \neq 0$ implies $F(\mathsf{v}) \not\equiv 0 \bmod p$. Next, $\mathcal{S}$ assigns $P_1 = aP$, $P_2 = bP$, $U' = (p - mk + x)P_2 + yP$ and $U_i = x_i P_2 + y_i P$ for $1 \leq i \leq \ell$. It provides $\mathcal{A}$ the public parameters $\langle P, P_1, P_2, U', U_1, \ldots, U_\ell\rangle$. Everything else is internal to $\mathcal{S}$. Note that from $\mathcal{A}$'s point of view the distribution of the public parameters is identical to the distribution of the public parameters in an actual setup.

Phase 1: The adversary $\mathcal{A}$ issues key extraction queries. Suppose, the adversary asks for the private key corresponding to an identity v. $\mathcal{S}$ first checks whether $K(\mathsf{v}) = 0$ and aborts in that situation and outputs a random bit. Otherwise, it gives $\mathcal{A}$ the pair

$$(D_1, D_2) = \left(-\frac{J(\mathsf{v})}{F(\mathsf{v})} P_1 + r(F(\mathsf{v})P_2 + J(\mathsf{v})P), \frac{-1}{F(\mathsf{v})} P_1 + rP \right)$$

where r is chosen at random from Z_p. As in Waters proof it is possible to show that (D_1, D_2) is a valid private key for v following the proper distribution. $\mathcal{S}$ will be able to generate this pair (D_1, D_2) if and only if $F(\mathsf{v}) \not\equiv 0$, for which it suffices to have $K(\mathsf{v}) \neq 0$.

Challenge: At this stage the adversary $\mathcal{A}$ submits two messages $M_0, M_1 \in G_2$ and an identity v^* with the constraint that it has not asked for the private key of v^* in Phase 1. $\mathcal{S}$ aborts if $F(\mathsf{v}^*) \neq 0$ and outputs a random bit. Otherwise, $\mathcal{S}$ chooses a random bit $\gamma \in \{0, 1\}$ and gives $\mathcal{A}$ the tuple $C' = \langle ZM_\gamma, cP, J(\mathsf{v}^*)cP \rangle$.

If $\langle P, aP, bP, cP, Z \rangle$ given to $\mathcal{S}$ is a valid DBDH tuple, i.e., $Z = e(P, P)^{abc}$ then C' is a valid encryption for M_γ. Since,

$$e(P, P)^{abc} = e(aP, bP)^c = e(P_1, P_2)^c$$

and using $F(\mathsf{v}^*) = p - mk + x + \sum_{i=1}^{\ell} x_i \mathsf{v}_i^* \equiv 0 \bmod p$ it is possible to show that $J(\mathsf{v}^*)cP = cV$. Note that, this condition is satisfied as long as $F(\mathsf{v}^*) \equiv 0 \bmod p$, which holds if $x + \sum_{j=1}^{\ell} x_j \mathsf{v}_j^* = km$.

Otherwise, Z is a random element of G_2 and C' gives no information about $\mathcal{S}$'s choice of γ.

Phase 2: This phase is similar to Phase 1, with the obvious restriction that $\mathcal{A}$ cannot ask for the private key of v^*. We note that the total number of key extraction queries together in Phase 1 and 2 should not exceed q.

Guess: $\mathcal{A}$ outputs a guess γ' of γ. Then $\mathcal{S}$ outputs $1 \oplus \gamma \oplus \gamma'$.
Suppose the adversary has not aborted upto this point. Waters introduces a technique whereby the simulator is allowed to abort under certain condition. The simulator samples the transcript it received from the adversary during the attack phase. Based on the sample, it decided whether to abort and output a random string. The rationale for such "artificial abort" is the following: The probability of abort during the attack phase depends on the adversarial transcript and can be different for different transcripts. The purpose of artificial abort is to ensure that the simulator aborts with (almost) the same probability for all adversarial queries. This ensures that the adversary's success is independent of whether the simulator aborts or not. The probability analysis performed by Waters in [24] requires this independence. For details of this method see [24]. Here we just note that the artificial abort stage requires an additional

$\chi = O(\epsilon^{-2}\ln(\epsilon^{-1})\lambda^{-1}\ln(\lambda^{-1}))$ time. Further, it is independent of the parameter ℓ which defines the generalisation over Waters [24] that we introduce here.

Let abort be the probability of the simulator aborting during the actual attack (as opposed to artificial abort) and let $\lambda = Pr[\overline{\mathsf{abort}}]$. In Appendix B, we calculate the lower bound of λ to be $\frac{1}{m(\mu_\ell+1)}(1-2\frac{q}{m})$. Using $m = 4q$ gives $\lambda \geq \frac{1}{4q(\mu_\ell+1)}$. Now using the analysis performed by Waters [24], we obtain

$$\epsilon \leq 16q(\mu_\ell+1)\ \mathsf{Adv}^{\mathsf{DBDH}}(t+O(\tau q)+\chi).$$

The time component of $O(\tau q)$ comes because of the scalar multiplications performed in Phase 1 and 2 of Key Generation (these scalar multiplications are the only computationally intensive part in the simulation). This completes the proof. □

Remark 1: Note that, in the simulation, only the computation of $F(\mathsf{v}), J(\mathsf{v}) \in Z_p$ depends on the size parameter ℓ. Once $F(\mathsf{v})$ and $J(\mathsf{v})$ are obtained, the key generation in Phase 1 and 2 and cipher text generation in Challenge is done through some scalar multiplications involving $F(\mathsf{v})$ and $J(\mathsf{v})$. Cost of computation of $F(\mathsf{v})$ and $J(\mathsf{v})$ are insignificant compared to the cost of a scalar multiplication. So the simulation time is independent of the size parameter ℓ.

Remark 2: The technique of "artificial abort" is new to security proofs and was introduced by Waters [24]. (It is not present in the security proof of Boneh and Boyen [3] which is also an identity based encryption protocol which is secure in the full model.) We feel that the technique of artificial abort can be avoided. This technique only lowers the probability of not aborting. Hence, it should be possible to directly work with the lower bound λ of not aborting, without actually going through the artificial abort step. Avoiding the artificial abort step will require performing a new probability analysis. We hope to do that in the future.

6 CCA Security

Recent works of Boneh, Canetti, Halevi and Katz [5, 8, 15] show how to build CCA secure encryption scheme from identity based encryption. One way to achieve CCA-security for our scheme is to follow the strategy suggested in [24]. As our scheme closely resembles that of [24] it is possible to build a hybrid 2-level HIBE [19, 18] in essentially the same way and the reduction follows.

We show that it is possible to take a different approach based on the oracle bilinear decision Diffie-Hellman (OBDH) assumption which is a variation of the ODH assumption used in [1]. The OBDH assumption is as follows [22].

- Instance : $\langle P, aP, bP, cP, \mathsf{str}\rangle$ where $a, b, c \in Z_p$ and $\mathsf{str} \in \{0,1\}^k$.
- Oracle : $\mathcal{H}_a(X, Y)$, with $X, Y \in G_1$. When invoked with (a_1P, b_1P) returns $H(a_1P, e(a_1P, a_2P)^a)$, where $H : G_1 \times G_2 \rightarrow \{0,1\}^k$ is a hash function.
- Restriction : Cannot query $\mathcal{H}_a(,)$ on (cP, bP).
- Task : Determine whether $\mathsf{str} = H(cP, e(cP, bP)^a)$ or str is random.

Any algorithm $\mathcal{A}$ for OBDH takes as input an instance $(P, aP, bP, cP, \mathsf{str})$ of OBDH and produces as output either zero or one. The advantage of an algorithm $\mathcal{A}$ in solving OBDH is formally defined in the following manner.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{OBDH}} = |\mathsf{Pr}[\mathcal{A} \text{ outputs } 1|E_1] - \mathsf{Pr}[\mathcal{A} \text{ outputs } 1|E_2]|$$

where E_1 is the event that $\mathsf{str} = H(cP, e(cP, bP)^a)$ and E_2 is the event that str is random. The quantity $\mathsf{Adv}^{\mathsf{OBDH}}(t, q)$ denotes the maximum of $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{OBDH}}$ where the maximum is taken over all adversaries running in time at most t and making at most q queries to the oracle $\mathcal{H}_a(,)$.

To suit into the OBDH assumption we modify our constructions of Section 3 as follows: **Setup** and **Key Generation** remain unaltered. To encrypt a message, we first generate a symmetric key $\mathsf{sym.key} = H(tP, e(P_1, P_2)^t)$. Then the cipher is $C = \langle tP, tV, y)\rangle$, where y is the encryption of the message using the symmetric key $\mathsf{sym.key}$. To decrypt, all that we need is $e(P_1, P_2)^t = e(D_1, tP)/e(D_2, tV)$ and then find $\mathsf{sym.key}$ using H.

Security: Breaking the (modified) IBE implies either solving OBDH or breaking the symmetric encryption scheme. The later we assume to be unbreakable under chosen ciphertext attack. CCA security under the OBDH assumption is expressed in the following theorem proof of which will be provided in the full version of the paper.

Theorem 2. *For $t \geq 1$, $q \geq 1$; $\mathsf{Adv}^{\mathsf{IBE}}(t, q) \leq 16q(\mu_\ell+1)\mathsf{Adv}^{\mathsf{OBDH}}(t+O(\tau q)+\chi)$, where identities are chosen from Z_N, $1 < \ell \leq \lg N$ is a size parameter, $\mu_\ell = \ell(N^{1/\ell} - 1)$.*

Note: Subsequent to the acceptance notification of this submission to ICISC 2005, we came to know that a paper describing a similar construction as ours has been posted on the eprint archive by David Naccache. (We also note that an earlier version of the present paper was submitted to Asiacrypt 2005, whose submission deadline was May 30, 2005.) Though the construction is similar, the paper by Naccache does not perform any concrete security analysis. In fact, the paper mentions that the loss of security due to the generalisation is "insignificant". As discussed in Section 4, this is not correct. In fact, the conversion of security degradation into a trade-off between time and space is original to our paper and is the most important feature of the generalisation of Waters scheme. On the other hand, we would like to mention that Naccache's paper presents a better exposition of the security proof than that given in Waters.

References

1. M. Abdalla, M. Bellare and P. Rogaway. DHIES : An encryption scheme based on the Diffie-Hellman problem, *Proceedings of CT-RSA 2001*, Lecture Notes in Computer Science, Springer-Verlag, pages 143–158.
2. D. Boneh, X. Boyen. Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles, EUROCRYPT 2004, LNCS, pp 223–238.

3. D. Boneh, X. Boyen. Secure Identity Based Encryption without Random Oracles. In *Proceedings of the Advances in Cryptology* – (CRYPTO'04), 2004.
4. D. Boneh, X. Boyen, E. Goh, Hierarchical Identity Based Encryption with Constant Size Ciphertext, EUROCRYPT 2005, Vol. 3494 of LNCS, pp 440-456.
5. D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. Journal Submission. Available from D. Boneh's website.
6. D. Boneh, M. Franklin. Identity Based Encryption from the Weil Pairing. CRYPTO – 2001, volume 2139 of LNCS, pp. 213–229, 2001.
7. D. Boneh, M. Franklin. Identity Based Encryption from the Weil Pairing. SIAM J. of Computing, Vol. 32, No. 3, pp. 586–615, 2003.
8. D. Boneh and J. Katz. Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity Based Encryption. In *Proceedings of RSA-CT '05*, LNCS 3376, pp. 87-103, 2005.
9. P. S. L. M. Barreto, H. Y. Kim, B. Lynn and M. Scott. Efficient Algorithms for Pairing-Based Cryptosystems. CRYPTO 2002, LNCS 2442, pp. 354–368.
10. P. S. L. M. Barreto and M. Naehrig. Pairing-Friendly Elliptic Curves of Prime Order. Cryptology ePrint Archive, Report 2005/133. Available from http://eprint.iacr.org/2005/133/. Accepted for presentation at SAC 2005.
11. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In ACM Conference on Computer and Communications Security - CCS 1993, pp 62-73, 1993.
12. X. Boyen, Q. Mei and B. Waters. Direct Chosen Ciphertext Security from Identity-Based Techniques. In 12th ACM Conference on Computer and Communication Security – CCS 2005, To appear. This version is available from Cryptology ePrint Archive, Report 2005/288.
13. C. Cocks. An Identity Based Encryption Scheme Based on Quadratic Residue. In Proceedings of the 8th IMA International Conference on Cryptography and Coding, pp 26–28, 2001.
14. R. Canetti, S. Halevi and J. Katz. A Forward-Secure Public-Key Encryption Scheme. In EUROCRYPT 2003, Volume 2656 of LNCS, pp 255-271. 2003.
15. R. Canetti, S. Halevi and J. Katz. Chosen-ciphertext Security from Identity Based Encryption. In *Proceedings of Eurocrypt 2004*. LNCS, 2004.
16. D. Galindo. The Exact Security of Pairing Based Encryption and Signature Schemes. Talk at Workshop on Provable Security, INRIA, Paris. November 3-5, 2004. Available from author's website.
17. S. Galbraith, K. Harrison and D. Soldera. Implementing the Tate Pairing. ANTS V, LNCS 2369, pp. 324-337, 2002.
18. C. Gentry and A. Silverberg, Hierarchical ID-Based Cryptography, ASIACRYPT 2002, LNCS, 2002.
19. J. Horwitz and B. Lynn. Towards Hierarchical Identity-Based Encryption. In EUROCRYPT 2002, pp 466–481, 2002.
20. N. Koblitz and A. Menezes, Another look at "provable security", Cryptology ePrint Archive, Report 2004/152, http://eprint.iacr.org/2004/152/, final version (to appear in Journal of Cryptology).
21. A. K. Lenstra and E. R. Verheul, Selecting Cryptographic Key Sizes, Jr. Cryptology 14(4), pp. 255-293 (2001)
22. P. Sarkar, HEAD: Hybrid Encryption with Delegated Decryption Capability, Proceedings of Indocrypt 2004, LNCS, pp 230-244.
23. A. Shamir. Identity-based Cryptosystems and Signature Schemes. CRYPTO 84, LNCS, pp 47-53, 1985.

24. B. Waters. Efficient Identity-Based Encryption Without Random Oracles. *Eurocrypt 2005*, Eurocrypt 2005, Vol. 3494 of LNCS, Also available from Cryptology ePrint Archive, Report 2004/180, http://eprint.iacr.org/2004/180/.

A Definitions

A.1 Cryptographic Bilinear Map

Let G_1 and G_2 be cyclic groups of same prime order p and $G_1 = \langle P \rangle$, where we write G_1 additively and G_2 multiplicatively. A mapping $e : G_1 \times G_1 \rightarrow G_2$ is called a cryptographic bilinear map if it satisfies the following properties:

- Bilinearity : $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in Z_p$.
- Non-degeneracy : If $G_1 = \langle P \rangle$, then $G_2 = \langle e(P, P) \rangle$.
- Computability : There exists an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.

Since $e(aP, bP) = e(P, P)^{ab} = e(bP, aP)$, $e()$ also satisfies the symmetry property. Modified Weil pairing [6] and Tate pairing [9, 17] are examples of cryptographic bilinear maps.

A.2 IBE Protocol

Following [6] an identity based encryption scheme is specified by four algorithms: Setup, Key Generation, Encryption and Decryption.

Setup: It takes input a security parameter and returns the system parameters together with the master key. The system parameters include a description of the message space, the ciphertext space and the identity space. They are publicly known while the master key is known only to the private key generator (PKG).

Key Generation: It takes as input an identity v and returns a private key D_{v}, using the master key. The identity v is used as the public key while D_{v} is the corresponding private key.

Encryption: It takes as input the identity v and a message from the message space and produces a ciphertext in the cipher space.

Decryption: It takes as input the ciphertext and the private key of the corresponding identity v and returns the message or bad if the ciphertext is not valid.

A.3 Security Model

Here we define indistinguishability under chosen ciphertext attack for identity based encryption schemes under a chosen identity. In this model, an adversary is allowed to choose adaptively the public key it wishes to attack. In concrete terms, security of an IBE scheme can be defined using the following game.

An adversary (whom we denote by $\mathcal{A}$) is allowed to query two oracles – a decryption oracle and a key-extraction oracle. At the initiation it is provided with the system public parameters.

Phase 1: Adversary $\mathcal{A}$ makes a finite number of queries where each query is addressed either to the decryption oracle or to the key-extraction oracle. In a query to the decryption oracle it provides the ciphertext as well as the identity under which it wants the decryption. Similarly, in a query to the key-extraction oracle, it asks for the private key of the identity it provides. Further, $\mathcal{A}$ is allowed to make these queries adaptively, i.e., any query may depend on the previous queries as well as their answers.

Challenge: At this stage $\mathcal{A}$ fixes an identity, v^* and two equal length messages M_0, M_1 under the (obvious) constraint that it has not asked for the private key of v^* and gets a ciphertext (C^*) corresponding to M_γ, where γ is chosen uniformly at random from $\{0, 1\}$.

Phase 2: $\mathcal{A}$ now issues additional queries just like Phase 1, with the (obvious) restriction that it cannot ask the decryption oracle for the decryption of C^* under v^* nor the key-extraction oracle for the private key of v^*.

Guess: $\mathcal{A}$ outputs a guess γ' of γ.

The advantage of the adversary $\mathcal{A}$ in attacking the IBE scheme is defined as:

$$\mathsf{Adv}^{\mathsf{IBE}}_{\mathcal{A}} = 2|Pr[(\gamma = \gamma')] - 1/2|$$

The quantity $\mathsf{Adv}^{\mathsf{IBE}}(t, q_{\mathsf{ID}}, q_{\mathsf{C}})$ denotes the maximum of $\mathsf{Adv}^{\mathsf{IBE}}_{\mathcal{A}}$ where the maximum is taken over all adversaries running in time at most t and making at most q_{C} queries to the decryption oracle and q_{ID} queries to the key-extraction oracle. Any IBE scheme secure against such an adversary is said to be secure against chosen ciphertext attack (CCA).

In our security reduction of Theorem 1, we restrict the adversary $\mathcal{A}$ from making any query to the decryption oracle. An IBE scheme secure against such an adversary is said to be secure against chosen plaintext attack (CPA). $\mathsf{Adv}^{\mathsf{IBE}}(t, q)$ in this context denotes the maximum advantage where the maximum is taken over all adversaries running in time at most t and making at most q queries to the key-extraction oracle.

A.4 Hardness Assumption

We define the security of our identity based encryption scheme in terms of the *decision bilinear Diffie-Hellman* problem (DBDH). The DBDH problem [7] in G_1 is as follows: given a tuple $\langle P, aP, bP, cP, Z\rangle$, where $Z \in G_2$, decide whether $Z = e(P, P)^{abc}$ which we denote as Z $\mathsf{is\ real}$ or Z is random. The advantage of a probabilistic algorithm $\mathcal{B}$, which takes as input a tuple $\langle P, aP, bP, cP, Z\rangle$ and outputs a bit, in solving the DBDH problem is defined as

$$\begin{aligned}\mathsf{Adv}_{\mathcal{B}}^{\mathsf{DBDH}} = &|Pr[\mathcal{B}(P, aP, bP, cP, Z) = 1|\mathsf{Z\ is\ real}] \\ &- Pr[\mathcal{B}(P, aP, bP, cP, Z) = 1|\ \mathsf{Z\ is\ random}]|\end{aligned}$$

where the probability is calculated over the random choice of $a, b, c \in Z_p$ as well as the random bits used by $\mathcal{B}$. The quantity $\mathsf{Adv}^{\mathsf{DBDH}}(t)$ denotes the maximum of $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{DBDH}}$ where the maximum is taken over all adversaries running in time at most t.

B Lower Bound of λ

We calculate a lower bound on λ for any set of q queries $\mathsf{v}^{(1)}, \ldots, \mathsf{v}^{(q)}$ and a challenge identity v^* as:

$$\begin{aligned}
\lambda &= Pr[\bigwedge_{i=1}^{q}\Big(K(\mathsf{v}^{(i)}) = 1\Big) \wedge (x + \sum_{j=1}^{\ell} x_j \mathsf{v}_j^* = km)] \\
&= Pr[\bigwedge_{i=1}^{q}\Big(K(\mathsf{v}^{(i)}) = 1\Big)]Pr[(x + \sum_{j=1}^{\ell} x_j \mathsf{v}_j^* = km)|\bigwedge_{i=1}^{q}\Big(K(\mathsf{v}^{(i)}) = 1\Big)] \\
&= (1 - Pr[\bigvee_{i=1}^{q}\Big(K(\mathsf{v}^{(i)}) = 0\Big)]Pr[(x + \sum_{j=1}^{\ell} x_j \mathsf{v}_j^* = km)|\bigwedge_{i=1}^{q}\Big(K(\mathsf{v}^{(i)}) = 1\Big)] \\
&\geq (1 - \sum_{i=1}^{q} Pr[\Big(K(\mathsf{v}^{(i)}) = 0\Big)]Pr[(x + \sum_{j=1}^{\ell} x_j \mathsf{v}_j^* = km)|\bigwedge_{i=1}^{q}\Big(K(\mathsf{v}^{(i)}) = 1\Big)] \\
&= (1 - \frac{q}{m})Pr[(x + \sum_{j=1}^{\ell} x_j \mathsf{v}_j^* = km)|\bigwedge_{i=1}^{q}\Big(K(\mathsf{v}^{(i)}) = 1\Big)] \\
&= \frac{1}{\mu_\ell + 1}(1 - \frac{q}{m})Pr[K(\mathsf{v}^*) = 0|\bigwedge_{i=1}^{q}\Big(K(\mathsf{v}^{(i)}) = 1\Big)] \\
&= \frac{1}{\mu_\ell + 1}(1 - \frac{q}{m})\frac{Pr[K(\mathsf{v}^*) = 0]}{Pr[\bigwedge_{i=1}^{q} K(\mathsf{v}^{(i)} = 1)]}Pr[\bigwedge_{i=1}^{q}\Big(K(\mathsf{v}^{(i)}) = 1\Big)\,|K(\mathsf{v}^*) = 0] \\
&\geq \frac{1}{m(\mu_\ell + 1)}(1 - \frac{q}{m})Pr[\bigwedge_{i=1}^{q}\Big(K(\mathsf{v}^{(i)}) = 1\Big)\,|K(\mathsf{v}^*) = 0] \\
&= \frac{1}{m(\mu_\ell + 1)}(1 - \frac{q}{m})(1 - Pr[\bigvee_{i=1}^{q}\Big(K(\mathsf{v}^{(i)}) = 0\Big)\,|K(\mathsf{v}^*) = 0] \\
&\geq \frac{1}{m(\mu_\ell + 1)}(1 - \frac{q}{m})(1 - \sum_{i=1}^{q} Pr[K(\mathsf{v}^{(i)}) = 0|K(\mathsf{v}^*) = 0] \\
&= \frac{1}{m(\mu_\ell + 1)}(1 - \frac{q}{m})^2 \\
&\geq \frac{1}{m(\mu_\ell + 1)}(1 - 2\frac{q}{m})
\end{aligned}$$

In the above derivation the equality in the last but one step comes from the fact that

$$Pr[K(\mathsf{v}^{(i)}) = 0 | K(\mathsf{v}^*) = 0] = Pr[K(\mathsf{v}^{(i)}) = 0] = 1/m$$

since $K(\mathsf{v}^{(i)}) = 0$ for $1 \leq i \leq q$ and $K(\mathsf{v}^*) = 0$ are mutually independent events.

C Security of the Signature Scheme

Brief sketch: This proof also is a reduction. Suppose $\mathcal{A}$ is a CPA adversary for the signature scheme. Then we construct an algorithm $\mathcal{S}$ for Comutational Diffie-Hellman problem (CDH). $\mathcal{S}$ will take as input a 3-tuple $\langle P, aP, bP \rangle$ where P is a generator of G_1 and $aP, bP \in G_1$. We define the following game between $\mathcal{S}$ and $\mathcal{A}$.

The Setup and Signature Generation steps of this game is exactly same as the Setup and Phase 1 of Section 5.

Forge: At this stage the adversary $\mathcal{A}$ submits a message $M^* \in Z_N$ and a signature $\sigma^* = (\sigma_1^*, \sigma_2^*)$ with the constraint that it has not asked for the signature of M^* in the Signature Generation phase. $\mathcal{A}$ wins if σ^* is a valid signature on M^*.

If $\mathcal{A}$ is successful in forging the signature, $\mathcal{S}$ first checks whether $F(M^*) \neq 0$ and aborts in that situation. Otherwise, $\mathcal{S}$ first computes $J(M^*)\sigma_2^*$ and then adds the inverse of this product with σ_1^*. It returns the end result as the value of abP.

Since $F(M^*) = 0$, then as in the Challenge part of proof of Theorem 1, we have

$$J(M^*)\sigma_2^* = rV.$$

Note that, this condition is satisfied as long as $F(M^*) \equiv 0 \bmod p$, which holds if $x + \sum_{j=1}^{\ell} x_j m_j^* = km$.

Now, $\sigma_1^* = abP + rV$ and hence $abP = \sigma_1^* - rV$.

Note that, the conditions under which $\mathcal{S}$ aborts this game is exacly the same under which $\mathcal{S}$ aborts the game in Theorem 1. So the lower bound on the probability of not aborting remains exactly the same.

Yet Another Forward Secure Signature from Bilinear Pairings

Duc-Liem Vo and Kwangjo Kim

International Research center for Information Security (IRIS),
Information and Communications University (ICU),
119 Munji-ro, Yuseong-gu, Daejeon, 305-714, Korea
{vdliem, kkj}@icu.ac.kr

Abstract. In this work, we have proposed yet another forward secure signature based on bilinear pairings. Our forward secure signature requires the general security parameters only independent to the total number of time periods. The scheme can perform key evolving for unlimited time periods while maintaining sizes of keys and signature fixed. In addition, the signing algorithm is very efficient with the simple verification algorithm. We also provide a formal definition along with a detailed security proof of our signature scheme under the assumption of Computational Diffie-Hellman problem.

Keywords: Forward security, pairings, key exposure, key evolution.

1 Introduction

Key Exposure Problem and Forward Secure Signatures. Digital signatures are very essential components in cryptography as well as in various applications nowadays. Using a digital signature, a signer can prove whether an electronic document is produced by him/her or not. Clearly, the signing ability must be restricted to the authorized people (signers), and the signing keys (or private keys) should be kept secret. Compromise of the private keys will cause severe damage to the applications using the digital signatures. In such cases, we cannot believe in any signature that will be produced in the future by the compromised key. Moreover, the signatures generated by that key in the past are suspicious too. We can imagine how serious this kind of damage can bring to the systems that rely on digital signatures such as banking systems, certificate authorities (CAs), and so on. Once a bank's private key is compromised, no one can be sure that given money (digitally signed by the bank) is true or forged one, hence for the safety reason, the bank may revoke those signatures issued previously, making some people lose their money. Obviously, reissuing all past signatures also is a burden to the bank. The similar devastation can be occurred if the CA's root private key is compromised. All certificates issued by this CA become unverifiable until new certificates are reissued with a new root key, and clients are updated this new key. This situation is more serious since the damage

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 441–455, 2006.
© Springer-Verlag Berlin Heidelberg 2006

may happen widely over the Internet and it may take an unexpected time to recover from the damage completely.

To deal with the key exposure problem, there are several solutions. The first one may think is to use key revocation mechanism by certificate revocation. This method can prevent the forgery of the signatures in the future (*i.e.*, after key is compromised), however, it cannot protect for the past signatures. Time-stamp service, introduced in [14], is another way to certify the creation date of a document. Anderson [3] suggested a new method for constructing signature scheme in which the private key is updated periodically while the public key is kept unchanged. With this approach, the compromise of a private key in a certain period does not affect to the past signatures signed by the previously updated private key. From the Anderson's proposal, there are many researches on this type of signature schemes including its formalization and applications.

In [5], Bellare and Miner formalized Anderson's idea by the concepts of the forward secure signature scheme and presented their construction of this signature using a key evolution scheme. This construction is based on a binary tree model in which the total life time of the scheme is divided into a small period. The total periods equal to the number of a binary tree's leaves. In each period, a different private key is used for signing messages and deriving a new private key for the next time period. The public key remains unchanged for all time periods. An adversary, who has the private key in a time period, can produce the next time period's private key but has no way to forge any signature of the previous time periods. Therefore, the signature scheme is forward secure.

From this work, a variety of the forward secure signature schemes has been proposed with many improvements. A new forward signature scheme with shorter keys and more practical was suggested by Abdalla and Reyzin [1]. Krawczyk [19] suggested further improvement by presenting a method to construct a forward secure signature scheme from any signature scheme, such as RSA or DSA signature scheme. Itkis and Reyzin [16] proposed another forward secure signature based on Guillou-Quisquater's signature scheme. Although this scheme is efficient in signing and verifying, it increases the size of key and signature. Malkin *et al.* [20] proposed a new construction of a forward secure signature scheme based on a product and sum composition method. Utilizing this method, one can construct a new forward signature scheme with more time periods from any two forward digital signature schemes. The combining of the sum and product compositions leads to a *MMM* [20] tree construction of a forward secure signature scheme. The advantage of *MMM* tree construction is that the maximal number of time periods needs not to be fixed in advance. F. Hu *et al.* [15] suggested a new forward secure signature scheme from bilinear maps influenced by the forward secure public key encryption schemes [9, 11]; Hu's construction was based on the scheme in [13]. The scheme achieved the small size of key and signature due to the property of an elliptic curve.

Along with these contributions, many valuable researches on other aspects of the forward secure signature have been carried out. Kozlov and Reyzin [18] proposed a forward signature scheme with fast key update; Song [21] suggested

a forward secure group signature scheme; Duc *et al.* [10] proposed a forward secure blind signature scheme based on the strong RSA problem. The researches on threshold signature scheme with forward secrecy are presented in [1] and [22]. Recently, key insulated and intrusion resilience mechanisms are also explored to provide high level of security. However, these mechanisms require interaction between the devices and the server for each time period. In some cases, these methods are not suitable.

Our Contribution. Clearly, the forward secure signature provides stronger security than the traditional signature. Nevertheless, the forward secure signature scheme requires additional computation as well as storage for the key updating processes. In addition, some schemes are limited in the number of time periods. When all time periods are over, the key generation process is invoked to create new keys and new time periods. The new public key needs to be republished too. To overcome this limitation, we have proposed a new forward secure signature scheme which has unlimited time periods using bilinear pairings. In fact, the proposed scheme does not utilize the total number of time periods as an input parameter. The proposed scheme also exhibits good properties: the private key has a fixed size through the key update process, and the public key is kept unchanged. Furthermore, since our signature scheme is based on bilinear pairings, which can be constructed from Weil or Tate pairings on an elliptic curve, the scheme achieves efficiency in terms of the size in key and signature.

Organization. We first introduced the background about key exposure problem and forward security concept in the digital signature paradigm. In Section 2, we will provide the mathematical treatment in bilinear pairings and definitions about the forward secure signature scheme. The details of our signature scheme are presented in Section 3 and its security analysis is discussed in Section 4. The complexity comparative with other schemes is discussed in Section 5. Finally, the conclusion and suggestion for future work are given.

2 Backgrounds

Like almost other forward secure signature schemes, the definitions of our forward secure signature scheme follow the formal definition from [1, 5].

2.1 Definitions

Forward Secure Signature Scheme. The basic idea of the forward secure signature scheme is to use the key evolution technique to update the private key periodically while keeping the public key unchanged. To do so, one can divide the lifetime of the signature scheme into a small period in which a different private key is used to sign messages. The Key Generation algorithm will initialize the lifetime of the signature scheme by creating the first period key pairs. However, the public key has to be fixed for whole lifetime of the signature scheme for convenience as well as keeping the Verification algorithm simple. The Signing

algorithm has to indicate time period in which the private key is used to produce signatures. In the forward secure signature scheme, there is an additional algorithm used to update private keys of the scheme. The Key Update algorithm takes the current private key as input and generates a new private key for the next period. Of course, after generating a new private key, the old private key must be erased immediately. Forward security is ensured by the fact that the Key Update algorithm is a kind of one-way functions, therefore given the current private key, it is hard to compute any previously used private key. The detailed definition of the forward secure signature scheme is given below:

Definition 1 (Key-evolving Signature Scheme). *A key-evolving digital signature scheme is a quadruple of algorithms,* $\mathsf{FSIG} = (\mathsf{FSIG.KeyGen}; \mathsf{FSIG.KeyUp}; \mathsf{FSIG.Sign}; \mathsf{FSIG.Verify})$*, where:*

- $\mathsf{FSIG.KeyGen}$*, the Key Generation algorithm, is a probabilistic algorithm which takes as input a security parameter* $k \in N$ *(given in unary as* 1^k*) and returns a pair* $(SK_0; PK)$*, the initial secret key and the public key;*
- $\mathsf{FSIG.Sign}$*, the (possibly probabilistic) Signing algorithm, takes as input the secret key* SK_i *of the current time period* i *and a message* M*, and returns a pair* $\langle i, \sigma \rangle$*, the signature of* M *for time period* i*;*
- $\mathsf{FSIG.KeyUp}$*, the (possibly probabilistic) Secret Key Update algorithm, takes the secret key for the current period* SK_i *as input and returns the new secret key* SK_{i+1} *for the next time period;*
- $\mathsf{FSIG.Verify}$*, the (deterministic) Verification algorithm, takes the public key* PK*, a message* M*, and a candidate signature* $\langle i, \sigma \rangle$ *as input, and returns* 1 *if* σ *is a valid signature of* M *or 0, otherwise. It is required that* $\mathsf{FSIG.Verify}_{PK}(M; \mathsf{FSIG.Sign}_{SK_i}(M)) = 1$ *for every message* M *and time period* i*.*

Security Analysis Using Random Oracle Model. We analyze our signature scheme in the random oracle model [6]. The security of the signature scheme means that it is computational infeasible for any adversary to forge a signature with respect to any of the previously used secret keys even if the exposure of the current secret key happens. We use the security model introduced by Bellare and Miner [5] with some modification.

In our model, besides knowing the user's public key PK, the adversary also gets to know the current time period. The adversary runs in three phases. In the first phase, the chosen message attack phase (cma), the adversary has access to a signing oracle, which it can query to obtain signatures of messages of its choice with respect to the current secret key. At the end of each time period, the adversary can choose whether to stay in the same phase or switch to the break-in phase ($\mathsf{breakin}$). In the break-in phase, which models the possibility of a key exposure, we give the adversary the secret key SK_j for the specific time period j it decided to break in. In the last phase, the forgery phase (forge), the adversary outputs a pair signature message, that is, a forgery. The adversary is considered to be successful if it forges a signature of some new message (that is, not previously queried to the signing oracle) for some time period prior to j. In

order to capture the notion of forward security of a key-evolving signature scheme $\mathsf{FSIG} = (\mathsf{FSIG.KeyGen}; \mathsf{FSIG.KeyUp}; \mathsf{FSIG.Sign}; \mathsf{FSIG.Verify})$ more formally, let $\mathcal{F}$ be an adversary for this scheme. To assess the success probability of $\mathcal{F}$ breaking the forward security of FSIG, consider the following experiment. Throughout this paper, $k, \ldots$ indicates that the arguments of the key generation algorithm could be more than k.

Experiment F-Forge-RO(FSIG, $\mathcal{F}$)
  `Select` $H : \{0,1\}^* \rightarrow \{0,1\}^l$ `at random`
  $(SK_0, PK) \stackrel{R}{\leftarrow} \mathsf{FSIG.KeyGen}^H(k, \ldots)$
  $i \leftarrow 0$
  `Repeat`
    $d \leftarrow \mathcal{F}^{H, \mathsf{FSIG.Sign}^H_{SK_i}(\bullet)}(\mathsf{cma}, PK)$;
    $SK_{i+1} \leftarrow \mathsf{FSIG.KeyUp}^H(SK_i); i \leftarrow i+1$
  `Until` $(d = \mathsf{breakin})$
  $i \leftarrow i-1$
  $(M, \langle b, \sigma \rangle) \leftarrow \mathcal{F}^H(\mathsf{forge}, SK_i)$
  `If` $\mathsf{FSIG.Verify}^H_{SK_i}(M, \langle b, \sigma \rangle) = 1$ `and` $0 \leq b \leq i-1$
    `and` M `was not queried of` $\mathsf{FSIG.Sign}^H_{SK_i}$ `in period` b
    `then return 1 else return 0`

With the above forger, we can define the notion of security of the forward secure signature scheme in the random oracle model.

Definition 2 (Forward-security in the Random Oracle Model). *Let* $\mathsf{FSIG} = (\mathsf{FSIG.KeyGen}; \mathsf{FSIG.KeyUp}; \mathsf{FSIG.Sign}; \mathsf{FSIG.Verify})$ *be a key-evolving signature scheme, H be a random oracle and $\mathcal{F}$ be an adversary as described above. We let* $\mathbf{Succ}^{\mathrm{fwsig}}(\mathsf{FSIG}[k, \ldots]; \mathcal{F})$ *denote the probability that the experiment* F-Forge-RO($\mathsf{FSIG}[k, \ldots]; \mathcal{F}$) *returns* 1. *Then the insecurity of* FSIG *is the function*

$$\mathbf{InSec}^{\mathrm{fwsig}}(\mathsf{FSIG}[k, \ldots]; t; q_{\mathrm{sig}}; q_{\mathrm{hash}}) = \max_{\mathcal{F}} \{\mathbf{Succ}^{\mathrm{fwsig}}(\mathsf{FSIG}[k, \ldots]; \mathcal{F})\},$$

where the maximum here is taken over all adversaries $\mathcal{F}$ making a total of at most q_{sig} queries to the signing oracles across all the stages and for which the running time of the above experiment is at most t and at most q_{hash} queries are made to the random oracle H.

2.2 Bilinear Pairings

We summarize some concepts of bilinear pairings using similar notations used by Zhang and Kim [23] which was used to design ID-based blind signature and ring signature based on pairings.

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be additive and multiplicative groups of the same prime order q, respectively. Let P is a generator of $\mathbb{G}_1$. Assume that the discrete logarithm problems in both $\mathbb{G}_1$ and $\mathbb{G}_2$ are hard. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a pairing which satisfies the following properties:

1. *Bilinear*: $e(aP, bP') = e(P, P')^{ab}$ for all $P, P' \in \mathbb{G}_1$ and all $a, b \in \mathbb{Z}$.
2. *Non-degenerate*: If $e(P, P') = 1 \; \forall \, P' \in \mathbb{G}_1$ then $P = \mathcal{O}$.
3. *Computable*: There is an efficient algorithm such as [4] to compute $e(P, P')$ for any $P, P' \in \mathbb{G}_1$.

To construct the bilinear pairing, we can use the Weil pairing or Tate pairing associated with supersingular elliptic curves. Under such group $\mathbb{G}_1$, we can define the following hard cryptographic problems:

- *Discrete Logarithm (DL) Problem:* Given $P, P' \in \mathbb{G}_1$, find an integer n such that $P = nP'$ whenever such integer exists.
- *Computational Diffie-Hellman (CDH) Problem:* Given a triple $(P, aP, bP) \in \mathbb{G}_1$ for $a, b \in \mathbb{Z}_q^*$, find the element abP.
- *Decision Diffie-Hellman (DDH) Problem:* Given a quadruple $(P, aP, bP, cP) \in \mathbb{G}_1$ for $a, b, c \in \mathbb{Z}_q^*$, decide whether $c = ab \pmod q$ or not.

A group, where the CDH problem is hard but the DDH problem is easy, is called Gap Diffie-Hellman (*GDH*) group. Details about GDH groups can be found in [7], [8], and [17].

For the sake of comparison, we assume that, as in [15], there is a parameter generator $\mathcal{IG}$ takes input k, and outputs $\mathbb{G}_1, \mathbb{G}_2$ of order q, and pairing e. The computational complexity of $\mathcal{IG}$ is $O(k^n)$. Also the computational complexity in groups $\mathbb{G}_1, \mathbb{G}_2$, and pairings e are at most $O(k^{n_1})$, $O(k^{n_2})$, and $O(k^e)$, respectively. We have $n, n_1, n_2, e \in N$ are order of the polynomial time algorithm.

The definition of the CDH assumption which is used for our security analysis as follows:

Definition 3 (CDH Assumption). *A probabilistic algorithm $\mathcal{A}$ is said to be (t, ϵ)-break-CDH in a cyclic group $\mathbb{G}$ if $\mathcal{A}$ runs at most time t, computes the Diffie-Hellman function $DH_{P,q}(aP, bP) = abP$, with input (P, q) and (aP, bP), with a probability of at least ϵ, where the probability is over the coins of $\mathcal{A}$ and (a, b) is chosen uniformly from $\mathbb{Z}_q \times \mathbb{Z}_q$. The group $\mathbb{G}$ is a (t, ϵ)-CDH group if no algorithm (t, ϵ)-break-CDH in this group.*

3 Our Scheme

As mentioned previously, our forward secure signature scheme FSIG consists of four algorithms. Our purpose in designing this scheme need not to define the *total number of time periods* in advance, hence we can have *unlimited* time periods forward signature scheme. In other words, the key evolution process can run forever.

Key Generation Algorithm. The Key Generation algorithm takes a secure parameter k and returns the initial key pair $(SK_0; PK)$. Our Key Generation algorithm uses the same strategy like [15] in order to generate system parameters: groups $\mathbb{G}_1, \mathbb{G}_2$ of the same prime order q; a generator P of $\mathbb{G}_1$; a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Let H_1 be a collision-free hash function which converts an

arbitrary string $\{0,1\}^*$ into $\mathbb{Z}_q^*$. We also assume that there is a collision-free hash function $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$. This hash function can be considered as a part of the Key Generation algorithm.

FSIG.KeyGen(1^k)

Run $\mathcal{IG}$ to get groups $\mathbb{G}_1, \mathbb{G}_2$ (prime order q), bilinear map e.
Select random generator $P \leftarrow \mathbb{G}_1$; $s, t, r_0 \overset{R}{\leftarrow} \mathbb{Z}_q^*$.
Compute: $Q = sP, T = tP$
Set $PK \leftarrow (\mathbb{G}_1, \mathbb{G}_2, e, P, Q, T)$
Compute:
$s_0 = s + r_0 H_1(0); t_0 = t - r_0 H_1(0)$
$Q_0 = r_0 H_1(0) P$;
$V_0 = t_0 Q_0$;
Erase s, t, r_0, t_0
Set $SK_0 \leftarrow (s_0, V_0, Q_0)$

Output $(SK_0; PK)$

Key Update Algorithm. The Key Update algorithm is the core part of the key evolution scheme. It refreshes the private key of the current time period to the new value corresponding to the new time period, then erases the previous private key. The Key Update algorithm is given below:

FSIG.KeyUp(i, SK_{i-1})

Pick a random element $r_i \in \mathbb{Z}_q^*$;
Parse SK_{i-1} as $(s_{i-1}, V_{i-1}, Q_{i-1})$
Compute:
$s_i = s_{i-1} + r_i H_1(i)$;
$Q_i = Q_{i-1} + r_i H_1(i) P$;
$V_i = V_{i-1} + r_i H_1(i)\,(T - Q_{i-1} - Q_i)$;
Erase $s_{i-1}, r_i, V_{i-1}, Q_{i-1}$
Set $SK_i \leftarrow (s_i, V_i, Q_i)$

Output SK_i

Signing Algorithm. The Signing algorithm produces a signature at the current time period using the private key of the considered time period.

FSIG.Sign(i, M, SK_i):

Parse SK_i as (s_i, V_i, Q_i)
Set $U = Q_i$;
Compute $\alpha = s_i Q_i + V_i$; and $\beta = s_i H_2(i, M, U)$;
Set $\sigma = (U, \alpha, \beta)$

Output signature for M as $\langle i, \sigma \rangle$

Verification Algorithm. The Verification algorithm tests if a given signature on a message at a specific time period is valid or not. The output of the test is 1 if the signature is valid and 0 otherwise.

FSIG.Verify$_{PK}(i, M, \sigma)$:

Parse σ as (U, α, β) Verify

$$e(\alpha, P) \stackrel{?}{=} e(U, T + Q) \tag{1}$$

$$e(\beta, P) \stackrel{?}{=} e(H_2(i, M, U), U + Q) \tag{2}$$

`Output 1 if Eqs (1) and (2) are correct, otherwise output 0.`

The first equation (1) verifies the value of U and the second one (2) verifies the signature on the message M. The time period value is embedded in the signature too.

Correctness. The correctness of the proposed signature scheme comes from the correctness of Eqs (1) and (2). The correctness of Eq (1) is shown bellow:

$$\begin{aligned}
&e(\alpha, P) = e\left(s_i Q_i + V_i, P\right) = e\left(\left[s_{i-1} + r_i H_1(i)\right] Q_i + V_i, P\right) \\
&= e\left(\left[s + \sum_{k=0}^{i} r_k H_1(k)\right] Q_i + V_i, P\right) = e\left(sQ_i + \sum_{k=0}^{i} r_k H_1(k) Q_i + V_i, P\right) \\
&= e\left(sQ_i + \sum_{k=0}^{i} r_k H_1(k) Q_i + V_{i-1} + r_i H_1(i)\left[T - Q_{i-1} - Q_i\right], P\right) \\
&= e\left(sQ_i + \sum_{k=0}^{} 0^{i-1} r_k H_1(k) Q_i + V_{i-1} + r_i H_1(i)\left[T - Q_{i-1}\right], P\right) \\
&= e\left(sQ_i + \sum_{k=0}^{i-1} r_k H_1(k)\left[Q_{i-1} + r_i H_1(i) P\right] + V_{i-1} + r_i H_1(i)\left[T - Q_{i-1}\right], P\right) \\
&= e\left(sQ_i + \sum_{k=0}^{i-1} r_k H_1(k) Q_{i-1} + r_i H_1(i) \sum_{k=0}^{i-1} r_k H_1(k) P + V_{i-1} + r_i H_1(i)\left[T - Q_{i-1}\right], P\right) \\
&= e\left(sQ_i + \sum_{k=0}^{i-1} r_k H_1(k) Q_{i-1} + V_{i-1} + r_i H_1(i) T, P\right) \\
&\vdots \\
&= e\left(sQ_i + r_0 H_1(0) Q_0 + V_0 + r_1 H_1(1) T + ... + r_i H_1(i) T, P\right) \\
&= e\left(sQ_i + r_0 H_1(0) Q_0 + t_0 Q_0 + r_1 H_1(1) T + ... + r_i H_1(i) T, P\right) \\
&= e\left(sQ_i + tQ_0 + r_1 H_1(1) T + ... + r_i H_1(i) T, P\right) \\
&= e\left(sQ_i + r_0 H_1(0) T + r_1 H_1(1) T + ... + r_i H_1(i) T, P\right) \\
&= e\left(sQ_i + \sum_{k=0}^{i} r_k H_1(k) T, P\right) = e\left(sQ_i + t \sum_{k=0}^{i} r_k H_1(k) P, P\right) \\
&= e\left(\left[s + t\right] Q_i, P\right) = e\left(Q_i, P\right)^{s+t} = e\left(Q_i, T + Q\right)
\end{aligned}$$

The correctness of this equation ensures that the value of U is correct. The validity of the signature is guaranteed by the correctness of Eq (2).

$$\begin{aligned}
&e(\beta, P) = e(s_i H_2(i, M, U), P) = e\left(\left[s_{i-1} + r_i H_1(i)\right] H_2(i, M, U), P\right) \\
&\vdots \\
&= e\left(\left[s + \sum_{k=0}^{i} r_k H_1(k)\right] H_2(i, M, U), P\right) \\
&= e\left(sH_2(i, M, U), P\right) e\left(H_2(i, M, U), P\right)^{\sum_{k=0}^{i} r_k H_1(k)} \\
&= e\left(H_2(i, M, U), Q\right) e\left(H_2(i, M, U), \sum_{k=0}^{i} H_1(k) r_k P\right) \\
&= e\left(H_2(i, M, U), Q\right) e\left(H_2(i, M, U), Q_i\right) \\
&= e\left(H_2(i, M, U), Q\right) e\left(H_2(i, M, U), U\right) = e\left(H_2(i, M, U), Q + U\right)
\end{aligned}$$

Efficiency. Our proposed forward secure signature scheme exhibits new properties. Firstly, our scheme does not have the total number of time periods parameter. The Key Update algorithm can perform infinite and stop only when a private key in a certain time period is compromised. At that time, we need to run the Key Generation algorithm again to initialize a new key pair.

Secondly, the key pair produced by the Key Generation algorithm has a fixed length. The sizes of the private key and public key do not grow after running the Key Update algorithm. In addition, the public key remains unchanged since after produced once at the first time by the Key Generation algorithm.

The Signing and Verifying algorithms also require the fixed amount of computational time. The signing algorithm of our scheme is very unique. For a certain time period, one can compute value of α once, then stores it for future signature issuing. From the next time, the signature issuing algorithm is just to compute one point multiplication over an elliptic curve.

4 Security Analysis

We analyze the security of our forward secure signature scheme used technique like in [1, 5, 15]. In addition, we assume that, partial key exposure also leads to key exposure problem. This is obviously since we may derive the remained part from exposed part of the private key. The following theorem shows the security of our scheme.

Theorem 1. *If there exists a forger $\mathcal{F}$ that runs in time at most t, asking at most q_{hash} hash queries and q_{sig} signing queries, such that* $\mathbf{Succ}^{\text{fwsig}}(\mathsf{FSIG}[k,\ldots]; \mathcal{F}) > \epsilon$ *then there exists a adversary $\mathcal{A}$ that (t', ϵ')-break CDH in group $\mathbb{G}_1$ where:*

$$t' = t + O\left(k^{n_1}\right);\ \textit{and}\ \epsilon' = \left(1 - \frac{1}{q_{\text{sig}} + 1}\right)^{q_{\text{sig}}+1} \cdot \frac{1}{q_{\text{sig}}\left(q_{\text{hash}} + q_{\text{sig}} + 1\right)} \cdot \epsilon$$

Proof (Sketch). To break CDH problem in the additive group $\mathbb{G}_1$ of the order q, an adversary $\mathcal{A}$ is given P (a random generator of $\mathbb{G}_1$), $P' = aP$, $Q' = bP$, where $a, b \in \mathbb{Z}_q^*$ are randomly chosen and remain unknown to $\mathcal{A}$. The task of $\mathcal{A}$ is to derive $S' = abP$ with the help of the forger $\mathcal{F}$. $\mathcal{A}$ provides the public key to $\mathcal{F}$ and answers its hash queries, signing queries, and breakin query. First, $\mathcal{A}$ guesses a random i at which $\mathcal{F}$ will ask for the breakin query. Then $\mathcal{A}$ set the public key $PK = (\mathbb{G}_1, \mathbb{G}_2, e, P, Q, T)$, where $Q = Q'$. $\mathcal{A}$ provides PK to $\mathcal{F}$ and runs it. $\mathcal{A}$ can answer the hash queries and the signing queries since it controls the hash oracle. During execution, $\mathcal{A}$ guesses a random index g', and hopes the forgery will base on g'-th hash query. $\mathcal{A}$ makes this hash value special, *i.e.*, P'. Suppose $\mathcal{F}$ outputs a signature on message $M_{g'}$ for time period $i' < i$. From this signature $\mathcal{A}$ can derive $S' = abP$, hence solves CDH problem. The detailed proof is given in Appendix.

Theorem 2. *Let* $\mathsf{FSIG}[k;\ldots]$ *represent our key-evolving signature scheme with modulus size k. Then for any t, q_{hash} and q_{sig},*

$$\mathbf{InSec}^{\mathrm{fwsig}}(\mathsf{FSIG}[k;\ldots];t;q_{\mathrm{sig}};q_{\mathrm{hash}}) \leq$$

$$q_{\mathrm{sig}}(q_{\mathrm{hash}}+q_{\mathrm{sig}}+1)\left(1-\frac{1}{q_{\mathrm{sig}}+1}\right)^{-(q_{\mathrm{sig}}+1)}\mathbf{InSec}^{\mathrm{cdh}}(k,t')$$

where $t' = t + O(k^{n_1})$

Proof. From Definition 2 and Theorem 1, the insecurity function is computed simply by solving function in Theorem 1 and express ϵ' in terms of ϵ we have:

$$\epsilon' = \left(1-\frac{1}{q_{\mathrm{sig}}+1}\right)^{q_{\mathrm{sig}+1}} \cdot \frac{1}{q_{\mathrm{sig}}\,(q_{\mathrm{hash}}+q_{\mathrm{sig}}+1)} \cdot \epsilon$$

$$\Longrightarrow q_{\mathrm{sig}}\,(q_{\mathrm{hash}}+q_{\mathrm{sig}}+1)\epsilon' \Big/ \left(1-\frac{1}{q_{\mathrm{sig}}+1}\right)^{q_{\mathrm{sig}+1}} = \epsilon$$

This completes the proof of Theorem 2. □

5 Evaluation

In this section, we compare our proposed signature scheme with the previous signature scheme [15] which has the same computational assumption. Computational complexity, the sizes of keys and signature are examined.

Table 1. Computational complexity of our signature scheme

Algorithm	Ours	Hu *et al.*[15]
Key generation	$O\,(k^n + k^{n_1})$	$O\,(k^n + k^{n_1} + k^{n_1}\log T)$
Signing	$O\,(k^{n_1})$	$O\,(k^{n_1})$
Verification	$O\,(k^e + k^{n_1})$	$O\,(k^e \log T + k^{n_1}\log T)$
Key Update	$O\,(k^{n_1})$	$O\,(k^{n_1})$
Public key size	$O\,(k)$	$O(k)$
Private key size	$O\,(k)$	$O\,(k\log T + k)$
Signature size	$O\,(k)$	$O\,(k\log T + k)$

Table 2. Computational complexity of other schemes

Algorithm	BM[5]	AR[1]	IR[16]	*MMM*[20]
Key generation	lk^2T	lk^2T	$k^5 + (k+l^3)lT$	k^2l^2
Signing	$(T+l)k^2$	lk^2T	k^2l	k^2l
Verification	$(T+l)k^2$	k^2l	k^2l	$k^2l + l^2\log l$
Key Update	lk^2	lk^2	$(k^2+l^3)lT$	$k^2l + (k+l^2)\log t$
Public key size	lk	k	k	$k + l\log l$
Private key size	lk	k	k	l
Signature size	k	k	k	$k + l\log l$

In Table 1, T is total number of time periods and in Table 2, l is a security parameter of conventional cryptographic operation as explained in [20].

As we can see from Tables 1 and 2, our signature scheme is very efficient in terms of computation as well as performance. The signature and key sizes do not depend on the total number of time periods. Moreover, comparing with the schemes [1, 5, 16], the signature size is shorter for the same security level since our scheme is operating over an elliptic curve (so in the security parameter k is different). The signing algorithm will be similar to that of [8] if we store the fixed part in the signature for later use. Although *MMM* scheme has unbounded time periods, it still depends on the current time period parameter in the key update algorithm.

The verification of the signature just requires four pairing operations. This can be considered to be the same as that of [8]. For verifying multiple signatures of the same time period, the result of verification equation (1) can be saved for later use. In this case, the verifying process remains just two pairing computations. Although pairing computation is expensive, there are many improvements in implementation of the pairings as in [4, 12]. Utilizing those good implementations, our scheme can be efficient in performance. Considering above features, our signature scheme can be applied in the application where storage and computation power are limited like mobile devices. Forward secure properties will strengthen the security of the applications.

6 Concluding Remarks

We have proposed yet another forward signature scheme using bilinear pairings. Our signature scheme has a specific property, namely unlimited time periods. Under the assumption of the hardness of Computational Diffie-Hellman problem, we have presented the security proof of the signature scheme in the random oracle model. Moreover, the proposed signature scheme is very efficient in terms the signature size as well as performance compared to the previous schemes. The scheme's public and private key sizes are unchanged through the key evolving processes. With a good pairing computation algorithm, we can have an efficient signature verifying algorithm.

For further work, we consider integration of our scheme with other cryptographic techniques to have new applications.

Acknowledgements

The authors would like to express great thank to the anonymous reviewers for useful comments.

This work was partially supported by a grant No.R12-2003-004-01004-0 from Ministry of Commerce, Industry and Energy.

References

1. M. Abdalla and L. Reyzin, "A New Forward-Secure Digital Signature Scheme," *Advances in Cryptology* – ASIACRYPT 2000, LNCS 1976, pp. 116–129, Springer-Verlag, Dec. 2000.
2. M. Abdalla, S. Miner, and C. Namprempre, "Forward-Secure Threshold Signature Schemes," *Topics in Cryptology* – CT-RSA 2001, LNCS 2020, pp. 441–456, Springer-Verlag, 2001.
3. R. Anderson. "Two Remarks on Public-Key Cryptology From Invited Lecture," *Fourth ACM onference on Computer and Communications Security*, April, 1997. http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-549.pdf
4. P.S.L.M. Barreto, H.Y. Kim, B. Lynn and M. Scott, "Efficient Algorithms for Pairing-Based Cryptosystems", *Advances in Cryptology – Crypto'2002*, LNCS 2442, pp. 354-369, Springer-Verlag, 2002.
5. M. Bellare and S. K. Miner, "A Forward-Secure Digital Signature Scheme," *Advances in Cryptology* – CRYPTO '99, LNCS 1666, pp. 431–448, Springer-Verlag, Aug. 1999.
6. M. Bellare and P. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols", ACM Conference on Computer and Communications Security, pp. 62–73, 1993.
7. D. Boneh and M. Franklin, "ID-based Encryption from the Weil Pairing," *Advances in Cryptology* - Crypto'2001, LNCS 2139, pp. 213–229, Springer-Verlag, 2001.
8. D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing", *Advances in Cryptology – Asiacrypt'2001*, LNCS 2248, pp. 514–532, Springer-Verlag, 2001.
9. R. Canetti, S. Halevi, and J. Katz, "A Forward-Secure Public-Key Encryption Scheme," *Advances in Cryptology* – EUROCRYPT'03, LNCS 2656, pp. 255–271, Springer-Verlag 2003.
10. Dang Nguyen Duc, Jung Hee Cheon, and Kwangjo Kim, "A Forward-Secure Blind Signature Scheme Based on the Strong RSA Assumption," *Information and Communications Security* – ICICS'03, LNCS 2836, pp. 11–21. Springer-Verlag, 2003.
11. Y. Dodis, M. Franklin, J. Katz, A. Miyaji, and M. Yung, "Intrusion Resilient Public-Key Encryption," *Topics in Cryptology* – CT-RSA 2003, LNCS 2612, pp. 19–32, Springer-Verlag, 2003.
12. S. Galbraith, K. Harrison and D. Soldera, "Implementing the Tate Pairing," *Algorithm Number Theory Symposium* - ANTS V, LNCS 2369, pp. 324–337, Springer-Verlag, 2003.
13. C. Gentry and A. Silverberg, "Hierarchical ID-Based Cryptography," *Advances in Cryptology* – ASIACRYPT 2002, LNCS 2501, Y. Zheng ed., pp. 548–566, Springer-Verlag, 2002.
14. S. Haber and W. Stornetta, "How to Time-Stamp a Digital Document", *Advances in Cryptology* – CRYPTO 90, LNCS 537, A. J. Menezes and S. Vanstone, ed., Springer-Verlag, 1990.
15. F. Hu, C. Wu, and J.D. Irwin, "A New Forward Secure Signature Scheme using Bilinear Maps," http://eprint.iacr.org/2003/188.pdf.
16. G. Itkis and L. Reyzin. "Forward-secure signatures with optimal signing and verifying," Advances in Cryptology – CRYPTO'01, LNCS 2139, pp. 332–354. Springer-Verlag, 2001.
17. A. Joux and K. Nguyen, "Separating Decision Diffie-Hellman from Diffie-Hellman in Cryptographic Groups," *Cryptology ePrint Archive – 2001/03.*

18. A. Kozlov and L. Reyzin. "Forward-Secure Signatures with Fast Key Update," Proc. of the 3rd International Conference on Security in Communication Networks – SCN'02, 2002.
19. H. Krawczyk. "Simple Forward-Secure Signatures from Any Signature Scheme," Proc. of Seventh ACM Conference on Computer and Communications Security, pp. 108–115, Nov. 2000.
20. T. Maklin, D. Micciancio and S. Miner, "Efficient Generic Forward-Secure Signatures with an Unbounded Number of Time Periods," *Advances in Cryptology* – EUROCRYPT 2002, LNCS 2332, L. Knudsen ed., pp. 400–417, Springer-Verlag, 2002.
21. D. X. Song. "Practical Forward Secure Group Signature Schemes," Proc. of the 8th ACM Conference on Computer and Communications Security – CCS 2001, pp. 225–234. ACM, 2001.
22. W. Tzeng and Z. Tzeng, "Robust Forward-Secure Digital Signature with Proactive Security," *Public Key Cryptography* – PKC'01, LNCS 1992, K. Kim ed., pp. 264–276, Springer-Verlag, 2001.
23. F. Zhang and K. Kim, "ID-Based Blind Signature and Ring Signature from Pairings", *Advances in Cryptology – Asiacrypt'2002*, LNCS 2501, Springer-Verlag, pp. 533–547, 2002.

A Proof of Theorem 1

Proof. Using the same technique in [15], we describe the details procedure of $\mathcal{A}$ as follows. As in [1], first we assume that if $\mathcal{F}$ outputs a forgery of $\langle i, \sigma \rangle$ for message M, then the hash oracle has been queried on (i, M, Q_i). Any adversary can be modified to do that. Because of this, the number of hash queries may increase to $q_{\text{hash}} + 1$. We also assume that if $\mathcal{F}$ asks for the signing query for some message M in some time period i, then the hash query on (i, M, Q_i) must also be request simultaneously. Any adversary can be modified to do so and therefore, the number of hash queries may increase to $q_{\text{hash}} + q_{\text{sig}} + 1$. Assume that $\mathcal{F}$ maintains all necessary bookkeeping and does not ask for the same hash query twice. Note that, the number of hash queries on the hash oracle H_1 can be included in q_{hash} without any effect since it only happens in Key Update procedure.

First of all, $\mathcal{A}$ has to guess the time period i at which $\mathcal{F}$ will ask for the breakin query. It randomly selects $i > 0$, hoping that the breakin query will occur at this time period. $\mathcal{A}$ then generates the public key $PK \leftarrow (\mathbb{G}_1, \mathbb{G}_2, e, P, Q, T)$ but $\mathcal{A}$ sets $Q = Q' = bP$ directly. $\mathcal{A}$ also randomly picks $k \in \mathbb{Z}_q^*$ then sets $T = kP - Q$. At this moment, $s = b$ is unknown to both $\mathcal{A}$ and $\mathcal{F}$. $\mathcal{A}$ gives the public key to forger $\mathcal{F}$, and runs until there is a breakin query.

To answer the hash query and the signing query of $\mathcal{F}$, $\mathcal{A}$ maintains two tables: a signature query table and a hash query table.

Signing queries are answered at random since $\mathcal{A}$ controls the hash oracle. In order to answer a signature query number n on a message M'_n during time period $j'_n < i$ (breakin happens), $\mathcal{A}$ selects randomly: $x_{j'_n}, y_{j'_n} \in_R \mathbb{Z}_q^*$. $\mathcal{A}$ then selects randomly $s_{j'_n} \in_R \mathbb{Z}_q^*$ and computes $U'_n = y_{j'_n} P$; $\alpha'_n = y_{j'_n}(Q + T)$; $\beta'_n = x_{j'_n}(U'_n + Q)$; $V'_n = \alpha'_n - s_{j'_n} P$, $h'_n = x_{j'_n} P$. $\mathcal{A}$ checks in its signature query table to see if a signature query on M'_n during time period j'_n has already been asked, and if there is, $j'_n, U'_n, \alpha'_n, \beta'_n$ used in answering. If not, $\mathcal{A}$

answers the query as $j'_n, U'_n, \alpha'_n, \beta'_n$ and also records the signature query entry as $\left(n, j'_n, x_{j'_n}, y_{j'_n}, s_{j'_n}, U'_n, \alpha'_n, \beta'_n, V'_n, h'_n, M'_n\right)$. This setting satisfies the Verify algorithm:

$$\begin{aligned} e\left(\alpha'_n, P\right) &= e\left(y_{j'_n}(Q+T), P\right) = e\left(y_{j'_n}(s+t)P, P\right) \\ &= e\left(y_{j'_n}P, P\right)^{s+t} = e\left(y_{j'_n}P, (s+t)P\right) \\ &= e\left(U'_n, T+Q\right) \\ e\left(\beta'_n, P\right) &= e\left(x_{j'_n}(U'_n+Q), P\right) = e\left(x_{j'_n}(y_{j'_n}+s)P, P\right) \\ &= e\left(x_{j'_n}P, P\right)^{y_{j'_n}+s} = e\left(x_{j'_n}P, (y_{j'_n}+s)P\right) \\ &= e\left(h'_n, U'_n+Q\right) \end{aligned}$$

For a signing query n on a message M'_n during time period $j'_n \geq i$, $\mathcal{A}$ picks randomly $s_{j'_n}, x_{j'_n} \in_R \mathbb{Z}_q^*$ and computes $U'_n = s_{j'_n}P - Q$; $\alpha'_n = kU'_n$; $\beta'_n = s_{j'_n}x_{j'_n}P$; $V'_n = \alpha'_n - s_{j'_n}U'_n$, $h'_n = x_{j'_n}P$. Again, $\mathcal{A}$ also checks in its signature query table to see if a signature query on M'_n during time period j'_n has already been asked, and if there is, $j'_n, U'_n, \alpha'_n, \beta'_n$ used in answering. If not, $\mathcal{A}$ answers the query as $j'_n, U'_n, \alpha'_n, \beta'_n$ and also records the signature query entry as $\left(n, j'_n, x_{j'_n}, y_{j'_n}, s_{j'_n}, U'_n, \alpha'_n, \beta'_n, V'_n, h'_n, M'_n\right)$. The $y_{j'_n}$ can be set to 0 in this case. This setting also satisfies the Verify algorithm:

$$\begin{aligned} e\left(\alpha'_n, P\right) &= e\left(kU'_n, P\right) = e\left(U'_n, kP\right) \\ &= e\left(U'_n, Q+T\right) \\ e\left(\beta'_n, P\right) &= e\left(x_{j'_n}s_{j'_n}P, P\right) = e\left(x_{j'_n}P, s_{j'_n}P\right) \\ &= e\left(h'_n, U'_n+Q\right) \end{aligned}$$

The triple $(s_{j'_n}, U'_n, V'_n)$ also is valid if $\mathcal{F}$ checks it in this breakin phase.

Hash queries are answered at random. To answer the t-th hashing query for (j_t, M_t, U_t), $\mathcal{A}$ first checks the signature query table to see if there is an entry $\left(n, j'_n, x_{j'_n}, y_{j'_n}, s_{j'_n}, U'_n, \alpha'_n, \beta'_n, V'_n, h'_n, M'_n\right)$ such that $(j_t, M_t, U_t) = (j'_n, M'_n, U'_n)$. If so, it just outputs h'_n. Otherwise, $\mathcal{A}$ picks randomly $x_{j_t} \in_R \mathbb{Z}_q^*$, and set the output to $h_t = x_{j_t}P$. It also records value $(t, j_t, x_{j_t}, U_t, h_t, M_t)$. During execution, $\mathcal{A}$ has to guess a random index g'-th, with hope that forgery will happen. $\mathcal{A}$ sets it as special value $P' = aP$.

At the breakin occurs in the time period i, $\mathcal{A}$ simply outputs the secret key SK_i, which is an entry in the signing query table. The validity of SK_i is easy to check. If breakin occurs not in time period i, $\mathcal{A}$ will abort.

Suppose $\mathcal{A}$'s guesses for the time period breakin and the hash index are correct, and $\mathcal{F}$ outputs a forgery $\langle i', \sigma' \rangle$ on a message $M_{g'}$, where $\sigma' = (U', \alpha', \beta')$, and $i' < i$. If the verification holds, $\mathcal{A}$ can derive $S' = abP$ as follows:

We have:

$$\begin{aligned} e(\beta', P) &= e(H_2(i', M_{g'}, U'), U' + Q) \\ &= e(H_2(i', M_{g'}, U'), U')e(H_2(i', M_{g'}, U'), Q) \\ \Rightarrow \frac{e(\beta', P)}{e(H_2(i', M_{g'}, U'), U')} &= e(H_2(i', M_{g'}, U'), Q) \end{aligned}$$

$\mathcal{A}$ controls the hash oracle and the forger $\mathcal{F}$ does not have ability to alter or verify the hash oracle. We may assume that U' equals to the one in time period $i' < i$, in which $\mathcal{A}$ has queried for signatures and $\mathcal{A}$ has value $U' = y'P$ in that time period, otherwise $\mathcal{A}$ fails. Then we have:

$$\begin{aligned}
&\Rightarrow \frac{e(\beta', P)}{e(H_2(i', M_{g'}, U'), y'P)} = e(H_2(i', M, U'), Q) \\
&\Rightarrow \frac{e(\beta', P)}{e(y^{-1}H_2(i', M_{g'}, U'), P)} = e(H_2(i', M, U'), Q) \\
&\Rightarrow \frac{e(\beta', P)}{e(y'^{-1}H_2(i', M_{g'}, U'), P)} = e(H_2(i', M, U'), Q) \\
&\Rightarrow e(\beta' - y'^{-1}P', P) = e(P', Q) = e(aP, bP) = e(abP, P)
\end{aligned}$$

Therefore we can set $S' = abP = \beta' - y^{-1}P'$. We can have this since non-degenerate property of the bilinear pairings.

Run time. Suppose that bit operations in $\mathbb{G}_1$ is at most $O(k^{n_1})$ as in [15], to run $\mathcal{F}$, $\mathcal{A}$ needs to perform some key generation and some group operations. Therefore, we have the time running between $\mathcal{A}$ and $\mathcal{F}$ is different: $t' = t + O(k^{n_1})$

Probability. First, we can see that, $\mathcal{A}$ always acts as a real signer to the $\mathcal{F}$ from $\mathcal{F}$'s point of view except one case when $\mathcal{A}$ answers the hash query with other value. There are totally two guesses performed by $\mathcal{A}$.

The probability that $\mathcal{A}$ guesses the correct time period $\mathcal{F}$ sends the breakin query after q_{sig} signing queries is calculated as follows. Call the event in which breakin occurs E_b and ω is a probability constant which E_b depends on. ω will distribute over $\{0, 1\}$, where 1 is drawn with probability ω and 0 with probability $1-\omega$. We need to calculate the probability of E_b in a certain time period. Suppose that at each time period $1, 2, \ldots i$, the number of signatures has been queried is $q_1, q_2, \ldots q_i$, respectively. If E_b happens at time period i, probability of such event is calculated as $\Pr = (1-\omega)^{q_1}(1-\omega)^{q_2} \ldots (1-\omega)^{q_i}\omega$ Notice that, at the breakin period, there are total q_{sig} queries have been done, so $q_{\text{sig}} = \sum_{k=1}^{i} q_k$. And the probability of guessing break-in at time period i correctly will be $\Pr = (1-\omega)^{q_{\text{sig}}}\omega$. This value is maximized at $\omega = 1/(q_{\text{sig}} + 1)$ and we have

$$\Pr = \left(1 - \frac{1}{q_{\text{sig}} + 1}\right)^{q_{\text{sig}}} \cdot \frac{1}{q_{\text{sig}} + 1} = \frac{1}{q_{\text{sig}}} \cdot \left(1 - \frac{1}{q_{\text{sig}} + 1}\right)^{q_{\text{sig}}+1}$$

The probability to guess the correct hash query on which the forgery is based is $\Pr \geq 1/(1 + q_{\text{sig}} + q_{\text{hash}})$. Therefore the probability of $\mathcal{A}$'s success in deriving $S' = abP$ is at least:

$$\epsilon' = \frac{1}{q_{\text{sig}}} \cdot \left(1 - \frac{1}{q_{\text{sig}} + 1}\right)^{q_{\text{sig}}+1} \cdot \frac{1}{q_{hash} + q_{sig} + 1} \cdot \epsilon$$

where ϵ is the minimum probability with which $\mathcal{F}$ to successfully forge a signature. □

Author Index