Min Surp Rhee
Byoungcheon Lee (Eds.)

Information Security and Cryptology – ICISC 2006

LNCS 4296

9th International Conference
Busan, Korea, November/December 2006
Proceedings

Lecture Notes in Computer Science 4296

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

Min Surp Rhee Byoungcheon Lee (Eds.)

Information Security and Cryptology – ICISC 2006

9th International Conference
Busan, Korea, November 30 - December 1, 2006
Proceedings

Springer

Volume Editors

Min Surp Rhee
Dankook University
San 29, Anseo-dong, Cheonan-shi
Chungnam, 330-714, Korea
E-mail: msrhee@dankook.ac.kr

Byoungcheon Lee
Joongbu University
101 Daehak-Ro, Chubu-Myeon, Guemsan-Gun
Chungnam, 312-702, Korea
E-mail: sultan@joongbu.ac.kr

Library of Congress Control Number: 2006936103

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 3-540-49112-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-49112-5 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: 11927587 06/3142 5 4 3 2 1 0

Preface

ICISC 2006, the Ninth International Conference on Information Security and Cryptology was held in Busan, Korea, during November 30 - December 1, 2006. It was organized by the Korea Institute of Information Security and Cryptology (KIISC) in cooperation with the Ministry of Information and Communication (MIC), Korea. The aim of this conference was to provide a forum for the presentation of new results in research, development, and application in information security and cryptology. It also intended to be a place where research information can be exchanged.

Started in 1998, ICISC has grown into an important international conference in the information security and cryptology area with an established reputation. Based on this maturity, we tried an important change in the publication policy this year. Until last year, pre-proceedings were distributed at the conference and proceedings in Springer's *Lecture Notes in Computer Science* (LNCS) were published after the conference. This year ICISC proceedings were published in LNCS before the conference and distributed to the participants at the conference. We appreciate Springer for their full support and help in making this possible.

The conference received 129 submissions from 17 countries, covering all areas of information security and cryptology. The review and selection processes were carried out in two stages by the Program Committee of 57 prominent researchers via online meetings through the iChair Web server. First, each paper was blind reviewed by at least three PC members, and papers co-authored by the PC members were reviewed by at least five PC members. Second, individual review reports were revealed to PC members, and detailed interactive discussion on each paper followed. Through this process the Program Committee finally selected 26 papers from 12 countries. The authors of selected papers had a few weeks to prepare final versions of their papers, aided by comments from the reviewers. The proceedings contain the revised versions of the accepted papers. However, most of these final revisions were not subject to any further editorial review.

The conference program included two invited talks from eminent researchers in information security and cryptology. Serge Vaudenay from EPFL gave an interesting talk on RFID privacy entitled "RFID Privacy Based on Public-Key Cryptography." Palash Sarkar from the Indian Statistical Institute talked on "Generic Attacks on Symmetric Ciphers," which showed various time-memory trade-off attacks on symmetric cipher algorithms.

We would like to thank everyone who contributed to the success of this conference. First, thanks to all the authors who submitted papers to this conference. Second, thanks to all 57 members of the Program Committee listed overleaf. It was a truly nice experience to work with such talented and hard-working researchers. Third, thanks to all the external reviewers for assisting the Program Committee in their particular areas of expertise. Fourth, we would like to thanks

all the participants of the event who made this event an intellectually stimulating one through their active contribution. We also would like to thank the iChair developers in EPFL for allowing us to use their software. Finally, we are delighted to acknowledge the partial financial support provided by Redgate, SECUi.COM, MarkAny, and EK Manpower.

November 2006

Min Surp Rhee
Byoungcheon Lee

Organization

General Chair

JooSeok Song	Yonsei University, Korea

Program Co-chairs

Min Surp Rhee	Dankook University, Korea
Byoungcheon Lee	Joongbu University, Korea

Program Committee

Giuseppe Ateniese	The Johns Hopkins University, USA
Joonsang Baek	Institute for Infocomm Research, Singapore
Alex Biryukov	University of Luxembourg, Luxembourg
John Black	University of Colorado, USA
Jean-Sebastien Coron	University of Luxembourg, Luxembourg
Jung Hee Cheon	Seoul National University, Korea
Kyo-il Chung	ETRI, Korea
Ed Dawson	Queensland University of Technology, Australia
Yevgeniy Dodis	New York University, USA
Serge Fehr	CWI Amsterdam, Netherlands
Pierre-Alain Fouque	Ecole Normale Superieure, France
Marc Girault	France Telecom, France
Philippe Golle	Palo Alto Research Center, USA
Dieter Gollmann	Hamburg University of Technology, Germany
Yongfei Han	ONETS, China
Goichiro Hanaoka	AIST, Japan
Marc Joye	Gemplus, France
Jonathan Katz	University of Maryland, USA
Hiroaki Kikuchi	Tokai University, Japan
Hwankoo Kim	Hoseo University, Korea
Kwangjo Kim	ICU, Korea
Kaoru Kurosawa	Ibaraki University, Japan
Taekyoung Kwon	Sejong University, Korea
Chi Sung Laih	Kun Shan University, Taiwan
Kwok-Yan Lam	Tsinghua University, China
Dong Hoon Lee	Korea University, Korea
Pil Joong Lee	POSTECH, Korea
Sang-Ho Lee	Ewha Womans University, Korea
Arjen Lenstra	EPFL, Switzerland
Yingjiu Li	Singapore Management University, Singapore

Helger Lipmaa	Cybernetica AS and 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
Rei Safavi-Naini	Wollongong University, 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
Kouichi Sakurai	Kyushu University, Japan
Palash Sarkar	Indian Statistical Institute, India
Nigel Smart	University of Bristol, UK
Willy Susilo	University of Wollongong, Australia
Tsuyoshi Takagi	Future University - Hakodate, Japan
Serge Vaudenay	EPFL, Switzerland
Guilin Wang	Institute for Infocomm Research, Singapore
William Whyte	NTRU Cryptosystems, USA
Michael Wiener	Cryptographic Clarity, Canada
Dongho Won	Sungkyunkwan University, Korea
Sung-Ming Yen	National Central University, Taiwan
Yongjin Yeom	NSRI, Korea
Fangguo Zhang	Sun Yat-sen University, China
Alf Zugenmaier	DoCoMo Euro-Labs, Germany

Organizing Chair

Kyung-Hyune Rhee	Pukyong National University, Korea

Organizing Committee

Chang Kyu Kim	Dong-eui University, Korea
Heekuck Oh	Hanyang University, Korea
Im-Yeong Lee	Soonchunhyang University, Korea
Sang-Uk Shin	Pukyong National University, Korea
Weon Shin	Tongmyong University, Korea
HoonJae Lee	Dongseo University, Korea
Dong Kyue Kim	Hanyang University, Korea

External Reviewers

Imad Aad
Imad Abbadi
Michelle Abdalla
Toru Akishita
Patrick Amon
Thomas Baigneres
Simon Blackburn
Marina Blanton
Brian Carrier
Michael Cheng
Sangrae Cho
Seokhyang Cho
Yong-Je Choi
Sherman Chow
Andrew Clark
Yang Cui
John Daugman
Alain Durand
Andrzej Drygajlo
Dang Nguyen Duc
Gerardo Fernandez
Matthieu Finiasz
Aline Gouget
Matthew Green
JaeCheol Ha
Genebeck Hahn
Javier Herranz
Susan Hohenberger
Jungdae Hong
Yoshiaki Hori
Jeffrey Horton
Xinyi Huang
John Ioannidis
Toshiyuki Isshiki
Tetsuya Izu
Jingak Jang
Ik rae Jeong
Seny Kamara
Jeonil Kang
Eike Kiltz
Hyung Chan Kim
Jonghyun Kim
Tae Hyun Kim
Youngsoo Kim
Shinsaku Kiyomoto
Tetsutaro Kobayashi
Divyan M. Konidala
Noboru Kunihiro
Nam-Suk Kwarc
Sven Lachmund
Julien Laganier
Vinh The Lam
HoonJae Lee
Jin Li
Wanqing Li
Hsi-Chung Lin
JongHyup Lee
MunKyu Lee
Soo-hyung Lee
Yunho Lee
Jiqiang Lu
Liang Lu
Tal Malkin
Kanta Matsuura
Breno de Medeiros
Kunihiko Miyazaki
George Mohay
Jean Monnerat
Dae Sung Moon
Kazuto Ogawa
Takeshi Okamoto
Dan Page
Jung Hyung Park
Sangjoon Park
Tae Jun Park
Sylvain Pasini
Geong Sen Poh
Rodrigo Roman
Louis Salvail
Farzad Salim
Christian Schaefer
Jae Woo Seo
Nicholas Sheppard
Jong Hoon Shin
SeongHan Shin
Douglas Sicker
Hongwei Sun
Clark Thomborson
Dongvu Tonien
Eran Tromer
Yoshifumi Ueshige
Masashi Une
Frederik Vercauteren
Duc Liem Vo
Martin Vuagnoux
Camille Vuillaume
Thomas Walter
Baodian Wei
Chung-Huang Yang
Yeon Hyeong Yang
Eunsun Yoo
Sung-Soo Yoon
Dae Hyun Yum
Rui Zhang
Chang'an Zhao
Sebastien Zimmer

Sponsoring Institutions

Redgate, Korea	http://www.redgate.co.kr/
SECUi.COM, Korea	http://www.secui.com/
MarkAny, Korea	http://www.markany.com/
EK Manpower, Korea	http://www.ekmanpower.co.kr/

Table of Contents

Network Security and Access Control

Mobile Communications Security

Forensics

Copyright Protection

Biometrics

Hash Functions – II

Public Key Cryptosystems

Digital Signatures

RFID Privacy Based on Public-Key Cryptography
(Abstract)

Serge Vaudenay

EPFL
CH-1015 Lausanne, Switzerland
http://lasecwww.epfl.ch

Abstract. RFID systems makes it possible for a server to identify known tags in wireless settings. As they become more and more pervasive, people privacy is more and more threatened. In this talk, we list a few models for privacy in RFID and compare them. We review a few protocols. We further show that strong privacy mandates the use of public-key cryptography. Finally, we present a new cryptosystem which is dedicated to tiny hardware and which can be used to design secure RFID systems achieving strong privacy.

Note: this paper contains new definitions and results that are announced in this talk. Details and proofs will appear in future papers.

Credits: the work on RFID was done together with Salvatore Bocchetti as a part of his Master Thesis [3]. We received many suggestions from Gildas Avoine. The work on the new cryptosystem was done together with Matthieu Finiasz [4] and was extended together with Jean-Philippe Aumasson, and Willi Meier. Part of it was done in the Master Thesis of Jean-Philippe Aumasson [2].

1 RFID Schemes

We consider an environment with several participants. Some are called *systems*, others are called *tags*. Every tag is associated to a system. We say that the tag *belongs* to the system. Every tag is given an identification string ID. The purpose of RFID protocols is to design a communication protocol between a system and a tag so that the system knows whether or not the tag belongs to the system and learns the tag identification string ID when the tag belongs to the system.

Tags have memory which contains a *state*. Systems have a database which contains pairs of data associated to the tags that they own. This pair consists of the ID and a key. Systems may also have cryptographic key materials.

An RFID scheme is defined by the following processes.

- An initialization algorithm for the system. This produces cryptographic key materials (if any).
- An algorithm to set up a tag. This algorithm takes an ID as input and produces a tag key K and an initial state. The latter is the initial state of the tag. The former is inserted together with ID in the database of the system that owns the tag. Note that

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 1–6, 2006.
© Springer-Verlag Berlin Heidelberg 2006

from this definition tags do not necessarily know their own ID and key. This may (or not) be part of the initial state though.
- A 2-party communication protocol between a system and a tag. Protocols are usually initiated by the system and produce two types of outputs on the reader side: a public output and a private output. We distinguish two types of protocol: identification protocols and authentication protocols. As for public outputs, the two kinds of protocols do the same. The private output of an identification protocol should be the tag ID if it belongs to the system or $\perp$ if it does not. Both outputs of an authentication protocol should be the tag 1 if it belongs to the system or 0 if it does not.

A protocol is *complete* if the output of the protocol is correct with high probability. Depending on the application, we may want to have a stronger security notion, namely *soundness*, which says whether an adversary can make the protocol output some wrong information. A critical issue is *privacy*, which means that protocols do not leak any information which may be used by adversaries to trace tags.

2 Adversaries

In an *attack*, one system is first initialize and an adversary can play with it. In addition to this, he can create tags with chosen *ID* which belong to the system or not. That is, the tag initialization algorithm is run, the tag with specified initial state is created, and the database of the system is updated in the case where the tag belongs to the system. Here the adversary does not see the tag key or initial state. In addition to creating new tags, the adversary can play with the system and the tags. We distinguish two kinds of tags: tags that are *free* from tags that are *drawn*. Tags can move from a free status to a drawn one and vice versa. A drawn tag is a tag which is close to the adversary so that the adversary can trace it during the entire time it is a drawn tag. For this, drawn tags are identified by a temporary identity that we call a *virtual tag*.

More concretely, we assume that the adversary has access to the following oracles.

- $\mathsf{Init}(\mathsf{ID}, b)$ initializes new (free) tags of specified ID which belongs to System or not depending on bit b.
- $\mathsf{GetTag}(\mathsf{distribution}) \rightarrow (\mathsf{vtag}_1, b_1, \ldots, \mathsf{vtag}_n, b_n)$ draws one or several free tags at random with chosen probability distribution. This oracle returns "virtual tags" names and bits telling whether they belong to the system or not.
- $\mathsf{Free}(\mathsf{vtag})$ frees a drawn tag.
- $\mathsf{Launch} \rightarrow \pi$ launches a new protocol instance with reader.
- $\mathsf{SendReader}(m, \pi) \rightarrow m'$ resp. $\mathsf{SendTag}(m, \mathsf{vtag}) \rightarrow m'$ sends protocol message m to reader resp. a drawn tag and returns the answer m' (if any). By convention, we write $\mathsf{Execute}(\mathsf{vtag}) \rightarrow (\pi, \mathsf{transcript})$ as a macro oracle call instead of one $\mathsf{Launch} \rightarrow \pi$ followed by a succession of $\mathsf{SendReader}(m_i, \pi) \rightarrow m_{i+1}$ and $\mathsf{SendTag}(m_{i+1}, \mathsf{vtag}) \rightarrow m_{i+2}$ calls. The protocol transcript is the concatenation of all messages m_i.
- $\mathsf{Result}(\pi) \rightarrow x$ tells 1 if the output of the protocol instance π is a tag ID or 0 if the output is $\perp$.
- $\mathsf{Corrupt}(\mathsf{vtag}) \rightarrow S$ corrupts a drawn tag and gets its internal state S.

We define several classes of adversaries.

- *Strong* adversaries can use the oracles as they want.
- *Forward* adversaries can only use Corrupt queries at the end of the attack. That is, a Corrupt query can only be followed by other Corrupt queries.
- *Weak* adversaries are not allowed to make Corrupt queries.
- *Narrow-strong* (resp. narrow-forward, narrow-weak) adversaries are strong (resp. forward, weak) adversaries who are not allowed to make Result queries.

3 Security of RFID Schemes

Let us consider an arbitrary adversary which can be written as follows.

1: $\mathsf{Init}(1,b_1)$, ..., $\mathsf{Init}(n,b_n)$
2: pick $i \in \{1,\ldots,n\}$ at random
3: $(\mathsf{vtag},b) \leftarrow \mathsf{GetTag}(i)$
4: $\pi \leftarrow \mathsf{Execute}(\mathsf{vtag})$

The adversary creates n tags which belong or not to the system. Then, it draws one tag and runs a protocol. We say that this adversary fails iff the output of the protocol is what it is meant to be, namely i when $b_i = 1$ and $\perp$ otherwise. We say that the protocol is *complete* iff the probability of success of any of these adversaries is negligible.

Let us consider an arbitrary adversary which can be written as follows.

1: **for** $i = 1$ to n **do**
2: $\quad\mathsf{Init}(i,1)$
3: $\quad\mathsf{vtag}_i \leftarrow \mathsf{GetTag}(i)$
4: **end for**
5: (training phase) do any oracle call except Init, GetTag, Free
6: $\pi \leftarrow$ Launch
7: (attack phase) do any oracle call except Init, GetTag, Free

We say that the adversary succeeds iff

- instance π is complete at the end of the attack phase,
- the output of π is $\mathsf{ID} \neq \perp$ (i.e. π identified a legitimate tag ID),
- tag ID did not complete a protocol run during the attack phase,
- tag ID was not corrupted.

We say that the protocol is *sound* iff the probability of success of any of these adversaries is negligible.

4 Privacy

To define privacy, we consider adversaries who output a list of virtual tags and a relation between their ID strings. The adversary wins if the ID strings of these tags satisfy the relation. Since some adversaries may win by giving trivial relations, we define the significance of an adversary by his ability to distinguish from a simulated run. More concretely, a *blinder* is an interface between the adversary and the oracles which let all

queries pass except Lauch/SendReader/SendTag/Result queries whose output are simulated. The adversary "plugged" to a blinder is an adversary by itself who never queries the Lauch/SendReader/SendTag/Result oracles. The original adversary is significant if for any blinder the difference of the wining probability of the two adversaries is high.

An RFID system provides strong (resp. forward, weak, narrow-strong, narrow-forward, narrow-weak) privacy if there is no significant strong (resp. forward, weak, narrow-strong, narrow-forward, narrow-weak) adversary. The following implications are straightforward.

$$\begin{array}{ccccc} \text{strong privacy} & \Rightarrow & \text{forward privacy} & \Rightarrow & \text{weak privacy} \\ \Downarrow & & \Downarrow & & \Downarrow \\ \text{narrow-strong privacy} & \Rightarrow & \text{narrow-forward privacy} & \Rightarrow & \text{narrow-weak privacy} \end{array}$$

5 Our Results

We can prove that

- our six privacy notions are pairwise different;
- strong privacy cannot be achieved;
- forward privacy can be achieved without public-key cryptography, in principle;
- an RFID system with narrow-strong privacy can be transformed into a secure key agreement protocol, which implies that this notion of privacy cannot be achieve without paying the price of public-key cryptography;
- a semantically secure public-key cryptosystem can be used to make an RFID system with *narrow*-strong privacy;
- a public-key cryptosystem secure against adaptive chosen ciphertext attacks can be used to make a secure RFID system with *forward* privacy and *narrow*-strong privacy.

The protocol in the last two results works as follows.

- The initialization algorithm for the system generates a public/private key pair for the cryptosystem.
- The setup algorithm for the tag generates a random key K for the tag and set the initial state to the vector including ID, K, and the public key of the system.
- The identification protocol works as depicted on Fig. 1. The system first picks a random a, sends it to the tag. Then, the tag encrypts ID, K, and a together by using the public key and sends the ciphertext to the system. Finally, the system decrypts using the private key, checks that a is correct and that ID and K are consistent with the database. The output of the protocol is ID.

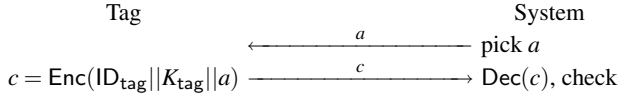


Fig. 1. Identification Protocol Based on a Public-Key Cryptosystem

6 TCHo

We present here a simple cryptosystem that can be used for tiny hardwares. We use a security parameter s which defines some parameters w, d, ℓ, $d_{\min}$, $d_{\max}$, γ, k. To make it more precise, we can take

$$w = 45 \quad d = 25\,820 \quad \ell = 50\,000 \quad d_{\min} = 5\,800 \quad d_{\max} = 7\,000 \quad \gamma = 0.9810 \quad k = 128.$$

Asymptotically, we take

$$w = \Theta(s) \quad d = \Theta(s^3) = \ell \quad d_{\min} = \Theta(s^2) = d_{\max} \quad \gamma = 1 - \Theta\left(\frac{1}{s}\right) \quad k = \Theta(s).$$

Key generation. We pick a random polynomial K over $\mathsf{GF}(2)$ of degree d with constant factor 1 and weight w until it has a primitive factor P of degree d_P in $[d_{\min}, d_{\max}]$. The public key is P. It is of length at most $d_{\max} = O(s^2)$ The private key is K. It is of length at most $w \log_2 d = O(s \log s)$ The complexity is $O(s^6 \log s \log\log s)$.

Encryption. To encrypt a k-bit plaintext, we set an LFSR with characteristic polynomial P to a random string, we produce a bit stream of length ℓ. We XOR it to ℓ/k repetitions of the plaintext. We XOR it again to the output of a random source producing ℓ independent bits with bias γ. The result is the ciphertext: a bit-string of length ℓ. Encryption is depicted on Fig. 2. The complexity is $O(s^5)$. The plaintext is of length $k = \Theta(s)$. The ciphertext is of length $\ell = O(s^3)$.

Decryption. To decrypt, we make combination of the ciphertext bits by using K. We obtain $\ell - d$ bits. We do majority logic decoding and recover the plaintext. The complexity is $O(s^4)$.

Reliability, performances, and security. Heuristic arguments show that the probability that decryption does not produce the right plaintext is less than $9 \cdot 10^{-9}$ for our parameters. (Asymptotically, this probability is $\exp\left(-\Omega(s^2)\right)$, heuristically.) The cost to implement encryption in RFID tags relates to the cost of a random generator and an

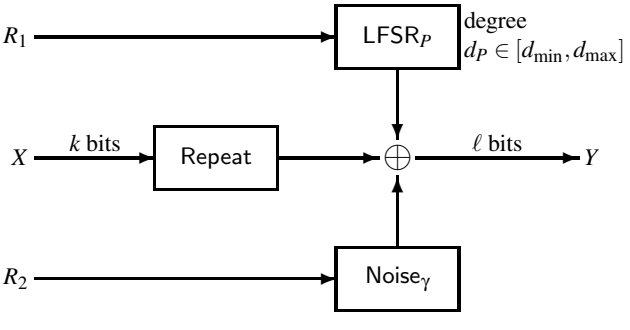


Fig. 2. TCHo Encryption

LFSR of d_P bits (that is at most 7000 gates here). We can encrypt arbitrary messages using hybrid encryption, requiring an additional symmetric encryption scheme. Our scheme is semantically secure provided that

- it is hard to find a multiple of a polynomial of degree d_P with weight w and degree d when such polynomial exists;
- we cannot distinguish the output of the LFSR of length d_P XORed to biased bits of bias γ from a uniformly distributed string.

So far, the best algorithm to break semantic security in TCHo with our parameters vector has an advantage/complexity ratio of 2^{-65} with pessimistic estimates. Asymptotically, this best ratio is $\exp(-\Omega(s))/s^2$.

7 IND-CCA Construction

Given two random oracles F and H, the [1] construction based on tag-KEM/DEM transforms TCHo into an IND-CCA secure public-key cryptosystem. To encrypt a plaintext X with random bits σ, we compute $y = X + F(\sigma)$ (the *data encapsulation*) and the TCHo encryption of σ with random bits $(R_1, R_2) = H(\sigma, y)$ (the *key encapsulation* with tag y). The ciphertext is the TCHo ciphertext concatenated with y. Namely,

$$\mathsf{Enc}(X;\sigma) = \mathsf{TCHo.enc}(X;H(\sigma,y)) \,||\, y.$$

To decrypt, we first decrypt the TCHo ciphertext to recover σ. We subtract $F(\sigma)$ to y and get the plaintext X. Additionally, we check that the TCHo encryption used the right $H(\sigma, y)$ random bits.

References

1. M. Abe, R. Gennaro, K. Kurosawa, V. Shoup. Tag-KEM/DEM: a New Framework for Hybrid Encryption and a New Analysis of Kurosawa-Desmedt KEM. In *Advances in Cryptology EUROCRYPT'05*, Aarhus, Denmark, Lecture Notes in Computer Science 3494, pp. 128–146, Springer-Verlag, 2005.
2. J.-P. Aumasson. A Novel Asymmetric Scheme with Stream Cipher Construction. Master Thesis. 2006. `http://lasecwww.epfl.ch/abstracts/abstract_tcho.shtml`
3. S. Bocchetti. Security and Privacy in RFID Protocols. Master Thesis. 2006. `http://lasecwww.epfl.ch/abstracts/abstract_RFID_bocchetti.shtml`
4. M. Finiasz, S. Vaudenay. When Stream Cipher Analysis Meets Public-Key Cryptography. (Invited Talk.) To appear in the proceedings of SAC' 2006. Lecture Notes in Computer Science, Springer. 2006.

Generic Attacks on Symmetric Ciphers

Palash Sarkar

Applied Statistics Unit
Indian Statistical Institute
203 B. T. Road, Kolkata,
India 700 108
palash@isical.ac.in

Abstract. In this talk, we consider the important problem of generic attacks on symmetric cipher algorithms. We review the work on exhaustive search attacks, especially with respect to DES. An interesting variant of exhaustive search is to use a so-called time-memory trade-off (TMTO) attack. This attack and its many variants will be described and recent research on TMTO attacks will be surveyed. The effectiveness of generic attacks is determined by improvements in computer technology and the related costs. We will attempt to derive suitable relationships between these parameters. Such relationships will be used to address questions about how secure are 80-bit and 128-bit ciphers. The talk will also cover research on generic attacks using non-conventional technology. Finally, we will relate some modern ideas to earlier ideas used in the cryptanalysis of Enigma.

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, p. 7, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Improved Collision Attack on the Hash Function Proposed at PKC'98*

Florian Mendel**, Norbert Pramstaller, and Christian Rechberger

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria
Florian.Mendel@iaik.TUGraz.at

Abstract. In this article, we present an improved collision attack on the hash function proposed by Shin *et al.* at PKC'98. The attack has a complexity of about $2^{20.5}$ hash computations, while the previous attack of Chang *et al.* presented at SAC 2002 has a complexity of about $2^{37.13}$ hash computations. In the analysis of the hash function we combined existing approaches with recent results in cryptanalysis of hash functions. We show that message-dependent rotations can be exploited to construct collisions. The weak design of the step function facilitates high-probability multi-block collisions.

Keywords: cryptanalysis, collision attack, differential attack, collision, near-collision, hash functions.

1 Introduction

Recently, weaknesses in many commonly used hash functions, such as MD5 and SHA-1 have been found. These breakthrough results in the cryptanalysis of hash functions are the motivation for intensive research in the design and analysis of hash functions. It is of special interest whether or not existing attack methods, which have been successfully used to break MD5 and SHA-1, can be extended to other hash functions as well. Based on this motivation, we present a collision attack on the hash function proposed by Shin *et al.* at PKC'98. The attack adapts and extends existing methods leading to an attack complexity of about $2^{20.5}$ hash computations. This is an improvement by a factor of $2^{16.5}$ compared to the collision attack presented by Chang *et al.* at SAC 2002. In addition to the improved collision attack this article illustrates how powerful recently invented methods are and shows how they can be extended to other hash functions such as the hash function proposal from PKC'98.

* The work described in this paper has been supported in part by the European Commission through the IST Programme under contract IST2002507 932 ECRYPT. The information in this paper is provided as is, and no guarantee or warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

** This author is supported by the Austrian Science Fund (FWF), project P18138.

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 8–21, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Table 1. Notation

Notation	Meaning
$A \vee B$	logical OR of two bit-strings A and B
$A \wedge B$	logical AND of two bit-strings A and B
$A \oplus B$	logical XOR of two bit-strings A and B
$A \lll n$	bit-rotation of A by n positions to the left
$A \ggg n$	bit-rotation of A by n positions to the right
M_j	message block j (512-bits)
m_i	message word i (32-bits)
w_i	expanded message word i (32-bits)
step	single execution of the step function
round	set of consecutive *steps*, has a size of 24 (1 *round* = 24 *steps*)

An important contribution of this article is that we analyze message-dependent rotations, a property not existing for instance in MD5 and SHA-1. Our conclusions are that the message-dependent rotations decrease the security of the hash function. The weak design of the step function in combination with the used Boolean functions facilitates the construction of high-probability multi-block collisions.

The remainder of this article is structured as follows. A description of the hash function is given in Section 2. The basic attack strategy that is used to improve all existing collision attacks on the hash function is described in Section 3. In Section 4, we present the characteristic used for the improved collision attack. Based on this characteristic we construct a near-collision in Section 5 and finally we present a collision in Section 6. A detailed analysis of the overall attack complexity is given in Section 7. A sample colliding message pair is presented in Section 8 and conclusions are given in Section 9.

2 Description of the Hash Function Proposed at PKC'98

The hash function was proposed by Shin *et al.* [5] at PKC'98. It is an iterative hash function that processes 512-bit input message blocks and produces a 160-bit hash value. In the following, we briefly describe the hash function. It basically consists of two parts: message expansion and state update transformation. A detailed description of the hash function is given in [5].

Since Shin *et al.* did not name their hash function, we will refer to it as PKC-hash for the remainder of this article. Throughout the remainder of this article, we will follow the notation given in Table 1.

Message Expansion. The message expansion of PKC-hash is a permutation of 24 expanded message words in each round, where different permutation values

are used in each round. The 24 expanded message words w_i used in each round are constructed from the 16 input message words m_i in the following way:

$$w_i = \begin{cases} m_i & 0 \leq i \leq 15 \\ (w_{i-4} \oplus w_{i-9} \oplus w_{i-14} \oplus w_{i-16}) \lll 1 & 16 \leq i \leq 23. \end{cases}$$

For the ordering of the message words the permutation ρ is used.

Round 1	Round 2	Round 3	Round 4
id	ρ	ρ^2	ρ^3

The permutation ρ is defined as following.

i	0	1	2	3	4	5	6	7	8	9	10	11
$\rho(i)$	4	21	17	1	23	18	12	10	5	16	8	0
i	12	13	14	15	16	17	18	19	20	21	22	23
$\rho(i)$	20	3	22	6	11	19	15	2	7	14	9	13

State Update Transformation. The state update transformation starts from a (fixed) initial value IV of five 32-bit registers and updates them in 4 rounds of 24 steps each. Figure 1 shows one step of the state update transformation of the hash function.

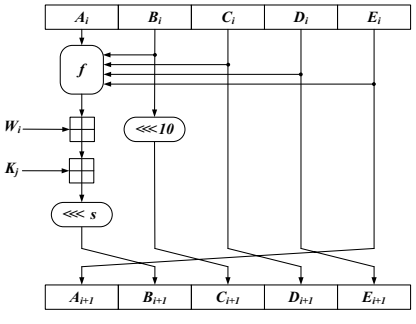


Fig. 1. The step function of the hash function

The function f is different in each round: f_0 is used in the first round, f_1 is used in round 2 and round 4, and f_2 is used in round 3.

$$f_0(x_1, x_2, x_3, x_4, x_5) = (x_1 \wedge x_2) \oplus (x_3 \wedge x_4) \oplus (x_2 \wedge x_3 \wedge x_4) \oplus x_5$$
$$f_1(x_1, x_2, x_3, x_4, x_5) = x_2 \oplus ((x_4 \wedge x_5) \vee (x_1 \wedge x_3))$$
$$f_2(x_1, x_2, x_3, x_4, x_5) = x_1 \oplus (x_2 \wedge (x_1 \oplus x_4)) \oplus (((x_1 \wedge x_4) \oplus x_3) \vee x_5)$$

A step constant K_j is added in every step; the constant is different for each round. Different rotation values s_i are used in each step. The rotation values are dependent on the message words. The rotation values s_i, for $i = 0, \ldots, 23$ are calculated in the following way:

$$s_i = w_i \mod 32$$

The rotation values are permuted in each round. Again the permutation ρ is used, but in reverse sequence.

Round 1	Round 2	Round 3	Round 4
ρ^3	ρ^2	ρ^1	id

After the last step of the state update transformation, the initial value and the output values of the last step are combined, resulting in the final value of one iteration known as Davies-Meyer hash construction (feed forward). In detail, the feed forward is a word-wise modular addition of the IV and the output of the state update transformation. The result is the final hash value or the initial value for the next message block.

3 Our Attack Strategy

In the following, we present the attack strategy we use to improve the collision attack on PKC-hash. It is based on recent results in cryptanalysis of hash functions [6,7,8]. The attack can be basically described as follows.

1. Find a characteristic for the hash function that holds with high probability for the last 3 rounds of the hash function.
2. Find a characteristic (not necessary with high probability) for the first round.
3. Use basic message modification techniques to fulfill all conditions for the characteristic in the first round.
4. Use random trials to find values for the message bits such that the message also follows the characteristic in the last 3 rounds.

Observation 1. *For the first steps the probability of the characteristic is not important, because the conditions that have to be satisfied such that the characteristic holds can be easily fulfilled for these steps using basic message modification techniques [6,8].*

Observation 2. *Multi-block messages can be used to turn related near-collisions into a collision.*

Since Biham and Chen observed in [1] that near-collisions are easier to find than collisions, we will use Observation 2 in Section 4 to improve our attack.

To find a characteristic which holds with high probability, we exploit the structure of the hash function. Considering the design of the step function we made the following observations. We use these observations in Section 4 to construct a characteristic which holds with high probability.

Observation 3. *The rotation values s_j of the state update transformation are dependent on the values of the expanded message words.*

This gives the attacker more degrees of freedom for constructing a *good* characteristic. The observation can be used to improve the probability of the characteristic significantly, see Section 4.

Observation 4. *Due to the message-dependent rotation values the step function is not invertible.*

This means we could try to construct collisions by using different w_j, w'_j and s_j, s'_j which have the property that $B_{j+1} = B'_{j+1}$. However, the complexity for constructing a collision with this method is too high for a reasonable attack.

Observation 5. *The function f can either preserve or absorb an input difference. This gives the attacker more flexibility for constructing the characteristic.*

Observation 6. *Only the expanded message word w and the output of the f function is used to update state variable B in each step.*

From Observation 5 it follows that differences in the state variables can be canceled by using the differential properties of the f function. In particular, a difference in B_{j+1} introduced in step j through a difference in the expanded message word w_j (referred to as disturbance) can be canceled within a few steps.

The number of steps needed for canceling a single disturbance depends on the function f and the rotation values s_j of the step function. While we need at least 5 steps to cancel a disturbance in round 3, we need at least 6 steps to cancel a disturbance in round 1, 2 and 4. This is due to the fact that we cannot always block the input differences of f_0 and f_1. A detailed analysis of the differential properties of f_0, f_1 and f_2 is given in [2]. In Appendix A, we give the local collisions and related probabilities for each round of the hash function.

However, to get a characteristic that holds with high probability, we have to minimize the number of disturbances in each round. This can be done by minimizing the number of differences in the expanded message.

Observation 7. *The minimal number of differences in the expanded message words is 2 for each round (8 in total).*

This follows from the inspection of the linear message expansion, which uses 16 input message words to generate 24 expanded message words. A permutation of these 24 words is used in each round of the hash function and hence the number of disturbances in each round is the same. Based on these observations, we will construct a characteristic which holds with high probability in Section 4.

4 The New Improved Collision Attack

In this section, we describe the characteristic we use for the improved collision attack on PKC-hash. Before presenting the characteristic in Section 4.2 and Section 4.3, we briefly describe how the characteristic has been obtained in the following section.

4.1 Finding a *good* Characteristic

In the past, it has been shown that it is easier to find near-collision than collision within a hash function. Since two message blocks can be used to turn a near-collision into a collision (see Observation 2), we will consider near-collisions in the analysis as well. Note that the attacker has full control on the expanded message words in the first 16 steps of the hash function. Hence, it is easy to find a message that follows the characteristic in the first 16 steps, because the conditions for the first 16 steps can be fulfilled by using basic message modification techniques. Therefore, only the probability of the characteristic in the remaining 80 steps determines the attack complexity. In order to keep the complexity low, we want to have as few disturbances in the last 80 steps as possible. This can be achieved by choosing the differences in the message words in such a way that there are only a few differences in the expanded message. We have found the following 4 options which have only 2 differences in the expanded message words in each round. Note that 2 is a matching lower bound for the number of differences in the expanded message words in each round (see Observation 7).

$$\Delta w_{8,j} = \Delta w_{13,j} \tag{1}$$
$$\Delta w_{9,j} = \Delta w_{14,j} \tag{2}$$
$$\Delta w_{10,j} = \Delta w_{15,j} \tag{3}$$
$$\Delta w_{11,j} = \Delta w_{20,(j+1) \mod 32} \tag{4}$$

Out of this four cases, we select option (2) with $j = 32$, *i.e.* $\Delta w_9 = \Delta w_{14} = 80000000$ to maximize the probability of the characteristic. By choosing the difference in the MSB no conditions are needed for the modular addition, which decreases the attack complexity. This is due to the fact that modular addition behaves like an XOR for differences in the MSB.

Furthermore, we reduce the number of local collisions in round 1, 2, and 4 from the expected value of 2 to 1 by selecting option (2). Therefore, the probability of our characteristic is much higher than the probability of the characteristics used in [2] and [3]. On the one hand, we increase the probability of the characteristic remarkably, but on the other hand this leads to a nonzero difference in the state variables after the last step of the state update transformation (a near-collision). However, as it will be described in Section 6, we can use a second message block to turn this near-collision into a collision without significantly increasing the attack complexity.

4.2 Characteristic for the First Round

The characteristic for the first round (24 steps) has probability $2^{-7.8}$. However, all the conditions for the first 16 steps can be fulfilled by basic message modification techniques. Since the characteristic has probability 1 for steps 16-23 (see Table 2), the probability for the characteristic in the first round is always 1. Therefore, the attack complexity only depends on the probability of the characteristic in the last 3 rounds. In Section 4.3, we give a characteristic for the remaining 3 rounds which holds with high probability. The characteristic for the first round of the hash function is given in Table 2. To improve readability, we use hexadecimal notation and denote the zero difference by '0'.

Table 2. Characteristic for the first round of PKC-hash

step	ΔA	ΔB	ΔC	ΔD	ΔE	Δw	s	probability
0	0	0	0	0	0	0	-	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
8	0	0	0	0	0	0	-	1
9	0	0	0	0	0	80000000	22	1
10	0	00200000	0	0	0	0	0	1/4
11	0	00200000	80000000	0	0	0	-	3/8
12	0	0	80000000	80000000	0	0	-	3/4
13	0	0	0	80000000	80000000	0	-	1/4
14	80000000	0	0	0	80000000	80000000	-	1/2
15	80000000	0	0	0	0	0	-	1/2
16	0	0	0	0	0	0	-	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
23	0	0	0	0	0	0	-	1
	0	0	0	0	0			$2^{-7.8}$

4.3 Characteristic for the Last 3 Rounds

Since all the conditions on the characteristic in the first round can be easily fulfilled, only the probability of the characteristic in the last 3 rounds determines the attack complexity. Therefore, we are searching for a characteristic which holds with high probability in the last 3 rounds. In this section, we give a characteristic for the last 3 rounds which has probability of $2^{-20.5}$. To maximize the probability of the characteristic, we use the fact that the rotation values of PKC-hash are dependent on the expanded message words (see Observation 3). The rotation values can be set to arbitrary values by setting additional conditions on the expanded message words. The characteristic and necessary rotation values are given Table 3. Note that we do not count the conditions for the rotation values to the attack complexity, since these can be easily fulfilled in advance by fixing the values of the expanded message words.

Table 3. Characteristic for the last 3 rounds (round 2-4) of PKC-hash

step	ΔA	ΔB	ΔC	ΔD	ΔE	Δw	s	probability
24	0	0	0	0	0	0	-	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
44	0	0	0	0	0	0	-	1
45	0	0	0	0	0	80000000	0	1
46	0	80000000	0	0	0	80000000	-	1
47	0	0	00000200	0	0	0	-	5/8
48	0	0	0	00000200	0	0	-	1/2
49	0	0	0	0	00000200	80000000	0	1/2
50	00000200	80000000	0	0	0	0	-	1/4
51	0	0	00000200	0	0	0	-	1/2
52	0	0	0	00000200	0	0	-	1/2
53	0	0	0	0	00000200	0	-	1/2
54	00000200	0	0	0	0	0	-	1/2
55	0	0	0	0	0	0	-	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
61	0	0	0	0	0	0	-	1
62	0	0	0	0	0	80000000	0	1
63	0	80000000	0	0	0	0	-	1/2
64	0	0	00000200	0	0	0	-	1/2
65	0	0	0	00000200	0	0	-	1/2
66	0	0	0	0	00000200	0	-	1/2
67	00000200	0	0	0	0	0	-	1/2
68	0	0	0	0	0	0	-	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
74	0	0	0	0	0	0	-	1
75	0	0	0	0	0	80000000	0	1
76	0	80000000	0	0	0	0	10	1
77	0	00000200	00000200	0	0	0	-	3/8
78	0	0	00080000	00000200	0	0	-	25/64
79	0	0	0	00080000	00000200	0	-	25/64
80	00000200	0	0	0	00080000	0	-	25/64
81	00080000	0	0	0	0	0	-	5/8
82	0	0	0	0	0	0	-	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
92	0	0	0	0	0	0	-	1
93	0	0	0	0	0	80000000	0	1
94	0	80000000	0	0	0	0	0	1
95	0	80000000	00000200	0	0	0	0	5/8
	0	80000000	00000200	00000200	0			$2^{-20.5}$

5 A Near-Collision Producing Characteristic

The characteristic for the first round (Table 2) and the characteristic for the remaining 3 rounds (Table 3) can be combined to construct a near-collision in one iteration of the hash function. Using the most naive method (random trials) to find a message following the characteristic in the last 80 steps, we get a complexity close to $2^{20.5}$ hash computations. Hence, a near-collision can be found in PKC-hash with complexity about $2^{20.5}$ hash computations. By using two message blocks, we can turn this near-collision into a collision. This is described in detail in the following section.

6 Collision Producing Characteristic

In [8], Wang *et al.* show how a two-block message can be used to construct a collision for MD5. The main idea is that a second message block can be used to turn a near-collision after the first block into a collision after feed forward of the second block with a certain probability. Therefore, a slightly modified characteristic is required in the first round of the second block. This is depicted in Fig. 2.

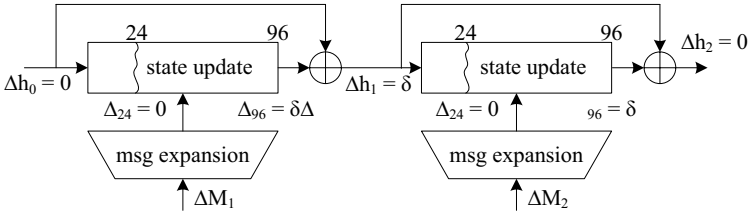


Fig. 2. A two-block collision in the hash function

While we use a characteristic of the form:

$$\Delta h_0(0,0,0,0,0) \mapsto \Delta_{24}(0,0,0,0,0) \mapsto \Delta_{96}(0,2^{31},2^9,2^9,0) \tag{5}$$

in the first message block, we need a characteristic of the form:

$$\Delta h_1(0,2^{31},2^9,2^9,0) \mapsto \Delta_{24}(0,0,0,0,0) \mapsto \Delta_{96}(0,2^{31},2^9,2^9,0) \tag{6}$$

in the second block. Constructing such a characteristic is quite easy in our particular case. Due to the weak design of the step function we can block differences in the state variables in each step of the hash function depending on the differential properties of the f function (see Observation 5). Hence, we can cancel all differences in the state variables at the beginning of the second block within a few steps in the first round. Note that the differential behavior of the f function depends on the values of the state variables and we cannot control it in the first

steps of the second block. This is due to the fact that the initial value of the second block is fixed by calculating the first block. Therefore, in principle the characteristic for the second block cannot be fixed until the first block has been calculated.

However, in practice due to the large degree of freedom we have in our collision attack, we can use the same characteristic in the second block as we use in the first block and cancel all differences in the state variables in the first 9 steps of the second block without affecting any of the conditions of the original characteristic.

Note that the probability of the characteristic of the second message block for the first 16 steps can again be neglected. Hence, the probability of the second block is also $2^{20.5}$. By combining both message blocks we can construct a collision after the feed forward of the second block with a certain probability. In detail, we can find a two message block collision for PKC-hash with probability close to $2^{-22.85}$. A detailed analysis is given the following section.

7 Overall Collision Attack Complexity

The complexity of the collision attack only depends on the probability of the characteristic in the last 3 rounds of both message blocks. Note that some additional conditions have to be met to guarantee that all differences cancel out in the feed forward after the second block. Therefore, the second block has a slightly lower probability than the first block.

As shown in Section 4, the characteristic for the last 3 rounds has a probability of $2^{-20.5}$ and therefore a straightforward implementation of the collision-search algorithm would yield a complexity of about $2^{20.5}$ hash computations for the first block and $2^{20.5} \cdot 2^2$ hash computations for the second block. In order to obtain a collision after the feed-forward of the second block, the following two conditions on the state variables at the output of the second block have to be satisfied

$$C_{0,10} \oplus C_{81,10} = 1$$
$$D_{0,10} \oplus D_{81,10} = 1$$

to guarantee that all differences cancel in the feed forward of the second block. Since the third difference is in the MSB (see Equation (6)), the difference cancels out with probability 1 and no condition is needed. Hence, the second block has a complexity of $2^{20.5} \cdot 2^2$ hash computations and the final attack complexity would be $2^{20.5} + 2^{20.5} \cdot 2^2 = 2^{22.85}$ hash computations to construct a collision in the hash function.

However, there are several simple methods to improve the efficiency of the attack. In [9], Wang *et al.* use a so-called *early-stop* technique to improve the attack complexity for SHA-0 by a factor of 8. The main idea is that only a few steps have to be computed after the basic message modification to check whether or not the message follows the characteristic. If the message does not follow the characteristic the collision-search algorithm aborts the current computation

and starts again with a new message. It is clear that this reduces the attack complexity. In our case, we can improve the attack complexity by a factor of 8/3, since we have to calculate on average 36 steps out of 96 steps to check whether the chosen message follows the characteristic or not.

Furthermore, it has been shown recently in [4] that it is useful to look at all possible characteristics in the second part of the attack (the part after message modification, *i.e.* round 2-4) to get an accurate estimation of the attack complexity. Since we use random trials to find a message following the characteristic after the first round, we do not need to stick to the characteristic given in Section 4.3. The only constraint we have is that there has to be a certain output difference after step 96. Hence, other characteristics (with lower probability) for the last 3 rounds do contribute as well. Therefore, the effective attack complexity is slightly lower. We have done this analysis for the third round of the hash function and have achieved an improvement by a factor of about 2. Note that we have fixed additional rotation values in round 3 to maximize that factor.

Hence, we can update the final attack complexity to $2^{20.5-2.4}$ hash computations for the first message block and $2^{22.5-2.4}$ for the second block. Therefore, the final attack complexity is $2^{20.5-2.4} + 2^{22.5-2.4} \approx 2^{20.5}$ hash computations.

8 A Colliding Message for the Hash Function

Applying our improved collision attack to PKC-hash, we can construct a two message block collision with a complexity of $2^{20.5}$ hash computations. The colliding message pair is given in Table 4. Note that h_0 is the initial value, h_1 is the intermediate hash value after the first block, and h_2 is the final hash value after the second block (see Fig. 2 in Section 4).

Table 4. Colliding message pair for the hash function

h_0	67452301 EFCDAB89 98BADCEF 10325476 C3D2E1F0
M_0	210A7ED6 69EC9B20 71E79487 ECDB11C0 CCE394EA EBA83742 44A26AC0 9A644570 E78BA0F0 D0CD3794 AC8A28BB 29303480 F9A7F632 0F886620 28E118E9 39E4CF77
M_0'	210A7ED6 69EC9B20 71E79487 ECDB11C0 CCE394EA EBA83742 44A26AC0 9A644570 E78BA0F0 50CD3794 AC8A28BB 29303480 F9A7F632 0F886620 A8E118E9 39E4CF77
ΔM_0	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 80000000 00000000 00000000 00000000 00000000 00000000 80000000 00000000
h_1	E1A286A2 5E619A9E F341C16E 8A3B4927 AFCFF8D2
h_1'	E1A286A2 DE619A9E F341C36E 8A3B4B27 AFCFF8D2
Δh_1	00000000 80000000 00000200 00000200 00000000
M_1	89221C96 237E9860 76346FC0 C5F4F3E0 B66B5EAA D025B4C9 BE742420 E1362EC6 084DB7A0 3F231F9A D883A03A AFCB10A0 CDA285EE 24630660 BE9599C0 F6F63E65
M_1'	89221C96 237E9860 76346FC0 C5F4F3E0 B66B5EAA D025B4C9 BE742420 E1362EC6 084DB7A0 BF231F9A D883A03A AFCB10A0 CDA285EE 24630660 3E9599C0 F6F63E65
ΔM_1	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 80000000 00000000 00000000 00000000 00000000 00000000 80000000 00000000
h_2	B141281F FC5A987C FB473C39 A9864410 21ACD08E
h_2'	B141281F FC5A987C FB473C39 A9864410 21ACD08E

9 Conclusion

In this article, we used recent results in the cryptanalysis of hash functions to improve the collision attack on the hash function proposed by Shin *et al.* at PCK'98. We have shown that a collision can be found in the hash function with a complexity below $2^{20.5}$ hash computations. In detail, we improve the results of Chang *et al.* [2] with the new collision attack by a factor of $2^{16.5}$ using a new differential characteristic and exploiting basic message modification techniques and multi-block collisions.

We point out that the weakness of the hash function comes from the message-dependent rotation values and the weak design of step function. Firstly, the degrees of freedom the attacker has to choose the rotation values can be used to increase the probability of the attack. Secondly, the weak design of the step function facilitates high-probability multi-block collisions. Differences in state variables in the first step can easily be canceled within a few steps using the differential properties of the f function.

Hence, we conclude that the Boolean functions used in the state update transformation have to be chosen carefully and using only the expanded message words and the output of the f function to update the state variables is insufficient. Furthermore, rotation values depending on the message words can reduce the security of hash functions.

Acknowledgements

The authors wish to thank Vincent Rijmen, and the anonymous referees for useful comments and discussions.

References

1. Eli Biham and Rafi Chen. Near-Collisions of SHA-0. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *LNCS*, pages 290–305. Springer, 2004.
2. Donghoon Chang, Jaechul Sung, Soo Hak Sung, Sangjin Lee, and Jongin Lim. Full-Round Differential Attack on the Original Version of the Hash Function Proposed at PKC'98. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, volume 2595 of *Lecture Notes in Computer Science*, pages 160–174. Springer, 2002.
3. Daewan Han, Sangwoo Park, and Seongtaek Chee. Cryptanalysis of the Modified Version of the Hash Function Proposed at PKC'98. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers*, volume 2365 of *Lecture Notes in Computer Science*, pages 252–262. Springer, 2002.
4. Florian Mendel, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. The Impact of Carries on the Complexity of Collision Attacks on SHA-1. In

Matt Robshow, editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Pre-Proceedings*, 2006.

5. Sang Uk Shin, Kyung Hyune Rhee, Dae-Hyun Ryu, and Sangjin Lee. A New Hash Function Based on MDx-Family and Its Application to MAC. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography, First International Workshop on Practice and Theory in Public Key Cryptography, PKC '98, Pacifico Yokohama, Japan, February 5-6, 1998, Proceedings*, volume 1431 of *Lecture Notes in Computer Science*, pages 234–246. Springer, 1998.
6. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 1–18. Springer, 2005.
7. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.
8. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.
9. Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient Collision Search Attacks on SHA-0. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 1–16. Springer, 2005.

A Local Collisions

In this section, we will give the best local collision for each round of the PKC-hash. Since f_1 is used in round 2 and round 4 the local collisions are equal for both rounds.

Table 5. Local Collision in Round 1 (f_0)

step	ΔA	ΔB	ΔC	ΔD	ΔE	Δw	s	probability
j	0	0	0	0	0	80000000	0	1
j+1	0	80000000	0	0	0	0	0	1/2
j+2	0	80000000	00000200	0	0	0	-	3/8
j+3	0	0	00000200	00000200	0	0	-	3/4
j+4	0	0	0	00000200	00000200	0	-	1/4
j+5	00000200	0	0	0	00000200	0	-	1/2
j+6	00000200	0	0	0	0	0	-	1/2
	0	0	0	0	0			$2^{-6.8}$

Table 6. Local Collision in Round 2/4 (f_1)

step	ΔA	ΔB	ΔC	ΔD	ΔE	Δw	s	probability
j	0	0	0	0	0	80000000	0	1
j+1	0	80000000	0	0	0	0	10	1
j+2	0	00000200	00000200	0	0	0	-	3/8
j+3	0	0	00080000	00000200	0	0	-	25/64
j+4	0	0	0	00080000	00000200	0	-	25/64
j+5	00000200	0	0	0	00080000	0	-	25/64
j+6	00080000	0	0	0	0	0	-	5/8
	0	0	0	0	0			$2^{-7.2}$

Table 7. Local Collision in Round 3 (f_2)

step	ΔA	ΔB	ΔC	ΔD	ΔE	Δw	s	probability
j	0	0	0	0	0	80000000	0	1
j+1	0	80000000	0	0	0	0	-	1/2
j+2	0	0	00000200	0	0	0	-	1/2
j+3	0	0	0	00000200	0	0	-	1/2
j+4	0	0	0	0	00000200	0	-	1/2
j+5	00000200	0	0	0	0	0	-	1/2
	0	0	0	0	0			2^{-5}

Hashing with Polynomials

Vladimir Shpilrain*

Department of Mathematics, The City College of New York, New York, NY 10031
shpil@groups.sci.ccny.cuny.edu

Abstract. In this paper, we explore potential mathematical principles and structures that can provide the foundation for cryptographic hash functions, and also present a simple and efficiently computable hash function based on a non-associative operation with polynomials over a finite field of characteristic 2.

1 Introduction

Hash functions are easy-to-compute compression functions that take a variable-length input and convert it to a fixed-length output. Hash functions are used as compact representations, or digital fingerprints, of data and to provide message integrity. Some hash functions in current use have been shown to be vulnerable. In [6], the author argues that their replacements should be based on a mathematical theory, which has yet to be created.

Before such a theory can be created, one has to describe, as detailed as possible, mathematical properties that a "good" hash function should have. Of course, basic requirements are well known:

1. *Preimage resistance* (sometimes called *non-invertibility*): it should be computationally infeasible to find an input which hashes to a specified output;
2. *Second pre-image resistance*: it should be computationally infeasible to find a second input that hashes to the same output as a specified input;
3. *Collision resistance*: it should be computationally infeasible to find two different inputs that hash to the same output.

Now the problem is to determine mathematical properties of a hash function that would ensure (or at least, make it likely) that the requirements above are met.

Early suggestions (especially the SHA family) did not really use any mathematical ideas apart from the Merkle-Damgard construction for producing collision-resistant hash functions from collision-resistant compression functions (see e.g. [7]); the main idea was just to "create a mess" by using complex iterations (this is not meant in a derogatory sense, but just as an opposite of using mathematical structure one way or another). We have to admit that a "mess" might be good for hiding purposes, but only to some extent. In particular, several early suggestions were successfully attacked, see e.g. [10]. However, the market has its own rules,

* Partially supported by the NSF grant DMS-0405105 and by an NSA grant.

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 22–28, 2006.
© Springer-Verlag Berlin Heidelberg 2006

and a product that has won the market is unlikely to be replaced by something altogether different; it is more likely that it will be slightly adjusted every time the older version becomes outdated for one reason or another. In other words, in 20 years from now, something like SHA-32768 is more likely to be commercially used than any hash function based on mathematical ideas. To be fair, we have to mention an interesting direction, namely, constructing hash functions being provably as secure as underlying assumptions, e.g. as discrete logarithm assumptions; see [4] and references therein. These hash functions however tend to be not very efficient. For a general survey of hash functions we refer to [7].

One especially discouraging example of the trend of ignoring elegant mathematical ideas is the Tillich-Zémor hash function [9], which is quite simple, efficient, and intelligent, and yet it was almost completely ignored; for example, Google search for this hash function produces about 200 results, compared to over 18,000,000 for SHA-1. This, by any standards, is not healthy, especially since at this time, there is no compelling reason to doubt the security of the Tillich-Zémor hash function. To the best of our knowledge, there are just 3 published papers [3], [5], [8] offering "attacks" (i.e., exposing collisions) on this hash function for some very special values of parameters. For "generic" values of parameters, the Tillich-Zémor hash function withstands all known attacks; in particular, it was shown in [1] that it is vulnerable to the attack of [3] with probability of (approximately) 10^{-27}. Of course, it is difficult to *prove* collision resistance of the Tillich-Zémor hash function, but this is true for most other hash functions, too. (We note that recently, a proposal for a hash function has been made [2] where collision resistance follows from the alleged hardness of a mathematical problem related to *expander graphs*; the authors of [2] acknowledge that one of their constructions is similar to that of Tillich and Zemor.)

In this paper, we try to somewhat remedy the situation. First, in Sections 2 and 3, we use the Tillich-Zémor hash function as a model example to analyze mathematical principles and structures behind a secure and efficient hash function. We speculate that a successful hash function should be based on a (finite) *dynamical system*, of which the Tillich-Zémor construction is a nice example.

Then, in Section 4, we present a simple and efficiently computable hash function which is based on essentially the same dynamical system as the Tillich-Zémor hash function is. In particular, it has the advantages of the Tillich-Zémor hash function, but does not have its (potential) weaknesses. We suggest specific parameters for our hash function in Section 5. However, we have to say up front that our proposal is on a conceptual level; in particular, while our hash function is obviously (by comparing the definitions) more efficient than that of Tillich-Zémor, we do not report actual runtimes here.

2 Tillich-Zémor Hash Function: Three Useful Features

The Tillich-Zémor hash function, unlike functions in the SHA family, is *not* a block hash function, i.e., each bit is hashed individually. More specifically, the "0" bit is hashed to a particular 2×2 matrix A, and the "1" bit is hashed to

another 2×2 matrix B. Then a bit string is hashed simply to the product of matrices A and B corresponding to bits in this string. For example, the bit string 1000110 is hashed to the matrix BA^3B^2A.

Obviously, this kind of arrangement is possible with *any* pair of elements A, B of *any* semigroup S. The question is: what choice of semigroup S and elements A, B makes the corresponding hash function secure? We argue here that the choice made by Tillich and Zémor in [9] has three useful features which are, in our opinion, significant for cryptographic security in general and for the security of a hash function in particular.

First we recall that Tillich and Zémor use matrices A, B from the group $SL_2(R)$, where R is a commutative ring (actually, a field) defined as $R = \mathbf{F}_2[x]/(p(x))$. Here $\mathbf{F}_2$ is the field with two elements, $\mathbf{F}_2[x]$ is the ring of polynomials over $\mathbf{F}_2$, and $(p(x))$ is the ideal of $\mathbf{F}_2[x]$ generated by an irreducible polynomial $p(x)$ of degree n (typically, n is a prime, $127 \leq n \leq 170$); for example, $p(x) = x^{131} + x^7 + x^6 + x^5 + x^4 + x + 1$. Thus, $R = \mathbf{F}_2[x]/(p(x))$ is isomorphic to $\mathbf{F}_{2^n}$, the field with 2^n elements.

Then, the matrices A and B are:

$$A = \begin{pmatrix} \alpha & 1 \\ 1 & 0 \end{pmatrix}, \qquad B = \begin{pmatrix} \alpha & \alpha + 1 \\ 1 & 1 \end{pmatrix},$$

where α is a root of $p(x)$.

Now the three useful features of the Tillich-Zémor hash function are:

1. The *commutativity* of the ring R is good for "diffusion", i.e., for hiding occurrences of A or B in a product.
2. The *periodicity* of the ring R, too, is good for "diffusion". (Periodicity means that for any $u \in R$, there is a positive integer m such that $u^m = u$.)
3. The non-commutativity of matrix multiplication prevents from obvious collisions. For example, were the multiplication commutative, the bit strings "01" and "10" would hash to the same thing.

We emphasize once again that

$$COMMUTATIVITY \qquad and \qquad PERIODICITY$$

are two major tools for hiding factors in a product; their importance for cryptographic security in general and for the security of a hash function in particular cannot be overestimated. Furthermore, a commutative and periodic platform gives rise to a *dynamical system*, where two functions (corresponding to the "0" and the "1" bits) act in a rather complex way. Dynamical systems with a large number of states, even "innocent-looking" ones, usually exhibit very complex behavior; it is sufficient to mention the notorious "3x+1" problem. In particular, any instances of re-occurring state are usually very difficult to predict (again, recall the "3x+1" problem). In the context of hash functions, those correspond to collisions, thus making the latter difficult to detect.

At the same time, for better security, commutativity might be reinforced by non-commutativity pretty much the same way as concrete is reinforced by steel to produce ferroconcrete. Thus,

$$COMMUTATIVITY \quad \text{in the corset of} \quad NON{-}COMMUTATIVITY$$

is another important ingredient of cryptographic security. It prevents the attacker from using obvious "relations", such as $ab = ba$, to simplify a product.

To conclude this section, we say a few words about the efficiency of the Tillich-Zémor hash function. Computing this function involves the following operations with polynomials over $\mathbf{F}_2$:

1. Multiplication and addition of polynomials in α of degrees bounded by that of the polynomial $p(x)$.
2. Division of a polynomial in α whose degree is at most twice the degree of $p(x)$, by the polynomial $p(\alpha)$.

These operations are quite efficient; in fact, their time complexity is bounded by a constant which depends only on (the degree of) the polynomial $p(x)$. Since the suggested degree of $p(x)$ is fairly small (see above), the constant in question is small, too.

3 Tillich-Zémor Hash Function: Two Potential Weaknesses

In this section, we discuss two features of the Tillich-Zémor hash function which may, in our opinion, yield undesirable trapdoors.

(i) The matrices A and B are *invertible*.
(ii) The operation (matrix multiplication) used in hashing is *associative*, i.e., $a(bc) = (ab)c$ for any $a, b, c \in SL_2(R)$.

These two features together may not be good from the security point of view, for the following reasons. First, suppose the intruder knows part of a hashed bitstring $\mathcal{S}$: say, $\mathcal{S} = \mathcal{S}_1\mathcal{S}_2$, and assume the intruder knows $\mathcal{S}_1$. Then, because of the property (ii) above, we have for the hashes:

$$H(\mathcal{S}) = H(\mathcal{S}_1) \cdot H(\mathcal{S}_2). \tag{1}$$

Since we assume that the intruder knows $\mathcal{S}_1$, he also knows $H(\mathcal{S}_1)$, and therefore, by using property (i) (invertibility of hashes) and the above equality, he can recover $H(\mathcal{S}_2)$, thereby making it somewhat easier to recover $\mathcal{S}_2$ and then $\mathcal{S}$.

Another, more serious, weakness implied by the associativity is the following. Suppose a collision is found, say, $H(\mathcal{T}_1) = H(\mathcal{T}_2)$ for some bitstrings $\mathcal{T}_1, \mathcal{T}_2$. Then the equality (1) above yields, for any bitstring $\mathcal{S}$: $H(\mathcal{S}T_1) = H(\mathcal{S}T_2)$. Thus, one collision easily yields many other.

To be fair, we have to point out (see e.g. [8]) that the property (ii) above is useful to make hashing more efficient because it allows one to parallelize computation, e.g. to compute the hash $H(\mathcal{S}_1\mathcal{S}_2)$ as $H(\mathcal{S}_1) \cdot H(\mathcal{S}_2)$.

Finally, we note that a fairly well understood group structure of $SL_2(F_{2^n})$ may eventually help to find semigroup relations between the matrices A and B, thus revealing a collision. In this sense, the absence of structure (i.e., a "mess") is something that can be borrowed from the SHA family in this context.

4 Hashing with Polynomials

In this section, we present a simple and efficiently computable hash function which has the advantages of the Tillich-Zémor hash function, but does not seem to have its (potential) weaknesses.

Let $R = \mathbf{F}_{2^n} = \mathbf{F}_2[x]/(p(x))$ and α be as in the Tillich-Zémor construction (see also our Section 2). Let $P(\alpha)$ and $Q(\alpha)$ be two elements of R; they are going to be hashes of the "0" and the "1" bit, respectively:

$$H(0) = P(\alpha), \qquad H(1) = Q(\alpha).$$

The hash of the concatenation $\mathcal{S}_1\mathcal{S}_2$ of two bitstrings is computed by the following recursive formula, *which is only applied in the situations described below (after the formula)*:

$$H(\mathcal{S}_1\mathcal{S}_2) = H(\mathcal{S}_1) \circ H(\mathcal{S}_2) = H(\mathcal{S}_1) \cdot H(\mathcal{S}_2) + (H(\mathcal{S}_1))^2 \cdot u_1(\alpha) + (H(\mathcal{S}_2))^2 \cdot u_2(\alpha) + v(\alpha),$$

where $u_i(\alpha)$ and $v(\alpha)$ are some fixed elements of R. We note that the operation $\circ$ is non-associative and non-commutative if $u_1(\alpha) \neq u_2(\alpha)$.

Now suppose $\mathcal{S}$ is a bitstring of length $n \geq 2$. To hash $\mathcal{S}$:

1. Split $\mathcal{S}$ into blocks $\mathcal{B}_1, \mathcal{B}_2, \ldots$ of length 32 going left to right (the rightmost block may therefore have smaller length).
2. Compute the hash of each block $\mathcal{B}_i$ independently, going left to right bit by bit and using the recursive formula above with $\mathcal{S}_2$ being a single bit every time.
3. Compute the hash of $\mathcal{S}$ inductively, going left to right block by block and using the recursive formula above with $\mathcal{S}_2$ being a single block $\mathcal{B}_i$ every time.

We emphasize once again that if $u_1(\alpha) \neq u_2(\alpha)$, then the operation $\circ$ defined above is non-associative and non-commutative. However, since the ring R itself is commutative (and periodic), we take full advantage of commutativity and periodicity as hiding tools here. In fact, our hash function is based on essentially the same dynamical system as the Tillich-Zémor hash function; we just get rid of the matrices to avoid associativity and invertibility. At the same time, since our operation $\circ$ is non-commutative, we have the "commutativity in the corset of non-commutativity" property that was discussed in our Section 2.

Thus, our hash function has the same advantages as the Tillich-Zémor hash function does. On the other hand, it does not have the weaknesses discussed

in our Section 3. Indeed, we have already mentioned that the operation $\circ$ is non-associative. It is also "non-invertible" in the following sense. If, for some bitstring $\mathcal{S} = \mathcal{S}_1\mathcal{S}_2$, you know $H(\mathcal{S})$ and $H(\mathcal{S}_1)$, this does not allow you to find $H(\mathcal{S}_2)$ the way it can be done for the Tillich-Zémor hash function.

Moreover, if you know $H(\mathcal{S}_1)$ and $H(\mathcal{S}_2)$, this does not help you, in general, to find $H(\mathcal{S}_1S_2)$, because the formula for $H(\mathcal{S}_1\mathcal{S}_2) = H(\mathcal{S}_1) \circ H(\mathcal{S}_2)$ is applicable only in very special cases of $\mathcal{S}_1$ and $\mathcal{S}_2$, see the definition above. This implies, in particular, that knowing one collision does not immediately yield any other, contrasting the situation with the Tillich-Zémor hash function (see the previous section).

5 Parameters

In this section, we suggest particular polynomials that can be used in the definition of a hash function given in the previous section. There is no specific motivation behind this particular choice of parameters; as with most dynamical systems, "generic" parameters yield sufficiently complex behavior of the system.

1. In the definition of $R = \mathbf{F}_{2^n} = \mathbf{F}_2[x]/(p(x))$, we suggest to take

$$p(x) = x^{163} + x^7 + x^6 + x^5 + x^4 + x + 1.$$

   Thus, any bitstring will be hashed to a polynomial of degree at most 162 over $\mathbf{F}_2$, which is equivalent to hashing to a 163-bit string.

2. In the definition of $H(0) = P(\alpha),\ H(1) = Q(\alpha)$, we suggest to take

$$H(0) = P(\alpha) = \alpha^7 + 1, \qquad H(1) = Q(\alpha) = \alpha^8 + 1.$$

3. In the definition of $H(\mathcal{S}_1S_2) = H(\mathcal{S}_1) \circ H(\mathcal{S}_2) = H(\mathcal{S}_1) \cdot H(\mathcal{S}_2) + (H(\mathcal{S}_1))^2 \cdot u_1(\alpha) + (H(\mathcal{S}_2))^2 \cdot u_2(\alpha) + v(\alpha)$, we suggest to take

$$u_1(\alpha) = \alpha^2,\ u_2(\alpha) = \alpha, \qquad v(\alpha) = 1.$$

   Thus, whenever applicable,

$$H(\mathcal{S}_1S_2) = H(\mathcal{S}_1) \cdot H(\mathcal{S}_2) + (H(\mathcal{S}_1))^2 \cdot \alpha^2 + (H(\mathcal{S}_2))^2 \cdot \alpha + 1.$$

   Below we give examples of computing $H(\mathcal{S})$ for some simple bitstrings $\mathcal{S}$.

1. $H(00) = H(0) \cdot H(0) + (H(0))^2 \cdot \alpha^2 + (H(0))^2 \cdot \alpha + 1 = \alpha^{16} + \alpha^{15} + \alpha^{14} + \alpha^2 + \alpha.$
2. $H(01) = H(0) \cdot H(1) + (H(0))^2 \cdot \alpha^2 + (H(1))^2 \cdot \alpha + 1 = \alpha^{17} + \alpha^{16} + \alpha^{15} + \alpha^8 + \alpha^7 + \alpha^2 + \alpha.$
3. $H(10) = H(1) \cdot H(0) + (H(1))^2 \cdot \alpha^2 + (H(0))^2 \cdot \alpha + 1 = \alpha^{18} + \alpha^8 + \alpha^7 + \alpha^2 + \alpha.$

4. $H(11) = H(1)\cdot H(1)+(H(1))^2\cdot\alpha^2+(H(1))^2\cdot\alpha+1 = \alpha^{18}+\alpha^{17}+\alpha^{16}+\alpha^2+\alpha.$

5. $H(001) = H(00)\cdot H(1)+(H(00))^2\cdot\alpha^2+(H(1))^2\cdot\alpha+1 =$
$= \alpha^{34}+\alpha^{32}+\alpha^{30}+\alpha^{24}+\alpha^{23}+\alpha^{22}+\alpha^{17}+\alpha^{16}+\alpha^{14}+\alpha^{10}+\alpha^9+\alpha^6+\alpha^4+\alpha^2+1.$

6. $H(010) = H(01)\cdot H(0)+(H(01))^2\cdot\alpha^2+(H(0))^2\cdot\alpha+1 =$
$\alpha^{36}+\alpha^{34}+\alpha^{32}+\alpha^{24}+\alpha^{23}+\alpha^{22}+\alpha^{18}+\alpha^{15}+\alpha^{14}+\alpha^9+\alpha^8+\alpha^7+\alpha^6+\alpha^4+\alpha^2+1.$

7. $H(110) = H(11)\cdot H(0)+(H(11))^2\cdot\alpha^2+(H(0))^2\cdot\alpha+1 =$
$\alpha^{38}+\alpha^{36}+\alpha^{34}+\alpha^{25}+\alpha^{24}+\alpha^{23}+\alpha^{18}+\alpha^{17}+\alpha^{16}+\alpha^{15}+\alpha^9+\alpha^8+\alpha^6+\alpha^4+\alpha^2+1.$

Acknowledgement. I am indebted to Rainer Steinwandt for numerous helpful discussions.

References

1. K. S. Abdukhalikov and C. Kim, *On the Security of the Hashing Scheme Based on SL_2*, in FSE 1998, Lecture Notes Comp. Sc. **1372** (1998), 93-102.
2. D. Charles, E. Goren, and K. Lauter, *Cryptographic hash functions from expander graphs*, preprint.
http://www.math.mcgill.ca/goren/PAPERSpublic/Hashfunction.pdf
3. C. Charnes and J. Pieprzyk, *Attacking the SL_2 hashing scheme*, in ASIACRYPT 1994, Lecture Notes Comp. Sc. **917** (1995), 322-330.
4. S. Contini, A. K. Lenstra and R. Steinfeld, *VSH, an Efficient and Provable Collision Resistant Hash Function*, in: Eurocrypt 2006, Lecture Notes Comp. Sc. **4004** (2006), 165–182.
5. W. Geiselmann, *A Note on the Hash Function of Tillich and Zémor*, in Cryptography and Coding, Lecture Notes Comp. Sc. **1025** (1995), 257-263.
6. S. Landau, *Find Me a Hash*, Notices Amer. Math. Soc. **53** (2006), 330–332.
7. A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
8. R. Steinwandt, M. Grassl, W. Geiselmann, T. Beth, *Weaknesses in the* $\mathrm{SL}_2(\mathbf{F}_{2^n})$ *Hashing Scheme*, in CRYPTO 2000, Lecture Notes Comp. Sc. **1880** (2000), 287–299.
9. J.-P. Tillich and G. Zémor, *Hashing with SL_2*, in CRYPTO 1994, Lecture Notes Comp. Sc. **839** (1994), 40–49.
10. X. Wang, Y. L. Yin, and H. Yu, *Finding Collisions in the Full SHA-1*, in CRYPTO 2005, Lecture Notes Comp. Sc. **3621** (2005), 17-36.

Birthday Paradox for Multi-collisions

Kazuhiro Suzuki[1], Dongvu Tonien[2], Kaoru Kurosawa[3], and Koji Toyota[3]

[1] Venture Business Laboratory, Ibaraki University 4–12–1 Nakanarusawa, Hitachi, Ibaraki 316-8511, Japan
[2] School of Information Technology and Computer Science, University of Wollongong, Wollongong 2522, Australia
[3] Department of Computer and Information Sciences, Ibaraki University 4–12–1 Nakanarusawa, Hitachi, Ibaraki 316-8511, Japan

Abstract. In this paper, we study multi-collision probability. For a hash function $H : D \to R$ with $|R| = n$, it has been believed that we can find an s-collision by hashing $Q = n^{(s-1)/s}$ times. We first show that this probability is at most $1/s!$ which is very small for large s. We next show that by hashing $(s!)^{1/s} \times Q$ times, an s-collision is found with probability approximately 0.5 for sufficiently large n. Note that if $s = 2$, it coincides with the usual birthday paradox. Hence it is a generalization of the birthday paradox to multi-collisions.

Keywords: hash function, birthday paradox, multi-collision, collision resistant.

1 Introduction

Let $H : D \to R$ be a hash function, where D is the domain and R is the range such that $|R| = n$. A collision for H is a distinct pair $x_1, x_2 \in D$ such that $H(x_1) = H(x_2)$. We usually require that H is collision resistant, which means that it is hard to find a collision. This security notion is used in many cryptographic applications such as digital signatures. All hash functions, however, suffer from the so-called birthday paradox which is a generic collision-finding attack. In this attack, we choose $x_1, \cdots, x_q \in D$ independently at random and compute $y_i = H(x_i)$ for $i = 1, \cdots, q$. We succeed if there is a pair i, j such that $H(x_i) = H(x_j)$. It is then well known that if $q = O(\sqrt{n})$, then we succeed with non-negligible probability (say, 0.5). Bellare, Kilian and Rogaway derived a nice upper bound and a lower bound on this success probability [1, Appendix].

Multi-collisions, on the other hand, are also an important notion of hash functions. An s-collision for H is s distinct points $x_1, \cdots, x_s \in D$ such that $H(x_1) = \cdots = H(x_s)$. As a negative side, Joux [5] showed a multi-collision attack on iterated hash functions at Crypto'04. As a positive side, the notion of multi-collisions was used for indentification schemes by Girault and Stern [4], for signature schemes by Brickell *et al.* [3] and for the micropayment scheme of Rivest and Shamir [6]. These schemes made use of an intuition such that finding an s-collision would be much harder than finding a usual collsion if s is large. Indeed, as a generalization of the birthday paradox, it has been believed that

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 29–40, 2006.
© Springer-Verlag Berlin Heidelberg 2006

"We can find an s-collision by hashing $q = n^{(s-1)/s}$ x-values"

as written in [6, Sec.4] [5, Sec.2].

In this paper, we first present a negative result which shows that the above sentence is wrong. More precisely, we prove that by hashing $Q = n^{(s-1)/s}$ x-values, an s-collision is found with probability at most $1/s!$. Note that this probability is very small if s is large. Hence the above sentence is wrong for large s.

We next show a positive result such that by hashing $q = (s!)^{1/s} \times Q$ x-values[1], an s-collision is found with probability approximately at least 0.5 for sufficiently large n. Note that if $s = 2$, it coincides with the usual birthday paradox. Hence we can consider that it is a generalization of the birthday paradox to multi-collisions.

Throughout this paper, we suppose that each image $y \in R$ has the same number of preimages, that is, $|H^{-1}(y)| = |D|/|R|$ for all $y \in R$. In Sec. 2, we present a recursive formula which expresses the exact probability of finding an s-collision. In Sec. 3, we present a general lower and an upper bound of the probability of finding an s-collision. In Sec. 4, we show a more tight lower and an upper bound which agree within a constant factor for $q \leq n^{(s-1)/s}$. In Sec. 5, we show our main (negative and positive) results for $q = O(n^{(s-1)/s})$.

2 Exact Probability of s-Collision

In this section, we present a recursive formula for the probability of s-collision. We will use this formula to find the exact value and to derive bounds for the probability.

Let $2 \leq s \leq q \leq n$ and consider the following experiment. Suppose that there are q balls and n buckets. We throw the balls one by one at random into the buckets. Let $C(n, q, s)$ denote the event (called *s-collision*) that there exists at least one bucket that contains at least s balls.

The above experiment mimics the generic hashing attack as follows. We call n elements of the set R buckets. The q random x-values $x_1, \ldots, x_q$ are called balls. Each time we calculate the hash value $H(x_i)$, we imagine that the ball x_i is thrown into the bucket $H(x_i)$. If a bucket r contains at least s balls, say $x_{i_1}, \ldots, x_{i_s}$, then we have found an s-collision $H(x_{i_1}) = \ldots = H(x_{i_s}) = r$. Thus, the probability $Pr[C(n, q, s)]$ models the s-collision probability.

We now present a recursive formula of $Pr[C(n, q, s)]$.

Theorem 1

$$Pr[C(n,q,s)] = \frac{1}{n^{s-1}} \sum_{i=s}^{q} \binom{i-1}{s-1} \left(1 - \frac{1}{n}\right)^{i-s} (1 - Pr[C(n-1, i-s, s)]).$$

Proof. In the experiment of throwing q balls one by one at random into n buckets, for each $s \leq i \leq q$, let $C(n, q, s, i)$ denote the event that the i^{th} ball causes the

[1] Approximately, $q \approx (s/2.71) \times n^{(s-1)/s}$ from Stirling formula.

first s-collision, that is, s-collision does not occur until the i^{th} ball but does when the ball is thrown. Then

$$Pr[C(n,q,s)] = \sum_{i=s}^{q} Pr[C(n,q,s,i)].$$

We can find $Pr[C(n,q,s,i)]$ as follows:

1. One bucket (denoted by B), where the first s-collision occurs, can be selected from n buckets in n ways;
2. $s-1$ balls, which are put into B can be selected from the previous $i-1$ balls in $\binom{i-1}{s-1}$ ways;
3. The probability that the s selected balls land in the one selected bucket is $1/n^s$;
4. The probability that for the s selected balls and the one selected bucket B, none of the other $i-s$ balls land in B and cause an s-collision is $(1-1/n)^{i-s} \times (1 - Pr[C(n-1,i-s,s)])$.

Thus we have

$$\begin{aligned} Pr[C(n,q,s,i)] &= n \times \binom{i-1}{s-1} \times \frac{1}{n^s} \times \left(1-\frac{1}{n}\right)^{i-s} \times (1 - Pr[C(n-1,i-s,s)]) \\ &= \frac{1}{n^{s-1}} \binom{i-1}{s-1} \left(1-\frac{1}{n}\right)^{i-s} (1 - Pr[C(n-1,i-s,s)]). \end{aligned}$$

Therefore,

$$Pr[C(n,q,s)] = \frac{1}{n^{s-1}} \sum_{i=s}^{q} \binom{i-1}{s-1} \left(1-\frac{1}{n}\right)^{i-s} (1 - Pr[C(n-1,i-s,s)]). \quad \blacksquare$$

We will use this recursive formula to calculate the exact value of the s-collision probability and derive its bounds in the next sections. Before doing that we need some auxiliary results. The proofs are shown in the Appendix.

Lemma 1. *The following statements must hold*

1. *For any positive integers k, s, and $i \geq (k+1)s$,*

$$\sum_{i=s}^{q} \binom{i-1}{s-1} = \binom{q}{s}.$$

2. *For any positive integers k, s, and $i \geq (k+1)s$,*

$$\binom{i-1}{ks-1}\binom{i-ks}{s} = \binom{(k+1)s-1}{s}\binom{i-1}{(k+1)s-1}.$$

3. *For any positive integers $k \geq 2$, s, and $q \geq ks$,*

$$\binom{ks-1}{s}\binom{q}{ks} = \frac{k-1}{k}\binom{q}{s}\binom{q-s}{(k-1)s}.$$

4. *For any integers* $n, s \geq 2$,
$$(n-1)^{s-1} > (n^{\frac{s-1}{s}} - 1)^s.$$
5. *For any* $1 < a \leq b$,
$$\frac{a-1}{b-1} \leq \frac{a}{b}$$
6. *Let* $e_k = (1-1/k)^{-k}$ *then* $\{e_k\}_{k=2}^{\infty}$ *is a decreasing sequence and* $\lim_{k\to\infty} e_k = e \approx 2.7$ *– the Euler constant. For any* $0 < x < 1$, *we have*
$$e_k^{-x} > 1 - x \ln e_k \approx 1 - x.$$
7. *For any integer* $s \geq 2$,
$$(s!)^{-1/s}(s+1)/2 > 1.$$

3 Bounds on the Probability of s-Collision

In this section, we present the following bounds on the probability of s-collision.

Theorem 2
$$Pr[C(n,q,s)] \leq \frac{1}{n^{s-1}}\binom{q}{s},$$
and
$$Pr[C(n,q,s)] \geq \frac{1}{n^{s-1}}\binom{q}{s}\left(1-\frac{1}{n}\right)^{q-s}\left\{1 - \frac{1}{2(n-1)^{s-1}}\binom{q-s}{s}\right\}.$$

Proof. By Theorem 1 and Lemma 1(1), we obtain the upper bound
$$\begin{aligned} Pr[C(n,q,s)] &= \frac{1}{n^{s-1}}\sum_{i=s}^{q}\binom{i-1}{s-1}\left(1-\frac{1}{n}\right)^{i-s}(1 - Pr[C(n-1,i-s,s)]) \\ &\leq \frac{1}{n^{s-1}}\sum_{i=s}^{q}\binom{i-1}{s-1} = \frac{1}{n^{s-1}}\binom{q}{s}. \end{aligned}$$

We have
$$\begin{aligned} Pr[C(n,q,s)] &= \frac{1}{n^{s-1}}\sum_{i=s}^{q}\binom{i-1}{s-1}\left(1-\frac{1}{n}\right)^{i-s}(1 - Pr[C(n-1,i-s,s)]) \\ &\geq \frac{1}{n^{s-1}}\left(1-\frac{1}{n}\right)^{q-s}\sum_{i=s}^{q}\binom{i-1}{s-1}(1 - Pr[C(n-1,i-s,s)]) \\ &= \frac{1}{n^{s-1}}\left(1-\frac{1}{n}\right)^{q-s}\left[\sum_{i=s}^{q}\binom{i-1}{s-1} - \sum_{i=s}^{q}\binom{i-1}{s-1}Pr[C(n-1,i-s,s)]\right] \\ &= \frac{1}{n^{s-1}}\left(1-\frac{1}{n}\right)^{q-s}\left[\binom{q}{s} - \sum_{i=2s}^{q}\binom{i-1}{s-1}Pr[C(n-1,i-s,s)]\right] \end{aligned}$$

where the last equality follows from the fact that $Pr[C(n-1, i-s, s)] = 0$ for $i \leq 2s-1$ and Lemma 1(1).

Now using the above upper bound, we derive the lower bound,

$$Pr[C(n,q,s)] \geq \frac{1}{n^{s-1}} \left(1-\frac{1}{n}\right)^{q-s} \left[\binom{q}{s} - \sum_{i=2s}^{q} \binom{i-1}{s-1} \frac{1}{(n-1)^{s-1}} \binom{i-s}{s}\right]$$
$$= \frac{1}{n^{s-1}} \left(1-\frac{1}{n}\right)^{q-s} \left[\binom{q}{s} - \frac{1}{(n-1)^{s-1}} \sum_{i=2s}^{q} \binom{i-1}{s-1}\binom{i-s}{s}\right].$$

By Lemma 1(2),

$$= \frac{1}{n^{s-1}} \left(1-\frac{1}{n}\right)^{q-s} \left[\binom{q}{s} - \frac{1}{(n-1)^{s-1}} \binom{2s-1}{s} \sum_{i=2s}^{q} \binom{i-1}{2s-1}\right].$$

By Lemma 1(1),

$$= \frac{1}{n^{s-1}} \left(1-\frac{1}{n}\right)^{q-s} \left[\binom{q}{s} - \frac{1}{(n-1)^{s-1}} \binom{2s-1}{s}\binom{q}{2s}\right].$$

By Lemma 1(3),

$$= \frac{1}{n^{s-1}} \left(1-\frac{1}{n}\right)^{q-s} \left[\binom{q}{s} - \frac{1}{2(n-1)^{s-1}} \binom{q}{s}\binom{q-s}{s}\right]$$
$$= \frac{1}{n^{s-1}} \left(1-\frac{1}{n}\right)^{q-s} \binom{q}{s} \left\{1 - \frac{1}{2(n-1)^{s-1}} \binom{q-s}{s}\right\}. \quad \blacksquare$$

From now on, we use the following notation,

$$f(n) = \left(1-\frac{1}{n}\right)^{q-s} \quad \text{and} \quad g(n) = \frac{1}{2(n-1)^{s-1}} \binom{q-s}{s}.$$

Theorem 2 can be rewritten as

$$f(n)(1-g(n)) \frac{1}{n^{s-1}} \binom{q}{s} \leq Pr[C(n,q,s)] \leq \frac{1}{n^{s-1}} \binom{q}{s}. \tag{1}$$

4 Bounds for $q = \Theta(n^{\epsilon})$ Where $\epsilon < (s-1)/s$

The graph in Figure 1 demonstrates the upper bound and the lower bound in Theorem 2 and the exact probability of $Pr[C(n,q,s)]$ for $n = 365$ and $s = 3$. From this figure, we can see that for $q < n^{(s-1)/s} \approx 52$, the difference between values of these three graphs is small. We will show that when $q = n^{\epsilon}$ with $\epsilon < \frac{s-1}{s}$, the upper bound and the lower bound are indeed very close to each other. We also show that in this case, the upper bound asymptotically tends to zero.

Theorem 3. *Let ϵ be a positive number such that $\epsilon < \frac{s-1}{s}$. Then for any positive number $c < 1$, there exists a positive number n_0 such that*

$$c \times \frac{1}{n^{s-1}} \binom{q}{s} < Pr[C(n,q,s)] \leq \frac{1}{n^{s-1}} \binom{q}{s},$$

for any $n > n_0$ and $2 \leq s \leq q = n^{\epsilon}$.

Proof. The theorem follows from the following two claims.
Claim 1.

$$g(n) < \frac{1}{2\ s!} \frac{q^s}{n^{s-1}} = \frac{1}{2\ s!\ n^{s-1-s\epsilon}},$$

thus, $g(n) \to 0$ when $n \to \infty$.
Proof. We have

$$\binom{q-s}{s} = \frac{(q-s)(q-s-1)\ldots(q-2s+1)}{s!} < \frac{(q-1)^s}{s!},$$

By Lemma 1(4),

$$(n-1)^{s-1} > (n^{\frac{s-1}{s}} - 1)^s.$$

Thus,

$$g(n) = \frac{1}{2(n-1)^{s-1}} \binom{q-s}{s} < \frac{1}{2(n^{\frac{s-1}{s}} - 1)^s} \frac{(q-1)^s}{s!} = \frac{1}{2\ s!} \left(\frac{q-1}{n^{\frac{s-1}{s}} - 1} \right)^s$$

By Lemma 1(5),

$$g(n) < \frac{1}{2\ s!} \left(\frac{q}{n^{\frac{s-1}{s}}} \right)^s = \frac{1}{2\ s!} \frac{q^s}{n^{s-1}} = \frac{1}{2\ s!\ n^{s-1-s\epsilon}}.$$

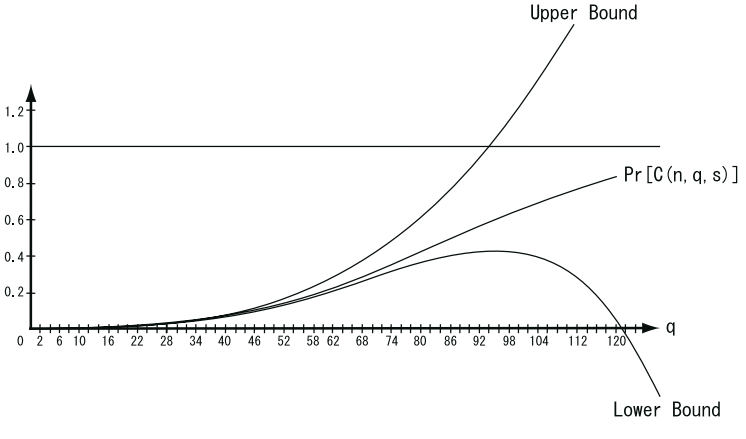


Fig. 1. The upper bound and the lower bound of Theorem 2 and the exact probability of $Pr[C(n,q,s)]$ for $n = 365$ and $s = 3$. We use the recursive formula in Section 2 to calculate the exact probability $Pr[C(n,q,s)]$.

Since $s-1-s\epsilon > 0$, we have $g(n) \to 0$ when $n \to \infty$.
Claim 2. With the notation in Lemma 1(6), for any $n > k$,

$$f(n) > e_k^{-q/n} = e_k^{-n^{\epsilon-1}},$$

where $e_k \approx e$, thus, $f(n) \to 1$ when $n \to \infty$.
Proof. We have

$$f(n) = \left(1-\frac{1}{n}\right)^{q-s} > \left(1-\frac{1}{n}\right)^{q} = \left[\left(1-\frac{1}{n}\right)^{-n}\right]^{-q/n} = e_n^{-q/n}.$$

Since $n > k$, by Lemma 1(6), $e_n < e_k$, thus,

$$f(n) > e_k^{-q/n} = e_k^{-n^{\epsilon-1}}.$$

Since $\epsilon < 1$, $n^{\epsilon-1} \to 0$ and $f(n) \to 1$ as $n \to \infty$.

From *Claim 1* and *Claim 2*, we have $f(n)(1-g(n)) \to 1$, thus, the theorem follows. ∎

Example. Let $s = 4$, $\epsilon = \frac{1}{2} < \frac{s-1}{s} = \frac{3}{4}$, and $n > 100$ then

$$g(n) < \frac{n^{s\epsilon-(s-1)}}{2\ s!} = \frac{n^{-1}}{48} < \frac{1}{4800} = 0.000208333,$$

$$f(n) > e_{100}^{-n^{\epsilon-1}} = e_{100}^{-n^{-1/2}} > e_{100}^{-100^{-1/2}} = \left[(1-1/100)^{-100}\right]^{-100^{-1/2}} > .9$$

Thus $f(n)(1-g(n)) > .8998$, and we have

$$.8998 \times \frac{1}{n^{s-1}}\binom{q}{s} < Pr[C(n,q,s)] \le \frac{1}{n^{s-1}}\binom{q}{s}.$$ ∎

Even though Theorem 3 shows that the upper bound and the lower bound are very closed to each other, the following lemma shows that these bounds asymptotically tend to zero.

Lemma 2. *Let ϵ be a positive number such that $\epsilon < \frac{s-1}{s}$ and $q = n^{\epsilon}$, then*

$$\frac{1}{n^{s-1}}\binom{q}{s} \to 0 \quad \text{when } n \to \infty.$$

Proof. We have

$$\frac{1}{n^{s-1}}\binom{q}{s} < \frac{1}{n^{s-1}}\frac{q^s}{s!} = \frac{1}{s!\ n^{s-1-s\epsilon}}.$$

Since $s-1-s\epsilon > 0$, we have

$$\frac{1}{n^{s-1}}\binom{q}{s} \to 0.$$ ∎

5 Bounds for $q = \Theta(n^{(s-1)/s})$

In this section, we consider the case $q = \Theta(n^{(s-1)/s})$. We prove two main theorems. Theorem 4 shows that if $q \approx n^{(s-1)/s}$ and n is sufficiently large then $Pr[C(n,q,s)] \approx 1/s!$, and Theorem 5 shows that if $q \approx (s!)^{1/s} n^{(s-1)/s}$ and n is sufficiently large then $Pr[C(n,q,s)] \gtrapprox 1/2$.

It implies the following *generalized birthday paradox*

> *For a hash function $H : D \to R$ with $|R| = n$, if n is sufficiently large then by $n^{(s-1)/s}$ number of hashings, an s-collision can be found with probability $\approx 1/s!$, and by $(s!)^{1/s}\, n^{(s-1)/s}$ number of hashings an s-collision can be found with probability $\gtrapprox 1/2$.*

Theorem 4. *We suppose that $q = \alpha\, n^{(s-1)/s}$, $q - s = \alpha'\, n^{(s-1)/s}$, where $0 < \alpha' < \alpha < 1$. If $2 \le s \le q$ then*

$$Pr[C(n,q,s)] \le \frac{1}{n^{s-1}} \binom{q}{s} < \frac{\alpha^s}{s!} < \frac{1}{s!} \tag{2}$$

and

$$Pr[C(n,q,s)] > \frac{\alpha'^s}{s!} - \left(\frac{\alpha'^{s+1} \ln e_n}{s!\, n^{1/s}} + \frac{(\alpha\alpha')^s}{2(s!)^2} \right)$$

where $e_n = (1 - 1/n)^{-n} \approx e$. In particular, if n is sufficiently large so that $1/n^{1/s} \approx 0$, and $\alpha' \lessapprox \alpha \lessapprox 1$, then we have

$$Pr[C(n,q,s)] > \frac{\alpha'^s}{s!} - \left(\frac{\alpha'^{s+1} \ln e_n}{s!\, n^{1/s}} + \frac{(\alpha\alpha')^s}{2(s!)^2} \right) \approx \frac{1}{s!} - \frac{1}{2(s!)^2}$$

Proof. We have

$$\frac{1}{n^{s-1}} \binom{q}{s} = \frac{1}{n^{s-1}} \frac{q(q-1)\dots(q-s+1)}{s!} < \frac{1}{n^{s-1}} \frac{q^s}{s!} = \frac{\alpha^s}{s!}$$

thus

$$Pr[C(n,q,s)] \le \frac{1}{n^{s-1}} \binom{q}{s} < \frac{\alpha^s}{s!} < \frac{1}{s!}.$$

We have

$$\frac{1}{n^{s-1}} \binom{q}{s} = \frac{1}{n^{s-1}} \frac{q(q-1)\dots(q-s+1)}{s!} > \frac{1}{n^{s-1}} \frac{(q-s)^s}{s!} = \frac{\alpha'^s}{s!}.$$

As in the proof of Theorem 3, we have

$$g(n) < \frac{1}{2\ s!} \frac{q^s}{n^{s-1}} = \frac{\alpha^s}{2\ s!}.$$

and by Lemma 1(6),

$$f(n) = e_n^{-(q-s)/n} > 1 - \frac{q-s}{n} \ln e_n = 1 - \frac{\alpha' \ln e_n}{n^{1/s}}.$$

Thus,

$$f(n)(1-g(n)) \geq f(n) - g(n) > 1 - \frac{\alpha' \ln e_n}{n^{1/s}} - \frac{\alpha^s}{2\ s!}.$$

Therefore,

$$\begin{aligned} Pr[C(n,q,s)] &\geq f(n)(1-g(n))\ \frac{1}{n^{s-1}}\binom{q}{s} \\ &> \left(1 - \frac{\alpha' \ln e_n}{n^{1/s}} - \frac{\alpha^s}{2\ s!}\right)\frac{\alpha'^s}{s!} \\ &= \frac{\alpha'^s}{s!} - \left(\frac{\alpha'^{s+1} \ln e_n}{s!\, n^{1/s}} + \frac{(\alpha\alpha')^s}{2(s!)^2}\right). \end{aligned}$$

■

Theorem 5. *If $2 \leq s \leq q$, and $q = (s!)^{1/s} n^{(s-1)/s} + s - 1 (< n)$, then we have*

$$Pr[C(n,q,s)] > \frac{1}{2} - \left(\frac{s!}{n}\right)^{1/s} \ln e_n.$$

In particular, if n is sufficiently large so that $(s!/n)^{1/s} \approx 0$, then we have

$$Pr[C(n,q,s)] > \frac{1}{2} - \left(\frac{s!}{n}\right)^{1/s} \ln e_n \approx \frac{1}{2}. \tag{3}$$

Proof. By Cauchy's inequality,

$$\begin{aligned} \binom{q-s}{s} &= \frac{(q-s)(q-s-1)\ldots(q-2s+1)}{s!} \\ &< \frac{1}{s!}\left(\frac{(q-s)+(q-s-1)+\ldots+(q-2s+1)}{s}\right)^s \\ &= \frac{[q-(3s-1)/2]^s}{s!} = \frac{[(s!)^{1/s}\, n^{(s-1)/s} - (s+1)/2]^s}{s!} \\ &= [n^{(s-1)/s} - (s!)^{-1/s}(s+1)/2]^s \end{aligned}$$

By Lemma 1(7), $(s!)^{-1/s}(s+1)/2 > 1$, thus,

$$\binom{q-s}{s} < (n^{(s-1)/s} - 1)^s.$$

By Lemma 1(4),

$$(n-1)^{s-1} > (n^{(s-1)/s} - 1)^s,$$

thus,

$$g(n) = \frac{1}{2(n-1)^{s-1}}\binom{q-s}{s} < \frac{1}{2}. \tag{4}$$

We have

$$f(n) = \left(1 - \frac{1}{n}\right)^{q-s} > \left(1 - \frac{1}{n}\right)^{q-s+1} = e_n^{-(q-s+1)/n},$$

thus, by Lemma 1(6),

$$f(n) > e_n^{-(q-s+1)/n} > 1 - \frac{q-s+1}{n} \ln e_n = 1 - \left(\frac{s!}{n}\right)^{1/s} \ln e_n. \tag{5}$$

From (4) and (5), we have

$$f(n)(1-g(n)) \geq f(n) - g(n) > \frac{1}{2} - \left(\frac{s!}{n}\right)^{1/s} \ln e_n. \tag{6}$$

We have

$$\frac{1}{n^{s-1}}\binom{q}{s} > \frac{(q-s+1)^s}{s!\ n^{s-1}} = \frac{((s!)^{1/s} n^{(s-1)/s})^s}{s!\ n^{s-1}} = 1. \tag{7}$$

Combining (6) and (7) gives

$$Pr[C(n,q,s)] \geq f(n)(1-g(n))\frac{1}{n^{s-1}}\binom{q}{s} > \frac{1}{2} - \left(\frac{s!}{n}\right)^{1/s} \ln e_n. \quad \blacksquare$$

Example. If $s \geq 2$, $n > s!\,32^s (\geq 2048)$ and $q = (s!)^{1/s} n^{(s-1)/s} + s - 1 (< n)$ then

$$\left(\frac{s!}{n}\right)^{1/s} < \frac{1}{32} \quad \text{and } \ln e_{2048} < 1.00025,$$

thus

$$Pr[C(n,q,s)] > \frac{1}{2} - \frac{1}{32} \times 1.00025 > .4687 \quad \blacksquare$$

6 Conclusion

In this paper, we have studied multi-collision probabilities for regular hash functions $H : D \to R$, where "regular" means that each image $y \in R$ has the same number of preimages. Suppose that that $|R| = n$. Then our main results are summarized as follows.

- By hashing about $n^{(s-1)/s}$ times, an s-collision is found with probability at most $1/s!$ (see eq.(2)). Since it is very small for large s, this disproves the folklore which has been believed so far.

- By hashing about $(s!)^{1/s} n^{(s-1)/s}$ times, an s-collision is found with probability approximately $1/2$ or more if n is large enough so that $(s!/n)^{1/s} \approx 0$ (see eq.(3)). Hence this is a true generalization of the birthday paradox to multicollisions.

Bellare and Kohno genralized the birthday paradox (for $s = 2$) to non-regular hash functions [2]. It will be a furter work to generalize our result on multicollision to non-regular hash functions.

Acknowledgement

The approximation of the footnote of Sec.1 was pointed out by an anonymous reviewer.

References

1. M.Bellare, J.Kilian and P.Rogaway: The Security of the Cipher Block Chaining Message Authentication Code. J. Comput. Syst. Sci. 61(3): pp.362–399 (2000)
2. M.Bellare and T.Kohno: Hash Function Balance and its Impact on Birthday Attacks. EUROCRYPT 2004, pp.401–418 (2004)
3. E.Brickell, D.Pointcheval, S.Vaudenay and M.Yung: Design Validations for Discrete Logarithm Based Signature Schemes. Public Key Cryptography 2000: pp.276–292 (2000)
4. M.Girault and J.Stern: On the Length of Cryptographic Hash-Values Used in Identification Schemes. CRYPTO 1994: pp.202–215 (1994)
5. Antoine Joux: Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. CRYPTO 2004: 306-316 (2004)
6. R.Rivest and A.Shamir: PayWord and MicroMint: Two Simple Micropayment Schemes. Security Protocols Workshop 1996: pp.69–87 (1996)

Appendix: Proofs of Lemma 1

Proof. (1) Since

$$\binom{i}{s} = \binom{i-1}{s} + \binom{i-1}{s-1},$$

we have

$$\sum_{i=s}^{q} \binom{i-1}{s-1} = 1 + \sum_{i=s+1}^{q} \binom{i-1}{s-1} = 1 + \sum_{i=s+1}^{q} \left[\binom{i}{s} - \binom{i-1}{s} \right] = 1 + \binom{q}{s} - \binom{s}{s} = \binom{q}{s}.$$

(2) We have

$$\begin{aligned} \binom{i-1}{ks-1}\binom{i-ks}{s} &= \frac{(i-1)!}{(ks-1)!(i-ks)!} \times \frac{(i-ks)!}{s!(i-(k+1)s)!} \\ &= \frac{((k+1)s-1)!}{s!(ks-1)!} \times \frac{(i-1)!}{((k+1)s-1)!(i-(k+1)s)!} \\ &= \binom{(k+1)s-1}{s}\binom{i-1}{(k+1)s-1}. \end{aligned}$$

(3) We have

$$\binom{ks-1}{s}\binom{q}{ks} = \frac{(ks-1)!}{s!((k-1)s-1)!} \times \frac{q!}{(ks)!(q-ks)!}$$

$$= \frac{q!}{(ks)\ s!\ ((k-1)s-1)!\ (q-ks)!}$$

$$= \frac{k-1}{k} \times \frac{q!}{s!(q-s)!} \times \frac{(q-s)!}{((k-1)s)!(q-ks)!}$$

$$= \frac{k-1}{k}\binom{q}{s}\binom{q-s}{(k-1)s}.$$

(4) Let $0 < t = \frac{s-1}{s} < 1$ and consider the function $a(n) = (n-1)^t - n^t + 1$. We have $a'(n) = t[(n-1)^{t-1} - n^{t-1}] > 0$. Thus, $a(n) \geq a(2) = 2 - 2^t > 0$. Therefore,

$$(n-1)^{\frac{s-1}{s}} > n^{\frac{s-1}{s}} - 1,$$

and thus,

$$(n-1)^{s-1} > (n^{\frac{s-1}{s}} - 1)^s.$$

(5) We have

$$\frac{a-1}{b-1} \leq \frac{a}{b} \leftrightarrow b(a-1) \leq a(b-1) \leftrightarrow a \leq b.$$

(6) It is a basic result that the sequence $\{e_k\}_{k=2}^{\infty}$ is a decreasing sequence, $e_k > e$, and $\lim_{k\to\infty} e_k = e$, the proof of this result can be found in any calculus textbook. We have

$$e^{-x} > 1 - x,$$

thus,

$$\begin{aligned} e_k^{-x} = (e^{-x})^{\ln e_k} &> (1-x)^{\ln e_k}, \text{ and by Bernoulli's inequality,} \\ &> 1 - x \ln e_k. \end{aligned}$$

(7) By Cauchy's inequality,

$$s! < \left(\frac{1+2+\ldots+s}{s}\right)^s = \left(\frac{s+1}{2}\right)^s,$$

thus, $(s+1)/2 > (s!)^{1/s}$. It follows that $(s!)^{-1/s}\,(s+1)/2 > 1$. ∎

New Variant of the Self-Shrinking Generator and Its Cryptographic Properties

Ku-Young Chang[1], Ju-Sung Kang[2], Mun-Kyu Lee[3,⋆,⋆⋆], Hangrok Lee[1], and Dowon Hong[1]

[1] Electronics and Telecommunications Research Institute
161 Gajeong-dong, Yuseong-gu, Daejeon 305-350, Korea
[2] Department of Mathematics, Kookmin University
Jeongreung3-Dong, Seongbuk-Gu, Seoul 136-702, Korea
[3] School of Computer Science and Engineering
Inha University, Incheon 402-751, Korea
mklee@inha.ac.kr

Abstract. We propose a variant of the self-shrinking generator, which is constructed using an extended selection rule. This new generator is called SSG-XOR since the selection rule is determined by the XORed value of a couple of bits. It is shown that the period and the linear complexity of an output sequence of SSG-XOR are better than those of the self-shrinking generator. It is also shown that the SSG-XOR generator has meaningful advantages from the viewpoint of practical cryptanalysis.

1 Introduction

The shrinking generator [2] and the self-shrinking generator [7] are attractive since they have very simple structures and good cryptographic properties. These features of the shrinking and the self-shrinking generators make them suitable for use in light-weight and low-cost stream ciphers. So it is worthwhile to examine cryptographic properties of these generators. As for the shrinking generator, Coppersmith, Krawczyk and Mansour [2] have shown that the period and the linear complexity of output sequences are exponential in the length of the linear feedback shift registers (LFSRs) which constitute the generator, and that these sequences have some nice distributional statistics. On the other hand, Meier and Staffelbach [7] have analyzed the period and the linear complexity of output sequences of self-shrinking generator. Although they proved only the lower bounds on period and linear complexity, they showed that the real values obtained from their experiments are sufficiently large.

The self-shrinking generator, which uses only one LFSR, has a simpler structure than the shrinking generator, which requires two LFSRs. Also, it is known that the self-shrinking generator is even more resistant to cryptanalysis than

⋆ Corresponding author.

⋆⋆ This work was supported by grant No.R01-2006-000-10957-0 from the Basic Research Program of the Korea Science & Engineering Foundation.

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 41–50, 2006.
© Springer-Verlag Berlin Heidelberg 2006

the shrinking generator. Thus the self-shrinking generator is more attractive for light-weight and low-cost implementations of stream ciphers.

In this paper we present a variant of the self-shrinking generator which will be called the self-shrinking generator with XOR, or briefly SSG-XOR. Then we show that SSG-XOR has better cryptographic properties than the self-shrinking generator in a practical setting, while it can be implemented with the efficiency similar to the self-shrinking generator. Specifically, we show that an output sequence of SSG-XOR is balanced and the period and the linear complexity of this sequence are twice longer than those of the self-shrinking generator. Also, it is shown that SSG-XOR is more secure than the self-shrinking generator against various previous known attacks using short known keystream sequences.

The remainder of this paper is organized as follows. In Section 2, we describe our new keystream generator, SSG-XOR. In Sections 3 and 4, we analyze the period and the linear complexity of SSG-XOR, respectively. Then we examine possible attacks on SSG-XOR in Section 5. Finally, we provide some concluding remarks in Section 6.

2 Extended Self-Shrinking Generator Using XOR

The shrinking generator (SG) is a well-known keystream generator proposed by Coppersmith, Krawczyk and Mansour [2]. Later, Meier and Staffelbach [7] proposed a modified version of shrinking generator called the self-shrinking generator (SSG). Let A and S be maximum length LFSRs generating the m-sequence $(a_i)_{i\geq 0}$ and $(s_i)_{i\geq 0}$, respectively. For each clock i, SG outputs a_i if $s_i = 1$. Otherwise, it discards a_i. On the other hand, SSG consists of only one LFSR A. For any bit pair (a_{2i}, a_{2i+1}), SSG outputs a_{2i+1} if $a_{2i} = 1$. Otherwise, it discards a_{2i+1}.

SSG has many features similar to SG. Meier and Staffelbach [7] have shown that SSG can be implemented using an SG with two LFSRs having identical feedback connections and SG can be implemented as a special case of SSG. Although SSG is closely related to SG, it has better cryptographic properties than SG. First, the keystreams of SSG are balanced, while those of SG are not. Next, SSG is more secure than SG against various known attacks as follows. Recently, Simpson, Golić and Dawson [9] and Golić [4] showed that SG is not secure against the probabilistic correlation attack, and Ekdahl, Meier and Johansson [3] proposed a practical distinguishing attack on SG. However, SSG has not revealed any weakness for these attacks yet. For the present, the most efficient attack against SSG for a known short keystream is the BDD-based attack [5], which requires $n^{O(1)}2^{0.6563n}$ computational steps for an m-LFSR of length n.

In this paper, we propose a new keystream generator called SSG-XOR, which is an extension of SSG. SSG-XOR consists of only one LFSR and its structure is as follows. Let $\widetilde{a} = (a_i)_{i\geq 0}$ be an output sequence of a non-trivially initialized m-LFSR of length n. Consider the sequence $\widetilde{a} = (a_i)_{i\geq 0}$ as a sequence of 4-tuples of bits $((a_0, a_1, a_2, a_3), (a_4, a_5, a_6, a_7), \ldots)$. For each 4-tuple $(a_{4i}, a_{4i+1}, a_{4i+2}, a_{4i+3})$, SSG-XOR outputs two bits a_{4i+2} and a_{4i+3}

if $a_{4i} \oplus a_{4i+1} = 1$, and discard a_{4i+2} and a_{4i+3} otherwise. The close relationship between SSG and SSG-XOR is shown in figure 1.

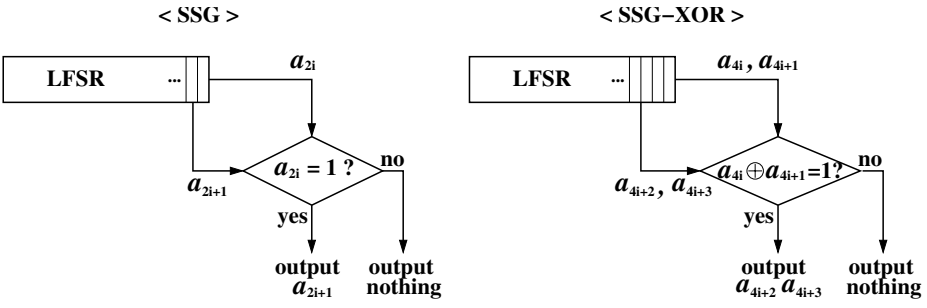


Fig. 1. SSG and SSG-XOR

Note that SSG requires four clocks to generate one bit of output stream on the average, while SSG-XOR requires eight clocks to generate two bits of output stream. Thus, SSG and SSG-XOR have the similar efficiency. However it will be shown in Sections 3 through 5 that SSG-XOR has better cryptographic properties than SSG.

3 Period of SSG-XOR

Let $\widetilde{a} = (a_i)_{i \geq 0}$ be an output sequence of a non-trivially initialized m-LFSR of length n. We can deduce the following proposition from Proposition 1 in [7].

Proposition 1. *Let $\widetilde{a}$ be an m-sequence generated by an LFSR of length n and let $\widetilde{s}$ be the output sequence of SSG-XOR obtained from $\widetilde{a}$. Then $\widetilde{s}$ is a balanced sequence and the period of $\widetilde{s}$ divides 2^n.*

Proof. This argument can be straightforwardly obtained from the proof of Proposition 1 in [7]. □

In order to obtain a lower bound on period of SSG-XOR, we need some properties of the trace map. For $\alpha \in GF(2^m)$, the trace of α over $GF(2)$ is defined by $Tr(\alpha) = Tr_{GF(2^m)/GF(2)}(\alpha) = \sum_{i=0}^{m-1} \alpha^{2^i}$. Note that the trace map is linear, that is, $Tr(\alpha + \beta) = Tr(\alpha) + Tr(\beta)$ for all $\alpha, \beta \in GF(2^m)$. The following theorem provides a convenient method representing the output sequence of an LFSR.

Proposition 2. ([6], Theorem 6. 24) *Let $\widetilde{a} = (a_i)_{i \geq 0}$ be an m-sequence generated by an LFSR of length n. Then there exists a non-zero element $c \in GF(2^n)$ and a primitive element $\beta \in GF(2^n)$ such that $a_i = Tr(c\beta^i)$ for all non-negative integer i.*

Let $\widetilde{a}$ and $\widetilde{b}$ be the sequences generated by the same LFSR of length n. If the LFSR is non-singular, $\widetilde{a}$ and $\widetilde{b}$ are called equivalent. If $\widetilde{a}$ and $\widetilde{b}$ are equivalent, $\widetilde{b}$ is a t-th shift sequence of $\widetilde{a}$ for some t. Let a sequence $\widetilde{a}^{(2)}$ be defined by $\widetilde{a}^{(2)} = (a_{2i})_{i\geq 0}$. Then, we obtain the following lemma.

Lemma 1. *Let $\widetilde{a} = (a_i)_{i\geq 0}$ be an m-sequence generated by an LFSR of length n. Then $\widetilde{a}^{(2)}$ and $\widetilde{a}$ are equivalent.*

Proof. By Proposition 2, there exist $c \in GF(2^n)$ and a primitive element $\beta \in GF(2^n)$ such that $a_i = Tr(c\beta^i)$ for all i. Therefore, $\widetilde{a}^{(2)}$ is represented by $(Tr(c), Tr(c\beta^2), Tr(c\beta^4), \ldots) = (Tr(c), Tr(c(\beta^2)), Tr(c(\beta^2)^2), \ldots)$. Let $f(x)$ be the primitive polynomial of β of degree n. Since $f(\beta^2) = f(\beta)^2 = 0$, $f(x)$ is also the minimal polynomial of β^2. That is, $\widetilde{a}^{(2)}$ and $\widetilde{a}$ are sequences generated by the same LFSR. Thus $\widetilde{a}^{(2)}$ and $\widetilde{a}$ are equivalent. □

Let P_{SSG} and $P_{SSG-XOR}$ be the periods of SSG and SSG-XOR, respectively. Let $\widetilde{s} = (s_i)_{i\geq 0}$ be the output sequence of SSG-XOR obtained from $\widetilde{a}$. Consider a sequence $\widetilde{s}^{+}$ defined by $(s_0 \oplus s_1, s_2 \oplus s_3, \ldots)$.

Theorem 1. *Let $\widetilde{a} = (a_i)_{i\geq 0}$ be an m-sequence generated by an LFSR of length n. Then, the output sequence of SSG is equivalent to $\widetilde{s}^{+}$. Thus*

$$2 \cdot P_{SSG} \leq P_{SSG-XOR} \cdot$$

Proof. By Proposition 2, there exist $c \in GF(2^n)$ and a primitive element $\beta \in GF(2^n)$ such that $a_i = Tr(c\beta^i)$ for all i. Consider a sequence $\widetilde{a}^{+}$ defined by $(a_0 \oplus a_1, a_2 \oplus a_3, \ldots)$. Since β is a primitive element of $GF(2^n)$, for each i, we obtain $a_{2i} \oplus a_{2i+1} = Tr(c\beta^{2i}) + Tr(c\beta^{2i+1}) = Tr(c(\beta^{2i} + \beta^{2i+1})) = Tr(c\beta^k) = a_k$ for some k. Similarly, $a_{2i+2} \oplus a_{2i+3} = Tr(c\beta^{k+2}) = a_{k+2}$. That is, $\widetilde{a}^{+} = (a_d, a_{d+2}, a_{d+4}, \ldots)$ for some d. By Lemma 1, sequences $\widetilde{a}^{+}$ and $\widetilde{a}$ are equivalent.

Now, consider the output sequence $\widetilde{s}$ of SSG-XOR. $\widetilde{s}$ is generated from $\widetilde{a}$ by the following rule: If $a_{4i} \oplus a_{4i+1} = 1$, $(s_{2j}, s_{2j+1}) = (a_{4i+2}, a_{4i+3})$ for some j. Otherwise, a_{4i+2} and a_{4i+3} are discarded. On the other hand, $\widetilde{s}^{+}$ can be generated from $\widetilde{a}^{+}$ by the following rule: If $a_{4i} \oplus a_{4i+1} = 1$, output $a_{4i+2} \oplus a_{4i+3}$. Otherwise, $a_{4i+2} \oplus a_{4i+3}$ is discarded. Note that since sequences $\widetilde{a}^{+}$ and $\widetilde{a}$ are equivalent, $a_{4i} \oplus a_{4i+1} = a_e$ and $a_{4i+2} \oplus a_{4i+3} = a_{e+1}$ for some e. In other words, the generation rule of $\widetilde{s}^{+}$ is that if $a_e = 1$ then the generator outputs a_{e+1}. This means that the output sequence of SSG and $\widetilde{s}^{+}$ are equivalent. From this fact, we can easily see that $2 \cdot P_{SSG} \leq P_{SSG-XOR}$. □

By Proposition 1 and Theorem 1, we know that $2 \cdot P_{SSG} \leq P_{SSG-XOR}$ and $P_{SSG-XOR}$ divides 2^n. Meier and Staffelbach [7] investigated the period of self-shrunken m-sequences generated by all possible LFSRs of length $n < 20$. As a result, they showed that the output sequences of SSG for all m-LFSR of length $n < 20$ except only one case, have maximum period 2^{n-1}. The case that does not reach the maximum period is the m-LFSR of length $n = 3$ defined by the recursion $a_n = a_{n-2} \oplus a_{n-3}$ and its period is 2 instead of the maximum period $2^{3-1} = 4$. In this case, the output sequence of SSG-XOR has the period 4 instead of maximum period $2^3 = 8$. Theorem 1 implies that the output sequences of SSG-XOR for all remaining cases have maximum period 2^n.

4 Linear Complexity of SSG-XOR

The linear complexity of a purely periodic sequence is defined to be the degree of the minimal polynomial $m(x)$ of the sequence. In other words, the linear complexity $L_{\widetilde{a}}$ of $\widetilde{a}$ is the length of the shortest LFSR which generates that sequence. Let P be the period of the sequence $\widetilde{a}$. Then we know that $m(x)$ divides $x^P - 1$. Let L_{SSG} and $L_{SSG-XOR}$ be the linear complexities of SSG and SSG-XOR, respectively. Recall that P_{SSG} divides 2^{n-1} (Proposition 1 in [7]) and $P_{SSG-XOR}$ divides 2^n (Proposition 1). Let $m_1(x)$ and $m_2(x)$ be the minimal polynomials of output sequences of SSG and SSG-XOR, respectively. Since P_{SSG} divides 2^{n-1}, $m_1(x) \mid x^{P_{SSG}} - 1 = (x+1)^{P_{SSG}}$ over $GF(2)$. Thus, $m_1(x) = (x+1)^{L_{SSG}}$. Similarly, $m_2(x) = (x+1)^{L_{SSG-XOR}}$.

Lemma 2. *Let t be odd and $g(x) = (x+1)^t = \sum_{i=0}^{t} \binom{t}{i} x^i$ over $GF(2)$. Then, for $0 \le m \le (t-1)/2$, $\binom{t}{2m+1} = \binom{t}{2m}$ over $GF(2)$.*

Proof. Since $\binom{t}{2m+1} = \binom{t}{2m} \frac{t-2m}{2m+1}$ and $t - 2m$ is odd, we obtain that $\binom{t}{2m+1} \equiv \binom{t}{2m} \bmod 2$. □

Theorem 2. *The linear complexity $L_{SSG-XOR}$ of SSG-XOR satisfies*

$$2 \cdot L_{SSG} \le L_{SSG-XOR} \ ,$$

where L_{SSG} is the linear complexity of SSG.

Proof. Let $L_{SSG-XOR} = L$. Let $(s_0, s_1, s_2, \ldots, s_{L-1})$ be the initial state of the shortest LFSR that generates an output sequence of SSG-XOR and $m_2(x)$ be its minimal polynomial. We first consider the case that L is even. Let $L = 2l$. We have $m_2(x) = (x+1)^{2l} = \sum_{i=0}^{2l} t_i x^i$ over $GF(2)$, where $t_i = 0$ if i is odd. Then $m_2(x)$ can be rewritten as

$$m_2(x) = 1 + x^{2i_1} + x^{2i_2} + \cdots + x^{2i_k} + x^{2l} \ ,$$

where $0 < i_1 < i_2 < \cdots < i_k < l$. Now consider the sequence $(s_{2l}, s_{2l+1}, s_{2l+2}, s_{2l+3}, \ldots)$ generated by the initial state $(s_0, s_1, s_2, \ldots, s_{2l-1})$ and $m_2(x)$. We have

$$\begin{aligned} s_{2l} &= s_0 \oplus s_{2l-2i_k} \oplus s_{2l-2i_{k-1}} \oplus \cdots \oplus s_{2l-2i_1} \ , \\ s_{2l+1} &= s_1 \oplus s_{2l-2i_k+1} \oplus s_{2l-2i_{k-1}+1} \oplus \cdots \oplus s_{2l-2i_1+1} \ . \end{aligned}$$

Similarly,

$$\begin{aligned} s_{2l+2j} &= s_{2j} \oplus s_{2l-2i_k+2j} \oplus s_{2l-2i_{k-1}+2j} \oplus \cdots \oplus s_{2l-2i_1+2j} \ , \\ s_{2l+2j+1} &= s_{2j+1} \oplus s_{2l-2i_k+2j+1} \oplus s_{2l-2i_{k-1}+2j+1} \oplus \cdots \oplus s_{2l-2i_1+2j+1} \ . \end{aligned}$$

Then, we obtain

$$s_{2l} \oplus s_{2l+1} = (s_0 \oplus s_1) \oplus (s_{2l-2i_k} \oplus s_{2l-2i_k+1}) \oplus \cdots \oplus (s_{2l-2i_1} \oplus s_{2l-2i_1+1})$$

and

$$s_{2l+2j} \oplus s_{2l+2j+1} = (s_{2j} \oplus s_{2j+1}) \oplus (s_{2l-2i_k+2j} \oplus s_{2l-2i_k+2j+1}) \oplus \cdots \oplus (s_{2l-2i_1+2j} \oplus s_{2l-2i_1+2j+1}) \ . \quad (1)$$

Therefore, sequence $(s_{2l} \oplus s_{2l+1}, s_{2l+2} \oplus s_{2l+3}, \ldots)$ is generated by the initial state $(s_0 \oplus s_1, s_2 \oplus s_3, \ldots, s_{2l-2} \oplus s_{2l-1})$ and linear recurrence (1). By Theorem 1, we can see that $(s_{2l} \oplus s_{2l+1}, s_{2l+2} \oplus s_{2l+3}, \ldots)$ is equivalent to the output sequence of SSG. Thus, $L_{SSG} \leq l$. This means that $2 \cdot L_{SSG} \leq L_{SSG-XOR}$.

Next we consider the case that L is odd. Let $L = 2l+1$. By Lemma 2, we have $m_2(x) = (x+1)^{2l+1} = \sum_{i=0}^{2l+1} t_i x^i$ over $GF(2)$, where $t_{2i} = t_{2i+1}$ for $0 \leq i \leq l$ and $t_0 = t_{2l} = 1$. Then $m_2(x)$ can be rewritten as

$$m_2(x) = 1 + x + x^{2i_1} + x^{2i_1+1} + x^{2i_2} + x^{2i_2+1} + \cdots + x^{2i_k} + x^{2i_k+1} + x^{2l} + x^{2l+1} \ ,$$

where $0 < i_1 < i_2 \cdots < i_k < l$. Now we can compute the sequence $(s_{2l+1}, s_{2l+2}, s_{2l+3}, s_{2l+4}, \ldots)$ as follows:

$$\begin{aligned} s_{2l+1} &= s_0 \oplus s_1 \oplus s_{2l-2i_k} \oplus s_{2l-2i_k+1} \oplus s_{2l-2i_{k-1}} \oplus s_{2l-2i_{k-1}+1} \\ &\quad \oplus \cdots \oplus s_{2l-2i_1} \oplus s_{2l-2i_1+1} \oplus s_{2l} \ , \\ s_{2l+2} &= s_1 \oplus s_2 \oplus s_{2l-2i_k+1} \oplus s_{2l-2i_k+2} \oplus s_{2l-2i_{k-1}+1} \oplus s_{2l-2i_{k-1}+2} \\ &\quad \oplus \cdots \oplus s_{2l-2i_1+1} \oplus s_{2l-2i_1+2} \oplus s_{2l+1} \ . \end{aligned} \quad (2)$$

Moving s_{2l+1} from the right-hand side to the left-hand side in (2), we obtain

$$s_{2l+1} \oplus s_{2l+2} = (s_1 \oplus s_2) \oplus (s_{2l-2i_k+1} \oplus s_{2l-2i_k+2}) \oplus \cdots \oplus (s_{2l-2i_1+1} \oplus s_{2l-2i_1+2}) \ .$$

Similarly,

$$s_{2l+2j+1} \oplus s_{2l+2j+2} = (s_{2j+1} \oplus s_{2j+2}) \oplus (s_{2l-2i_k+2j+1} \oplus s_{2l-2i_k+2j+2}) \oplus \cdots \oplus (s_{2l-2i_1+2j+1} \oplus s_{2l-2i_1+2j+2}) \ . \quad (3)$$

Therefore, sequence $(s_{2l+1} \oplus s_{2l+2}, s_{2l+3} \oplus s_{2l+4}, \ldots)$ is generated by the initial state $(s_1 \oplus s_2, s_3 \oplus s_4, \ldots, s_{2l-1} \oplus s_{2l})$ and the linear recurrence (3). By a technique similar to the case $L = 2l$, we can obtain that $2 \cdot L_{SSG} + 1 \leq L_{SSG-XOR}$. □

Meier and Staffelbach [7] presented experimental results which show that the upper bound on the linear complexity of an output sequence of SSG is $2^{n-1} - (n-2)$. This was proved later by Blackburn [1]. Hence, we can see by Theorem 2 that the upper bound on $L_{SSG-XOR}$ is at least $2^n - 2(n-2)$.

Table 1 shows the measured values of the linear complexities for output sequences of SSG and SSG-XOR generated by all m-LFSR sequences of length $n \leq 15$. By this table, we can see that all the output sequences of SSG-XOR with $n \leq 15$ have linear complexity near to $2^n - 2(n-2)$.

Table 1. Ranges of linear complexities (LCs) for SSG and SSG-XOR with $n \leq 15$

n	# of m-LFSRs	$2^n - 2(n-2)$	LC of SSG	LC of SSG-XOR
2	1	4	2	4
3	2	6	2–3	4–6
4	2	12	5	10–11
5	6	26	10–13	25–26
6	6	56	25–28	53–56
7	18	118	54–59	115–118
8	16	244	118–122	237–244
9	48	498	243–249	491–498
10	60	1008	498–504	1002–1008
11	176	2030	1009–1015	2023–2030
12	144	4076	2031–2038	4069–4076
13	630	8170	4072–4085	8164–8170
14	756	16360	8170–8180	16349–16360
15	1800	32742	16362–16371	32729–32742

5 Cryptanalysis

In this section we examine some possible attacks on SSG-XOR. First, we try to use known cryptanalytic techniques against SSG to break SSG-XOR. This approach is natural since SSG-XOR and SSG have close relation. Then we consider another attack suitable for SSG-XOR.

It is easy to see that in general, an algorithm that attacks SSG using l consecutive output bits can be modified to attack SSG-XOR using $2l$ consecutive output bits, according to Theorem 1. From now, we will try to modify other well-known cryptanalytic techniques against SSG to ones against SSG-XOR in a non-straightforward manner. In these attacks, we will assume that the attacker knows the feedback polynomial of the underlying LFSR.

Exhaustive search and entropy attack. Meier and Staffelbach [7] proposed two general methods in order to reconstruct the initial state of the internal LFSR from a known output sequence of SSG. These attacks operate on short keystream sequences, requiring $O(2^{0.79n})$ and $O(2^{0.75n})$ computational steps, respectively.

Now we apply these methods to SSG-XOR. The first method is an exhaustive search. Assume that $(s_0, s_1, s_2, s_3, \ldots)$ is the known output sequence generated by SSG-XOR. Let (a_0, a_1, a_2, a_3) be the first four bits of LFSR. Without loss of generality, we can assume $(a_0, a_1, a_2, a_3) = (a_0, a_1, s_0, s_1)$. Then there are two possible cases, i.e., $(a_0, a_1) = (1, 0)$ or $(a_0, a_1) = (0, 1)$. For the next four bits (a_4, a_5, a_6, a_7), there exist ten possibilities, i.e., two cases $(a_4, a_5, a_6, a_7) = (1, 0, s_2, s_3)$, $(a_4, a_5, a_6, a_7) = (0, 1, s_2, s_3)$, and other eight cases for $(a_4, a_5) = (0, 0)$ or $(a_4, a_5) = (1, 1)$, where (a_6, a_7) is discarded. This manipulation can be repeated for all subsequent 4-bit blocks of $\widetilde{a}$. Since there are $n/4$ blocks in total, there exist

$$S = 2 \cdot 10^{(n-1)/4} \approx 10^{n/4} = 2^{((\log_2 10)/4)n} = 2^{0.8305n}$$

possible initial states of the LFSR producing the known short sequence $(s_0, s_1, s_2, s_3, \ldots)$.

The second reconstruction algorithm using a short keystream sequence is to use the total entropy about the blocks. However, we can see easily that the average complexity of the algorithm using the entropy is much greater than $O(2^{0.8305n})$. Thus, SSG-XOR is more secure than SSG from the viewpoint of cryptanalysis proposed by [7].

BDD-based attack. Note that there are a few powerful attacks such as the backtracking-based attack [10] and the BDD-based attack [5]. Now we examine the BDD-based attack introduced by Krause [5], which is the best previously known short keystream attack against SSG. It computes the secret initial state from the first $\lceil 2.41n \rceil$ output bits in time $n^{O(1)}2^{0.6563n}$. (Although there is another fast attack using a tradeoff between time and length of available keystream, it requires unrealistic amount of keystream bits to achieve the time complexity comparable to the BDD-based attack [8].) Now, we will apply the BDD-based attack to SSG-XOR.

First, we explain the BDD-based attack briefly. A BDD (Binary Decision Diagram) is an efficient data structure to decide if a set of Boolean variables satisfy a Boolean equation. In the BDD-based attack, an attacker constructs a BDD which decides if a candidate initial state produces the given output sequence with the given generator. The attacker can find all candidate initial states using another efficient algorithm *SAT* which enumerates all candidates for a BDD. Note that there is only one candidate if a sufficiently long output sequence is available. According to the analysis given in [5], the required length of consecutive output bits is $\lceil \gamma\alpha^{-1}n \rceil$ and the time complexity is $n^{O(1)}2^{\frac{1-\alpha}{1+\alpha}n}$, where α and γ are defined as follows:

- α is the information rate (per bit) which an output stream $\widetilde{s} = shrink(\widetilde{a})$ reveals about the internal LFSR bitstream $\widetilde{a}$.
- γ is the maximal ratio of the length of an output bitstream $\widetilde{s}$ to the length of corresponding internal LFSR bitstream $\widetilde{a}$.

For SSG, $\alpha \approx 0.2075$ and $\gamma = 0.5$.

We can apply the BDD-based attack to SSG-XOR too, since SSG-XOR satisfies the same conditions which were required for the BDD-based attack against SSG, i.e., the BDD assumption and the pseudorandomness assumption. Now, we use a similar manipulation to that of [5] to estimate the parameters α and γ for the attack.

As in [5], we assume that for a fixed m the probability that $shrink(z)$ is a prefix of $\widetilde{s}$ for a randomly chosen and uniformly distributed $z \in \{0,1\}^m$ is the same for any output stream $\widetilde{s}$. Let us denote this probability by $p(m)$. Then there are $p(m)2^m$ possible z's such that $shrink(z)$ is a prefix of $\widetilde{s}$, since there are exactly 2^m z's with length m. Note that $p(m)$ can be supposed to behave as $p(m) = 2^{-\alpha m}$ by the definition of information rate α given in [5]. Thus, we get $p(m)2^m = 2^{(1-\alpha)m}$.

On the other hand, we observe that for all m with $m \equiv 0 \bmod 4$ and all output streams $\widetilde{s}$, $shrink(z)$ is a prefix of $\widetilde{s}$ for exactly $10^{m/4}$ strings z of length m. Hence, we obtain an information rate $\alpha = 1 - (\log_2 10)/4 \approx 0.1695$ for SSG-XOR by evaluating the relation $2^{(1-\alpha)m} = 10^{m/4}$. Note that γ for SSG-XOR is the same as that of SSG. Thus, we set $\gamma = 0.5$ for SSG-XOR.

Using the estimated values for α and γ, we see that the BDD-based attack can reconstruct the secret initial state from the $\lceil 2.95n \rceil$ consecutive output bits in time $n^{O(1)}2^{0.7101n}$. Note that the required length of output bits and the time complexity are increased. Therefore we conclude that SSG-XOR is more secure than SSG against the BDD-based attack.

Long-sequence attacks for SSG-XOR. If an attacker has a sufficiently long output sequence of SSG-XOR, then he or she can reduce the time complexity of attack significantly. One possible attack works as follows.

Let $(a_i, a_{i+1}, a_{i+2}, a_{i+3})$ be four consecutive bits of the underlying LFSR for SSG-XOR. If $a_i \oplus a_{i+1} = 1$, then $a_{i+2} = s_j$ and $a_{i+3} = s_{j+1}$ for some j. This produces three linear equations if s_j and s_{j+1} is known. Applying this fact repeatedly, we easily see that if an i is found such that

$$a_i \oplus a_{i+1} = a_{i+4} \oplus a_{i+5} = \cdots = a_{i+4*(n/3-1)} \oplus a_{i+4*(n/3-1)+1} = 1, \qquad (4)$$

then n linear equations for the LFSR sequence are generated when the corresponding sequence bits $s_j, \ldots$ are known, This is sufficient to reconstruct the secret initial state. Since the probability for (4) is $1/2^{n/3}$, the attacker can expect one success out of $2^{n/3}$ trials on different segments of the output sequence of SSG-XOR.

The complexity of the above attack is $O(n^3 2^{n/3})$ if we use a simple algorithm to solve a system of n linear equations for every trial. On the other hand, it is easy to see that the complexity of this kind of attack against SSG will be $O(n^3 2^{n/2})$. Hence SSG-XOR seems weaker than SSG from the viewpoint of the above attack, and moreover, this attack is better than any other known attacks against SSG and SSG-XOR described in this section. However, the problem is that the attacker has to know approximately $2^{n/3}$ and $2^{n/2}$ different segments of the output sequence to attack SSG-XOR and SSG, respectively. Therefore, we conclude that this kind of attack is impractical.

6 Conclusion

In this paper we proposed a variant of SSG constructed by using an extended selection rule, and examined cryptographic properties of this generator. This new generator is called SSG-XOR since the selection rule is determined by the XORed value of a couple of bits. We have proved that the period and the linear complexity of an output sequence of SSG-XOR are twice longer than those of SSG. Also we have shown that SSG-XOR is more secure than SSG against known previous attacks. While there is a new kind of attack that performs much better for SSG-XOR than for SSG, it seems impractical because the required amount of

known output sequence is too much. Since SSG-XOR can be implemented with the efficiency similar to SSG, SSG-XOR can be seen as an improved version of SSG from the viewpoint of security.

References

1. S. R. Blackburn, "The linear complexity of the self-shrinking generator," *IEEE Trans. Inform. Theory*, vol. 45, no. 6, pp. 2073–2077, Sep. 1999.
2. D. Coppersmith, H. Krawczyk, and Y. Mansour, "The shrinking generator," *Advances in Cryptology - CRYPTO '93*, LNCS 773, pp. 22–39, 1993.
3. P. Ekdahl, W. Meier, and T. Johansson, "Predicting the shrinking generator with fixed connections," *Advances in Cryptology-EUROCRYPT 2003*, LNCS 2656, pp. 330–344, 2003.
4. J. D. Golić, "Correlation analysis of the shrinking generator," *Advances in Cryptology-CRYPTO 2001*, LNCS 2139, pp. 440–457, 2001.
5. M. Krause, "BDD-based cryptanalysis of keystream generators," *Advances in Cryptology-EUROCRYPT 2002*, LNCS 2332, pp. 222–237, 2002.
6. R. Lidl and H. Niederreiter, *Introduction to finite fields and their applications*, Cambridge Univ. Press, 1994.
7. W. Meier and O. Staffelbach, "The self-shrinking generator," *Advances in Cryptology-EUROCRYPT '94*, LNCS 950, pp. 205–214, 1994.
8. M. J. Mihaljević, "A faster cryptanalysis of the self-shrinking generator," *Information Security and Privacy-ACISP '96*, LNCS 1172, pp. 182–189, 1996.
9. L. Simpson, J. D. Golić, and E. Dawson, "A probablistic correlation attack on the shrinking generator," *Information Security and Privacy-ACISP '98*, LNCS 1438, pp. 147–158, 1998.
10. E. Zenner, M. Krause, and S. Lucks, "Improved cryptanalysis of the self-shrinking generator," *Information Security and Privacy-ACISP 2001*, LNCS 2119, pp. 21–35, 2001.

On Constructing of a 32×32 Binary Matrix as a Diffusion Layer for a 256-Bit Block Cipher*

Bon Wook Koo, Hwan Seok Jang, and Jung Hwan Song

CAMP Lab., Hanyang University
17 Haengdang-dong, Seongdong-gu, Seoul, 133-791, Korea
{kidkoo, jhs1003, camp123}@hanyang.ac.kr
http://math.hanyang.ac.kr/camp/

Abstract. In this paper, we describe how to construct a 32×32 binary matrix of branch number 10, and use some mathematical techniques to find a form in product of matrices for increasing efficiency in software implementations of the binary matrix. We estimate a security against cryptanlysis when the binary matrix is used as a diffusion layer of a 256-bit SPN block cipher with an 8-bit s-box as a substitution layer in a round function. Also we describe the cryptanalytic properties such as the resistances to differential, linear, impossible differential, and truncated differential cryptanalysis. The number of operations to be required for implementing the binary matrix as a diffusion layer of a 256-bit SPN block cipher are given in this paper. We have a result that the binary matrix A is more efficient than the diffusion layer used Rijndael-256 on low bit platforms, such as 8-bit processors.

Keywords: Block cipher, diffusion layer, binary matrix, SPN.

1 Introduction

Diffusion layer is one of core components for a block cipher with confusion layer and the choice of a diffusion layer is an important factor on the security and efficiency of the cipher.

Most of diffusion layers that have been introduced are linear transformations on the vector space $GF(2^m)^n$ for mn-bit, which imply that they have matrix representations over $GF(2^m)$. The following 16×16 linear transformation(Table 1) is a matrix representation of diffusion layer of Rijndael-128.

Another widely used diffusion layer is linear transformation over $GF(2)$, so-called binary matrix. It has some advantages of implementing in bitwise operations. It gives relatively high diffusion effects without using any multiplication and it has a simple representation in $0, 1$-matrix form which is a binary matrix.

Some binary matrices are used as diffusion layers of block ciphers, such as in E2, Camellia, and ARIA. Block cipher E2 is a 12-round Feistel networks with an SPN round function using an 8×8 binary matrix whose branch number is 5.

* This research is supported by National Security Research Institute(NSRI), Korea.

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 51–64, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Table 1. The matrix representation of the diffusion layer of Rijndael-128

$$\begin{pmatrix}
02\ 00\ 00\ 00\ 00\ 00\ 00\ 03\ 00\ 00\ 01\ 00\ 00\ 00\ 01\ 00 \\
00\ 02\ 00\ 00\ 03\ 00\ 00\ 00\ 00\ 00\ 00\ 01\ 00\ 00\ 00\ 01 \\
00\ 00\ 02\ 00\ 00\ 03\ 00\ 00\ 01\ 00\ 00\ 00\ 00\ 01\ 00\ 00 \\
00\ 00\ 00\ 02\ 00\ 00\ 03\ 00\ 00\ 01\ 00\ 00\ 01\ 00\ 00\ 00 \\
01\ 00\ 00\ 00\ 00\ 00\ 00\ 02\ 00\ 00\ 03\ 00\ 00\ 00\ 01\ 00 \\
00\ 01\ 00\ 00\ 02\ 00\ 00\ 00\ 00\ 00\ 00\ 03\ 00\ 00\ 00\ 01 \\
00\ 00\ 01\ 00\ 00\ 02\ 00\ 00\ 03\ 00\ 00\ 00\ 00\ 01\ 00\ 00 \\
00\ 00\ 00\ 01\ 00\ 00\ 02\ 00\ 00\ 03\ 00\ 00\ 01\ 00\ 00\ 00 \\
01\ 00\ 00\ 00\ 00\ 00\ 00\ 01\ 00\ 00\ 02\ 00\ 00\ 00\ 03\ 00 \\
00\ 01\ 00\ 00\ 01\ 00\ 00\ 00\ 00\ 00\ 00\ 02\ 00\ 00\ 00\ 03 \\
00\ 00\ 01\ 00\ 00\ 01\ 00\ 00\ 02\ 00\ 00\ 00\ 00\ 03\ 00\ 00 \\
00\ 00\ 00\ 01\ 00\ 00\ 01\ 00\ 00\ 02\ 00\ 00\ 03\ 00\ 00\ 00 \\
03\ 00\ 00\ 00\ 00\ 00\ 00\ 01\ 00\ 00\ 01\ 00\ 00\ 00\ 02\ 00 \\
00\ 03\ 00\ 00\ 01\ 00\ 00\ 00\ 00\ 00\ 00\ 01\ 00\ 00\ 00\ 02 \\
00\ 00\ 03\ 00\ 00\ 01\ 00\ 00\ 01\ 00\ 00\ 00\ 00\ 02\ 00\ 00 \\
00\ 00\ 00\ 03\ 00\ 00\ 01\ 00\ 00\ 01\ 00\ 00\ 02\ 00\ 00\ 00
\end{pmatrix}$$

Camellia which is a modification of E2 uses a binary matrix similar to the one of E2.

In WISA 2003, mathematical techniques for efficient implementation of a binary matrix in 32-bit software environments has been introduced [1]. In ICISC 2003, Daesung Kwon et al. proposed a 128-bit block cipher ARIA [2] which used a 16×16 binary matrix of the maximum branch number 8. ARIA has been published in 2003 and established by KS(Korean Industrial Standards) in 2004. ARIA is an involutional SPN structure with the following 16×16 binary matrix(Table 2). The binary matrix in [1] and [2] has advantages to be implemented in 8-bit environments [1, 2, 13].

Since encryption and decryption of big size of input/output blocks imply more computation and space complexities, attacking such a block cipher of big size of input/output is more difficult to succeed in practice. Generally, a block cipher of big size of input/output is resisting against time and memory trade-off attacks, but not always. Since computing power is growing up and storage cost is getting down, attacking a cipher is getting fast to succeed. Therefore, we discuss a cryptographic component which is a binary matrix as a diffusion layer for a 256-bit input/output block cipher in this paper.

The diffusion layer of Rijndael-256 can be represented by 32×32 linear transformation over $GF(2^8)$. We find a binary matrix that is more efficient than a linear transformation over $GF(2^8)$ for implementing into 8-bit processor environments. We introduce a 32×32 binary matrix and will compare with the diffusion layer of Rijndael-256.

Notice that 32×32 binary matrix which we find is of branch number 10 that is not the maximum. We have found many binary matrices of branch number 12 regarded as the maximum, but there are some disadvantages to implement those matrices compared with other block ciphers in number of operations per round. Most of cases, it seems that the bigger branch number of the diffusion layer indicates that more computations are required for implementing the diffusion layer. So, we focus on a binary matrix of branch number 10 instead of a binary matrix of branch number 12.

Table 2. The binary matrix used in the block cipher ARIA

$$\begin{pmatrix}
0&0&0&1&1&0&1&0&1&1&0&0&0&1&1&0\\
0&0&1&0&0&1&0&1&1&1&0&0&1&0&0&1\\
0&1&0&0&1&0&1&0&0&0&1&1&1&0&0&1\\
1&0&0&0&0&1&0&1&0&0&1&1&0&1&1&0\\
1&0&1&0&0&1&0&0&1&0&0&1&0&0&1&1\\
0&1&0&1&1&0&0&0&0&1&1&0&0&0&1&1\\
1&0&1&0&0&0&0&1&0&1&1&0&1&1&0&0\\
0&1&0&1&0&0&1&0&1&0&0&1&1&1&0&0\\
1&1&0&0&1&0&0&1&0&0&1&0&0&1&0&1\\
1&1&0&0&0&1&1&0&0&0&0&1&1&0&1&0\\
0&0&1&1&0&1&1&0&1&0&0&0&0&1&0&1\\
0&0&1&1&1&0&0&1&0&1&0&0&1&0&1&0\\
0&1&1&0&0&0&1&1&0&1&0&1&1&0&0&0\\
1&0&0&1&0&0&1&1&1&0&1&0&0&1&0&0\\
1&0&0&1&1&1&0&0&0&1&0&1&0&0&1&0\\
0&1&1&0&1&1&0&0&1&0&1&0&0&0&0&1
\end{pmatrix}$$

In this paper, we describe how to construct such a 32×32 binary matrix of branch number 10, and use some mathematical techniques to find a form in product of matrices for increasing efficiency in software implementations. We estimate a security against cryptanlysis when the binary matrix is used as a diffusion layer of a 256-bit SPN block cipher with an 8-bit s-box as a substitution layer in a round function. Also we describe the cryptanalytic properties such as the resistance against differential [4], linear [5], impossible differential [6], and truncated differential [7] cryptanalysis for a 256-bit SPN block cipher. By counting the number of operations to be required for implementing round functions, the 12-round 256-bit SPN using the binary matrix and Rijndael-256 have been considered. And we conclude that the 12-round 256-bit SPN is more efficient than Rijndael-256 on low bit platforms.

2 Preliminaries

Throughout this paper, we consider the SPN with the round function shown as in Fig. 1. And let s_i be $m \times m$ bijective s-boxes(Later, we let $m = 8$).

Round keys that xored with data bits are derived from a master key by a key schedule algorithm which is not considered in this paper. We assume the round keys are independent and uniformly random, key addition layer does not effect on the number of active s-box.

We use the following definitions in this paper [3, 8].

Definition 1. *For any given $\Delta x, \Delta y, \Gamma x, \Gamma y \in Z_2^m$, the differential and linear probabilities of each s-box s_i are defined as :*

$$DP^{s_i}(\Delta x \to \Delta y) = \frac{\#\{x \in Z_2^m | s_i(x) \oplus s_i(x \oplus \Delta x) = \Delta y\}}{2^m}$$

$$LP^{s_i}(\Gamma y \to \Gamma x) = \left(2 \times \frac{\#\{x \in Z_2^m | x \cdot \Gamma x = s_i(x) \cdot \Gamma y\}}{2^m} - 1\right)^2.$$

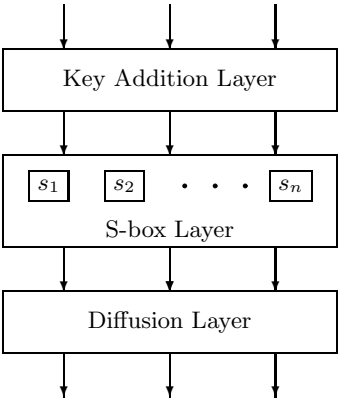


Fig. 1. A round function of an SPN structure

Definition 2. *The maximum differential and linear probabilities of s-boxes are defined as:*

$$p_s = \max_i \max_{\Delta x \neq 0, \Delta y} DP^{s_i}(\Delta x \rightarrow \Delta y)$$

$$q_s = \max_i \max_{\Gamma x, \Gamma y \neq 0} LP^{s_i}(\Gamma y \rightarrow \Gamma x)$$

This means that p_s, q_s are the upper bounds of the maximum differential and linear probabilities for all s-boxes.

Definition 3. *An s-box is a differential or linear active s-box if input difference or output mask value of the s-box is nonzero.*

Definition 4. *Let all s-boxes be bijective. The minimum numbers of differential s-boxes n_d and linear active s-boxes n_l in consecutive 2-round SPN for DC and LC are defined as*

$$n_d = \min_{\Delta x \neq 0} [w_H(\Delta x) + w_H(A(\Delta x))]$$

$$n_l = \min_{\Gamma A(x) \neq 0} [w_H(\Gamma x) + w_H(A(\Gamma x))],$$

where A is a diffusion layer and $w_H(x)$ is the hamming weight of n dimensional vector x which is the number of nonzero sub-blocks over $GF(2^m)$ of x; i.e. $w_H(x) = \#\{i | 0 \leq i < n, x_i \neq 0\}$.

Definition 5. *The branch number B_A of a diffusion layer A is defined by*

$$B_A = \min_{x \neq 0} [w_H(x) + w_H(A(x))].$$

Notice that the similarity of the definitions of the number of active s-boxes and branch number, the branch number of a diffusion layer gives the lower bound of number of active s-boxes.

s_1	s_5	s_9	s_{13}	s_{17}	s_{21}	s_{25}	s_{29}
s_2	s_6	s_{10}	s_{14}	s_{18}	s_{22}	s_{26}	s_{30}
s_3	s_7	s_{11}	s_{15}	s_{19}	s_{23}	s_{27}	s_{31}
s_4	s_8	s_{12}	s_{16}	s_{20}	s_{24}	s_{28}	s_{32}

Fig. 2. 4×8 array of s-boxes

3 Constructing 32×32 Binary Matrix

In this section, we introduce a method to construct 32×32 binary matrix which is used as a diffusion layer of 256-bit SPN block cipher. To construct a 'Good' binary matrix, we set some criteria as follows :

- The branch number is bigger than 10.
- Efficient implementation in 8-bit processor.
- Easy to implement in 32-bit processor.
- Secure against truncated and impossible differential attacks.

After generating huge number of matrices with the method in the following 3.1, matrices are filtered out by the criteria as described above.

3.1 Generation of Matrices

The basic concept for generating matrix is similar to 16×16 binary matrix introduced in [1]. For generating a 32×32 binary matrix, we rearrange 32 s-boxes to be a form of 4×8 s-box array shown in Fig. 2. From the formation of the 4×8 array, the following sequence of transformations are applied on array-by-array; the sequence is the "row-wise" transformation, "column-wise" transformation, and then "row-wise" transformation. A "row-wise" transformation is multiplying the same 8×8 binary matrix to each row of the s-box array. And a "column-wise" transformation is multiplying eight 4×4 binary matrices(allow to be same) to each column of the array. Each transformations are represented as 32×32 binary matrices.

Let R_0 be an 8×8 binary matrix for "row-wise" transformation;

$$R_0 = \begin{pmatrix} 1\,1\,1\,1\,1\,0\,0\,0 \\ 1\,0\,1\,1\,1\,1\,0\,0 \\ 1\,1\,1\,0\,0\,1\,1\,0 \\ 1\,1\,0\,1\,1\,0\,0\,1 \\ 1\,1\,0\,1\,0\,0\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1\,1 \\ 0\,0\,1\,0\,1\,1\,1\,1 \\ 0\,0\,0\,1\,1\,1\,1\,1 \end{pmatrix}.$$

The following 32×32 binary matrix R is representing a "row-wise" transformation.

$$R = \begin{pmatrix} I\ I\ I\ I\ I\ 0\ 0\ 0 \\ I\ 0\ I\ I\ I\ I\ 0\ 0 \\ I\ I\ I\ 0\ 0\ I\ I\ 0 \\ I\ I\ 0\ I\ I\ 0\ 0\ I \\ I\ I\ 0\ I\ 0\ 0\ I\ I \\ 0\ I\ I\ 0\ 0\ I\ I\ I \\ 0\ 0\ I\ 0\ I\ I\ I\ I \\ 0\ 0\ 0\ I\ I\ I\ I\ I \end{pmatrix},$$

where I is 4×4 identity matrix, and 0 is 4×4 zero matrix.

Let $C_0, C_1, \cdots, C_7$ be 4×4 binary matrices for column-wise transformation. The following 32×32 binary matrix C is representing a "column-wise" transformation.

$$C = \begin{pmatrix} C_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & C_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & C_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & C_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & C_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & C_6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & C_7 \end{pmatrix}.$$

Combining the above two transformations, 32×32 binary matrix

$$A = R \cdot C \cdot R \tag{1}$$

is the matrix which we use as a diffusion layer.

There are many choices of R and C with respect to $R_0, C_i (i = 0, 1, \cdots, 7)$. And we find R_0, $C_i (i = 0, 1, \cdots, 7)$ as follows so that $A = R \cdot C \cdot R$ satisfies the criteria in the above section in 3.

$$R_0 = \begin{pmatrix} 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0 \\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 0 \\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0 \\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1 \\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1 \\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1 \\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 1 \\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1 \end{pmatrix},$$

$$C_0 = C_3 = \begin{pmatrix} 1\ 1\ 1\ 0 \\ 0\ 1\ 1\ 1 \\ 1\ 0\ 1\ 1 \\ 1\ 1\ 0\ 1 \end{pmatrix}, C_1 = C_2 = \begin{pmatrix} 0\ 1\ 1\ 1 \\ 1\ 0\ 1\ 1 \\ 1\ 1\ 0\ 1 \\ 1\ 1\ 1\ 0 \end{pmatrix}$$

$$C_4 = C_7 = \begin{pmatrix} 1\ 0\ 1\ 1 \\ 1\ 1\ 0\ 1 \\ 1\ 1\ 1\ 0 \\ 0\ 1\ 1\ 1 \end{pmatrix}, C_5 = C_6 = \begin{pmatrix} 1\ 1\ 0\ 1 \\ 1\ 1\ 1\ 0 \\ 0\ 1\ 1\ 1 \\ 1\ 0\ 1\ 1 \end{pmatrix}$$

The 32×32 binary matrix $A = R \cdot C \cdot R$ in Table 3 is the matrix we will use in this paper.

The branch number of the binary matrix we find is 10, that is not the maximum since we have found binary matrices of branch number 12 by using different ways which are not explained in this paper. The maximum branch number of

Table 3. 32×32 binary matrix satisfying the criteria

$$A = \begin{pmatrix}
1&0&1&1&1&1&0&0&1&1&1&0&1&1&0&0&0&1&1&1&0&0&0&0&1&1&0&0&0&1&0&1\\
1&1&0&1&0&1&1&0&0&1&1&1&0&1&1&0&1&0&1&1&0&0&0&0&0&1&1&0&1&0&1&0\\
1&1&1&0&0&0&1&1&1&0&1&1&0&0&1&1&1&1&0&1&0&0&0&0&0&0&1&1&0&1&0&1\\
0&1&1&1&1&0&0&1&1&1&0&1&1&0&0&1&1&1&1&0&0&0&0&0&1&0&0&1&1&0&1&0\\
1&1&0&0&0&0&0&1&0&1&0&0&1&0&1&1&0&0&0&0&1&0&1&0&0&0&0&1&1&0&0&0\\
0&1&1&0&1&0&0&0&0&0&1&0&1&1&0&1&0&0&0&0&0&0&1&0&1&1&0&0&0&0&1&0&0\\
0&0&1&1&0&1&0&0&0&0&0&1&1&1&1&0&0&0&0&0&1&0&1&0&0&1&0&0&0&0&1&0\\
1&0&0&1&0&0&1&0&1&0&0&0&0&1&1&1&0&0&0&0&0&1&0&1&0&0&1&0&0&0&0&1\\
1&1&1&0&0&1&0&0&1&1&1&0&1&0&0&1&0&1&0&0&0&0&0&0&0&1&1&1&0&0&0&0\\
0&1&1&1&0&0&1&0&0&1&1&1&1&1&0&0&0&0&1&0&0&0&0&0&1&0&1&1&0&0&0&0\\
1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&0&0&0&0&1&0&0&0&0&1&1&0&1&0&0&0&0\\
1&1&0&1&1&0&0&0&1&1&0&1&0&0&1&1&1&0&0&0&0&0&0&0&1&1&1&0&0&0&0&0\\
1&1&0&0&1&0&1&1&1&0&0&1&0&1&1&1&1&1&0&0&1&1&0&0&0&0&0&0&1&1&1&0\\
0&1&1&0&1&1&0&1&1&1&0&0&1&0&1&1&0&1&1&0&0&1&1&0&0&0&0&0&0&1&1&1\\
0&0&1&1&1&1&1&0&0&1&1&0&1&1&0&1&0&0&1&1&0&0&1&1&0&0&0&0&1&0&1&1\\
1&0&0&1&0&1&1&1&0&0&1&1&1&1&1&0&1&0&0&1&1&0&0&1&0&0&0&1&1&0&0&1\\
0&1&1&1&0&0&0&0&0&1&0&0&1&1&0&1&0&0&0&0&1&0&0&0&1&0&1&1&0&1&0&0\\
1&0&1&1&0&0&0&0&0&0&0&1&0&0&1&1&0&1&0&0&0&1&0&0&0&0&0&1&1&0&1&0\\
1&1&0&1&0&0&0&0&0&0&0&0&1&0&0&1&1&0&1&0&0&0&1&0&0&1&0&0&1&0&0&1&0\\
1&1&1&0&0&0&0&0&0&1&0&0&0&1&0&0&1&0&0&1&0&0&0&1&0&1&1&0&0&0&0&0&1\\
0&0&0&0&1&0&1&0&0&0&0&0&1&1&0&0&0&0&0&1&1&0&1&1&1&1&0&0&1&0&1&1\\
0&0&0&0&0&1&0&1&0&0&0&0&0&1&1&0&1&0&0&0&1&1&0&1&0&1&1&0&1&1&0&1\\
0&0&0&0&1&0&1&0&0&0&0&0&0&0&1&1&0&1&0&0&1&1&1&0&0&0&1&1&1&1&1&0\\
0&0&0&0&0&1&0&1&0&0&0&0&1&0&0&1&0&0&1&0&0&1&1&1&1&0&0&1&0&1&1&1\\
1&1&0&0&0&0&0&1&0&1&1&1&0&0&0&0&0&1&1&0&1&1&0&0&0&1&1&1&0&0&0&0\\
0&1&1&0&1&0&0&0&1&0&1&1&0&0&0&0&0&0&1&1&0&1&1&0&1&0&1&1&0&0&0&0\\
0&0&1&1&0&1&0&0&1&1&0&1&0&0&0&0&1&0&0&1&0&0&1&1&1&1&0&1&0&0&0&0\\
1&0&0&1&0&0&1&0&1&1&1&0&0&0&0&0&1&1&0&0&1&0&0&1&1&1&1&0&0&0&0&0\\
0&1&0&1&1&0&0&0&0&0&0&1&1&1&0&1&0&0&0&1&0&1&1&0&0&0&0&1&1&1&0\\
1&0&1&0&0&1&0&0&0&0&0&0&1&1&1&0&1&0&0&1&1&0&1&0&0&0&0&0&1&1&1\\
0&1&0&1&0&0&1&0&0&0&0&0&1&0&1&1&0&0&1&0&1&1&1&0&0&0&0&1&0&1&1\\
1&0&1&0&0&0&0&1&0&0&0&0&1&1&0&1&0&0&0&1&0&1&1&1&0&0&0&0&1&1&0&1
\end{pmatrix}$$

32×32 binary matrix is regarded to be 12, which has not been proven yet. Those matrices of branch number 12 do not meet our criteria as we set in section 3, such as the criterion of easy to implement in 32-bit processor. There are trade-off relation between the minimum number of rounds and the number of operations per round for satisfying a certain level of security, that is the reason why we select the binary matrix of branch number 10. Many coding theory researchers suggest bounds for the maximum distance of binary linear codes. The bound for (64, 32) code is from 12 to 16, but there are examples of (64, 32, 12) code only and it has been proved that there is no (64, 32, 17) code [9]. Although we have found many 32×32 binary matrix of branch number 12, we do not show them because the generation method is different and they are not suitable for 32-bit processors comparing to use matrices of branch number 10.

4 Implementations

The binary matrix we found is suitable for efficient implementations on various environments, especially on 8-bit processors. We present some techniques which optimize implementations on 8-bit processors and on 32-bit processors similar to the techniques used to implement the block cipher ARIA. We set the input/output size of s-box is 8 which is widely used.

4.1 8-Bit Processor

On an 8-bit processor, the equation $A \cdot x = y$ is representing byte-wise xor's where A is the 32×32 binary matrix, and x, y are 32-dimensional binary vectors. Since the Hamming wight of the matrix A is 432, there would be 400 xor's for implementing the equation by elementary calculations. The total number of byte xor's can be reduced to 286 by using the following two equations.

$$\begin{aligned} y_{16} &= x_1 \oplus x_4 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_{11} \oplus x_{12} \oplus x_{13} \oplus x_{14} \oplus x_{15} \oplus \\ &\quad x_{17} \oplus x_{20} \oplus x_{21} \oplus x_{24} \oplus x_{29} \oplus x_{30} \oplus x_{32} \\ y_{22} &= x_6 \oplus x_8 \oplus x_{14} \oplus x_{15} \oplus x_{17} \oplus x_{21} \oplus x_{22} \oplus x_{24} \oplus x_{26} \oplus x_{27} \oplus \\ &\quad x_{29} \oplus x_{30} \oplus x_{32} \end{aligned} \tag{2}$$

Note that there are 28 xor's to get y_{16} and y_{22}. Using an additional variable T_1 reduces the total number of xor's from 28 to 19.

$$\begin{aligned} T_1 &= x_6 \oplus x_8 \oplus x_{14} \oplus x_{15} \oplus x_{17} \oplus x_{21} \oplus x_{24} \oplus x_{29} \oplus x_{30} \oplus x_{32} \\ y_{16} &= T_1 \oplus x_1 \oplus x_4 \oplus x_7 \oplus x_{11} \oplus x_{13} \oplus x_{20} \\ y_{22} &= T_1 \oplus x_{22} \oplus x_{24} \oplus x_{26} \oplus x_{27} \end{aligned} \tag{3}$$

By applying similar arguments, we reduce the total number of xor to 286 for implementing $A \cdot x = y$.

4.2 32-Bit Processor

On a 32-bit processor, the substitution and diffusion layers of Rijndael is encoded by 16 table-lookups followed by 16 xor's which are combined s-box layer with the diffusion layer [10].

We use this method to implement $A \cdot S$ where $A = R \cdot C \cdot R$(Section 3. 1) and S is the s-box layer. A permutation matrix P is introduced for reducing the number of byte operations by matrix decomposition and combining with s-box layer as followings.

The s-box layer S is combined with the following pre-defined binary matrix M and implemented into 32 table-lookups.

Table 4. The binary matrix M

$$M = \begin{pmatrix} T & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & T & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & T & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & T & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & T & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & T & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & T & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & T \end{pmatrix}, \text{ where } T = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

There exist the binary matrix C^* such that

$$A \cdot S = R \cdot C \cdot R \cdot S = R \cdot C^* \cdot C \cdot S. \tag{4}$$

It is clear that $C^* = C \cdot R \cdot C^{-1}$. Notice that M is an involution, we can find a permutation matrix Q such that $C = Q \cdot M$, which is $Q = C \cdot M$. We have the following equation (5),

$$A \cdot S = R \cdot C^* \cdot Q \cdot M \cdot S. \tag{5}$$

Finding a matrix P such that

$$P \cdot R = C^* \cdot Q \tag{6}$$

is equivalent to have $P = C^* \cdot Q \cdot R^{-1}$. Now we have the following equation (7).

$$A \cdot S = R \cdot P \cdot R \cdot M \cdot S. \tag{7}$$

Note that the matrix P is derived from C and the number of xor's for implementing $M \cdot S$ is the same as implementing S. So the choice of C is important for the efficiency of implementation on 32-bit processors.

5 Security

5.1 Resistance Against DC/LC

We consider a $2r$-round SPN where a round is consisting of the s-box layer followed by the 32×32 binary matrix A of branch number $B_A = 10$ as a diffusion layer as in Fig. 1. The s-box layer is consisting of the same 32 8-bit s-box which represents an exponential function x^e followed by an affine transformation.

Letting the maximum probabilities of differential and linear characteristic be 2^{-6}, the maximum probabilities of differential(P_D) and linear characteristic(P_L) for $2r$-round SPN are the followings [8].

$$P_D^{2r} \leq (2^{-6})^{(r \times B_A)}, \qquad P_L^{2r} \leq (2^{-6})^{(r \times B_A)}.$$

Since the branch number of the binary matrix A is 10, the maximum probabilities of differential and linear of 2-round SPN is bounded by $(2^{-6})^{(1 \times 10)} = 2^{-60}$.

The lower bounds for the number of active s-boxes and the upper bounds for probabilities of differential and linear characteristic in each rounds are listed in the following Table 5. If 256-bit key is used, the maximum probabilities of differential and linear characteristic of 10-round SPN is bounded by $(2^{-6})^{(5 \times 10)} = 2^{-300}$. Therefore the minimum number of rounds to be secure against differential and linear cryptanalysis is 10.

5.2 Resistance Against TDC/IDC

The probability p_r of an iterative difference can be estimated for r-round by the following equation (8) [11].

$$\begin{aligned} p_r &\approx (2^{-8})^{w_H(\Delta\widetilde{x}^1)-1} \times \cdots \times (2^{-8})^{w_H(\Delta\widetilde{x}^r)-1} \\ &= (2^{-8})^{w_H(\Delta\widetilde{x}^1)+\cdots+w_H(\Delta\widetilde{x}^r)-r}, \end{aligned} \tag{8}$$

Table 5. A lower bounds for the number of active S-boxes, upper bound for the probabilities of differential and linear characteristic

Rounds	The lower bounds on the number of active s-boxes	Upper bound on differential and linear probabilities
2	10	2^{-60}
3	11	2^{-66}
4	20	2^{-120}
5	21	2^{-126}
6	30	2^{-180}
7	31	2^{-186}
8	40	2^{-240}
9	41	2^{-246}
10	50	2^{-300}
11	51	2^{-306}
12	60	2^{-366}

where $\Delta\widetilde{x}^1$ is the byte pattern of an input difference and $\Delta\widetilde{x}^{i+1} = A(\Delta\widetilde{x}^i)$ is the byte pattern of the output difference.

By our investigation on all byte patterns, there is no 2∼10-round iterative byte pattern of input difference whose hamming weight is less than 6. There exists a 2-round iterative byte pattern of input difference of hamming weight 6, so we have $p_r = (2^{-8})^{6+4-2} = 2^{-64}$ and $(2^{-64})^5 = 2^{-320} \leq 2^{-256}$, which implies that the minimum number of rounds to be secure against TDC for the SPN with the matrix A is 10.

We have checked that there is no byte whose difference is zero(or nonzero) after two rounds where the round function is shown in Fig. 1. Also, we have checked that there is no byte whose difference is always zero or nonzero after two rounds in reverse direction, where the reverse direction means that the inverse of the round function. Therefore, we see that there is no impossible differential in four or more rounds. The number in each byte position in Table 7 of appendix A is representing the number of previous nonzero byte position affecting the current byte position.

6 Efficiency Comparison

Comparing with Rijndael-256, we consider a round function which is consisting of s-box layer, diffusion layer, and key xor layer. We show that the minimum number of rounds to be secure against DC and LC is 10. We set the total number of rounds 12, which is the minimum number of rounds derived from security analysis in Section 5 and [12]. We give some theoretic numbers for measuring efficiency of two SPN structures. In the tables 6, the numbers of operations to be required for implementation on 8-bit and 32-bit processors are listed. Both structure have a round function with the same size and number of table look-ups for each implementation.

Table 6. Number of operations for implementation

Processor	Cipher name	Rounds	Number of xor per round	Number of shift operation	Total number of xor
8-bit	Rijndael-256	14	280	-	3920
	SPN using A	12	286	-	3432
32-bit	Rijndael-256	14	32	24	448
	SPN using A	12	54	36	648

The total number of operations required for implementing of the SPN using A on an 8-bit processor is less than the those of Rijndael-256 as described in the Table 6. However the total number of operations required for implementing of the SPN using A on an 32-bit processor is bigger than the those of Rijndael-256 as described in the Table 6.

7 Conclusion

In this paper, we describe how to generate a cryptographically good 32×32 binary matrix. We generate the binary matrix A of branch number 10 and that is secure against well-known attacks when it is applying in reasonable rounds and suitable for various platforms. We compare the SPN structure using the binary matrix A with Rijndael-256 in efficiency point of view.

As the comparison results, the binary matrix A is more efficient than the diffusion layer used Rijndael-256 in 8-bit environments. This indicates that the binary matrix has better performance on low bit implementations. In fact, some implementing results show that the hardware performance of block cipher with binary matrix(e.g. ARIA) is better than Rijndael-128 [13].

Since the binary matrix A we introduce can be regarded as an extended version of binary matrix in ARIA, 32×32 binary matrix can be used as a diffusion layer of a 256-bit input/output block cipher. As we have discussed in Section 1, if we need to design a 256-bit block cipher, then the SPN-structure with the binary matrix that we suggest in Section 3 is a candidate of extension of ARIA. So we are working on designing an SPN structure block cipher with using the 32×32 binary matrix A and hope to have a detailed results of a complete block cipher.

References

1. BonWook Koo, HwanSeok Jang, JungHwan Song, *'Constructing and Cryptanalysis of a* 16×16 *Binary Matrix as a Diffusion Layer'*, WISA2003, LNCS 2908 , pp. 489-503, Springer-Verlag, 2003.
2. D. Kwon, J. Kim, S. Park, S.H. Sung, Y. Sohn, J.H. Song, Y. Yeom, E-J. Yoon, S. Lee, J. Lee, S. Chee, D. Han, and J. Hong, *'New Block Cipher : ARIA'*, ICISC2003, LNCS 2971 , pp. 432-445, Springer-Verlag, 2003.

3. Kanda, M., Takashima, Y., Matsumoto, T., Aoki, K., and Ohta, K. *'A strategy for construction fast round functions with practical security against differential and linear cryptanalysis'*. Selected Areas in Cryptography-SAC'98, LNCS 1556, , pp.264-279, Springer-Verlag, 1999.
4. E. Biham, A. Shamir. *'Differential Cryptoanalysis of DES-like Cryptosystems,'* Journal of Cryptology, Vol. 4 No. 1, pp.3-72, 1991. (The extended abstract appeared at CRYPTO'90)
5. M. Matsui. *'Linear Cryptoanalysis Method for DES cipher,'* Advances in Cryptology - EUROCRYPT'93, LNCS 765, pp.386-397, Springer-Verlag, 1994.
6. Kazumaro Aoki, Masayuki Kanda. *'Search for impossible Differential of E2'*. 1999, available at http://csrc.nist.gov/CryptoToolkit/aes/round1/pubcmnts.htm
7. Lars.R.Knudsen, *'Truncated and Higher Order Differentials'*, Fast Software Encryption Second International Workshop, LNCS 1008, pp.196-211, Springer-Verlag, 1995.
8. Kanda, *'Practical Security Evaluation against Differential and Linear Cryptanalysis for Feistel Ciphers with SPN Round Function'*, Selected Areas in Cryptography 2000: 324-338
9. Simon Litsyn, E. M. Rains, N. J. A. Sloane. available at http://www.math.unl.edu/djaffe/codes/webcodes/codeform.html
10. J.Daemen, V.Rijmen. *'AES proposal:Rijndael(Version 2)'*. Available at NIST AES website http://csrc.nist.gov/encryption/aes, 1999.
11. NTT Laboratories. *'Security of E2 against Truncated Differential Cryptanalysis(in progress)'*. 1999, available at http://csrc.nist.gov/CryptoToolkit/aes/round1/comments/990415-smoriai.pdf
12. Lars.R.Knudsen, *'The Number of Rounds in Block Ciphers'*, NESSIE public reports, NES/DOC/UIB/WP3/003/a, Available at http://www.cosic.esat.kuleuven.ac.be/nessie/reports/, 2000.
13. Jinsub Park, Yeonsang Yun, Yongdae Kim, Younggap You, *'ARIA Processor Design for Video Encryption'*, Proceedings of WISC2004, pp.83-90, 2004.

Appendix A

Table 7. The byte pattern of 2-round for 32 input states of hamming weight 1

Input	(1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(1 1 1 0 1 0 0 1 1 0 1 1 1 0 0 1 0 1 1 1 0 0 0 0 1 0 0 1 0 1 0 1)
2-round result	(12 6 13 12 6 8 3 4 9 8 9 7 9 11 8 8 4 5 7 9 5 3 5 8 8 10 6 5 4 6 5 8)
Input	(0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(0 1 1 1 1 1 0 0 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 1 0 0 1 0 1 0)
2-round result	(12 12 6 13 4 6 8 3 7 9 8 9 8 9 11 8 9 4 5 7 8 5 3 5 5 8 10 6 8 4 6 5)
Input	(0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(1 0 1 1 0 1 1 0 1 1 1 0 0 1 1 0 1 1 0 1 0 0 0 0 0 1 1 0 0 1 0 1)
2-round result	(13 12 12 6 3 4 6 8 9 7 9 8 8 8 9 11 7 9 4 5 5 8 5 3 6 5 8 10 5 8 4 6)
Input	(0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(1 1 0 1 0 0 1 1 0 1 1 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 0 1 0)
2-round result	(6 13 12 12 8 3 4 6 8 9 7 9 11 8 8 9 5 7 9 4 3 5 8 5 10 6 5 8 6 5 8 4)
Input	(0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(1 0 0 1 0 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0)
2-round result	(6 8 3 4 6 2 9 4 4 4 6 5 6 4 6 9 5 5 4 4 6 6 4 3 4 2 5 4 7 5 6 5)
Input	(0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0)
2-round result	(4 6 8 3 4 6 2 9 5 4 4 6 9 6 4 6 4 5 5 4 3 6 6 4 4 4 2 5 5 7 5 6)
Input	(0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(0 1 1 0 0 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 1 0)
2-round result	(3 4 6 8 9 4 6 2 6 5 4 4 6 9 6 4 4 4 5 5 4 3 6 6 5 4 4 2 6 5 7 5)
Input	(0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(0 0 1 1 1 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1)
2-round result	(8 3 4 6 2 9 4 6 4 6 5 4 4 6 9 6 5 4 4 5 6 4 3 6 2 5 4 4 5 6 5 7)
Input	(0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(1 0 1 1 0 0 0 1 1 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0)
2-round result	(9 8 9 7 4 4 6 5 8 8 11 6 5 4 6 8 7 7 4 5 4 4 2 3 7 7 7 6 3 3 2 6)
Input	(0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0)
2-round result	(7 9 8 9 5 4 4 6 6 8 8 11 8 5 4 6 5 7 7 4 3 4 4 2 6 7 7 7 6 3 3 2)
Input	(0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(1 1 1 0 0 1 0 0 1 1 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0)
2-round result	(9 7 9 8 6 5 4 4 11 6 8 8 6 8 5 4 4 5 7 7 2 3 4 4 7 6 7 7 2 6 3 3)
Input	(0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(0 1 1 1 0 0 1 0 0 1 1 1 1 0 0 1 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0)
2-round result	(8 9 7 9 4 6 5 4 8 11 6 8 4 6 8 5 7 4 5 7 4 2 3 4 7 7 6 7 3 2 6 3)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(1 0 0 1 1 1 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 0 0 1 0 0 0 0 1 0 1 1)
2-round result	(9 11 8 8 6 4 6 9 5 4 6 8 11 8 12 12 6 6 5 4 9 8 7 5 3 3 7 8 9 9 8 7)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(1 1 0 0 0 0 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1 0 0 0 0 0 0 1 1 0 1)
2-round result	(8 9 11 8 9 6 4 6 8 5 4 6 12 11 8 12 4 6 6 5 5 9 8 7 8 3 3 7 7 9 9 8)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(0 1 1 0 1 0 1 1 0 0 1 1 1 1 0 1 0 1 1 0 0 1 1 0 0 0 0 0 1 1 0)
2-round result	(8 8 9 11 6 9 6 4 6 8 5 4 12 12 11 8 5 4 6 6 7 5 9 8 7 8 3 3 8 7 9 9)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(0 0 1 1 1 1 0 1 1 0 0 1 1 1 1 0 0 0 1 1 0 0 1 1 0 0 0 0 0 1 1 1)
2-round result	(11 8 8 9 4 6 9 6 4 6 8 5 8 12 12 11 6 5 4 6 8 7 5 9 3 7 8 3 9 8 7 9)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(0 1 1 1 0 0 0 0 0 0 0 1 1 0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 0)
2-round result	(4 5 7 9 5 5 4 4 7 7 4 5 6 6 5 4 6 4 7 4 2 3 6 4 6 6 4 3 4 4 6 5)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(1 0 1 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0)
2-round result	(9 4 5 7 4 5 5 4 5 7 7 4 4 6 6 5 4 6 4 7 4 2 3 6 3 6 6 4 5 4 4 6)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(1 1 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 1 0)
2-round result	(7 9 4 5 4 4 5 5 4 5 7 7 5 4 6 6 7 4 6 4 6 4 2 3 4 3 6 6 6 5 4 4)

Table 7. (*Continued*)

Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 0 1)
2-round result	(5 7 9 4 5 4 4 5 7 4 5 7 6 5 4 6 4 7 4 6 3 6 4 2 6 4 3 6 4 6 5 4)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0)
1-round result	(0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 1 1 0)
2-round result	(5 3 5 8 6 6 4 3 4 4 2 3 9 8 7 5 2 3 6 4 8 4 11 8 4 5 3 4 7 6 8 6)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0)
1-round result	(0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 1 1 1)
2-round result	(8 5 3 5 3 6 6 4 3 4 4 2 5 9 8 7 4 2 3 6 8 8 4 11 4 4 5 3 6 7 6 8)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0)
1-round result	(0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 1 1 0 1 1 0 1 0 1 1)
2-round result	(5 8 5 3 4 3 6 6 2 3 4 4 7 5 9 8 6 4 2 3 11 8 8 4 3 4 4 5 8 6 7 6)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0)
1-round result	(0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 0 1 0 0 1 1 1 1 0 1)
2-round result	(3 5 8 5 6 4 3 6 4 2 3 4 8 7 5 9 3 6 4 2 4 11 8 8 5 3 4 4 6 8 6 7)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0)
1-round result	(1 0 0 1 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 0 0)
2-round result	(8 10 6 5 4 2 5 4 7 7 7 6 3 3 7 8 6 6 4 3 4 5 3 4 9 6 8 8 3 4 3 3)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0)
1-round result	(1 1 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 1 0 0 1 1 1 0 0 1 0 1 1 0 0 0 0)
2-round result	(5 8 10 6 4 4 2 5 6 7 7 7 8 3 3 7 3 6 6 4 4 4 5 3 8 9 6 8 3 3 4 3)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0)
1-round result	(0 1 1 0 0 0 0 1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1 1 0 1 0 0 0 0)
2-round result	(6 5 8 10 5 4 4 2 7 6 7 7 7 8 3 3 4 3 6 6 3 4 4 5 8 8 9 6 3 3 3 4)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0)
1-round result	(0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 1 1 0 0 0 0 0)
2-round result	(10 6 5 8 2 5 4 4 7 7 6 7 3 7 8 3 6 4 3 6 5 3 4 4 6 8 8 9 4 3 3 3)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0)
1-round result	(0 1 0 1 1 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 0 1 1)
2-round result	(4 6 5 8 7 5 6 5 3 3 2 6 9 9 8 7 4 4 6 5 7 6 8 6 3 4 3 3 10 6 11 6)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0)
1-round result	(1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0 0 1 1 0 1)
2-round result	(8 4 6 5 5 7 5 6 6 3 3 2 7 9 9 8 5 4 4 6 6 7 6 8 3 3 4 3 6 10 6 11)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0)
1-round result	(0 1 0 1 0 0 1 0 0 0 0 0 1 1 1 0 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0)
2-round result	(5 8 4 6 6 5 7 5 2 6 3 3 8 7 9 9 6 5 4 4 8 6 7 6 3 3 3 4 11 6 10 6)
Input	(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1)
1-round result	(1 0 1 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 1 0 0 0 0 0 0 1 1 1)
2-round result	(6 5 8 4 5 6 5 7 3 2 6 3 9 8 7 9 4 6 5 4 6 8 6 7 4 3 3 3 6 11 6 10)

On Algebraic Immunity and Annihilators

Xian-Mo Zhang[1], Josef Pieprzyk[1], and Yuliang Zheng[2]

[1] Centre for Advanced Computing - Algorithms and Cryptography
Department of Computing, Macquarie University
Sydney, NSW 2109, Australia
{xianmo, josef}@ics.mq.edu.au
[2] Department of Software & Information Systems
The University of North Carolina at Charlotte
9201 University City Blvd, Charlotte
NC 28223-0001, USA
yzheng@uncc.edu

Abstract. Algebraic immunity $AI(f)$ defined for a boolean function f measures the resistance of the function against algebraic attacks. Currently known algorithms for computing the optimal annihilator of f and $AI(f)$ are inefficient. This work consists of two parts. In the first part, we extend the concept of algebraic immunity. In particular, we argue that a function f may be replaced by another boolean function f^c called the algebraic complement of f. This motivates us to examine $AI(f^c)$. We define the extended algebraic immunity of f as $AI^*(f) = \min\{AI(f), AI(f^c)\}$. We prove that $0 \leq AI(f) - AI^*(f) \leq 1$. Since $AI(f) - AI^*(f) = 1$ holds for a large number of cases, the difference between $AI(f)$ and $AI^*(f)$ cannot be ignored in algebraic attacks. In the second part, we link boolean functions to hypergraphs so that we can apply known results in hypergraph theory to boolean functions. This not only allows us to find annihilators in a fast and simple way but also provides a good estimation of the upper bound on $AI^*(f)$.

Keywords: Algebraic Attacks, Algebraic Immunity, Hypergraph Theory, Greedy Algorithm.

1 Introduction to Algebraic Immunity

Recent algebraic attacks [4,5,3,6,14,7,8,2,1,16,9] have become a powerful tool that can be used for almost all types of cryptographic systems. Normally an algebraic attack is run in two stages. In the first stage, attackers build algebraic equations that reflect the relations between inputs, outputs and a secret key. In the second stage, attackers solve the algebraic equations in order to discover the secret key or restrict the secret key to a small domain (then exhaustively search the small domain). Algebraic attacks will be more efficient if algebraic equations have low degrees because the number of monomials (terms) of low degree is relatively small. Using annihilators is one of techniques to enable us to produce algebraic equations of low degree. Algebraic attacks have been used

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 65–80, 2006.
© Springer-Verlag Berlin Heidelberg 2006

very successfully to analyse LFSR-based stream ciphers because all the algebraic equations preserve their algebraic degree. The concept of annihilators for algebraic attacks was introduced by Courtois and Meier in [5]. For a boolean function f with n-bit inputs, $AN(f)$ is a set of boolean functions, defined as $AN(f) = \{g : (GF(2))^n \rightarrow GF(2) | f(x)g(x) = 0, \text{ for all } x \in (GF(2))^n\}$. Each function $g \in AN(f)$ is called an *annihilator* of f. Courtois and Meier [5] proposed three different scenarios to reduce the degree of algebraic equations. They discussed the relation among the three scenarios S3a, S3b and S3c in [6]. Later Meier at el. [14] showed that the scenario S3c can be replaced by the scenario S3a. Dalai et al [8] demonstrated that all the scenarios are equivalent to finding the union of two related annihilators, namely, $AN(f)$ and $AN(1 \oplus f)$ and then they defined the *algebraic immunity* $AI(f)$ as the minimum degree of nonzero boolean functions in $AN(f) \cup AN(1 \oplus f)$.

We now explain an application of annihilators to algebraic attacks. We may consider two types of algebraic equations, namely $f(x) = 0$ or $f(x) = 1$. For an algebraic equation $f(x) = 0$, multiplying the equation by g_1, such that $g_1 f = h$ is of a lower degree than the degree of f. Consequently, the attackers obtain a lower degree equation $h(x) = 0$. For the algebraic equation $f(x) = 1$, multiplying the equation by g_2 of a low degree such that $g_2 f$ is identical to the constant zero. Then the attackers obtain a lower degree equation $g_2(x) = 0$.

Courtois and Meier [5,6] studied $AI(f)$ and proved that $AI(f) \leq \lceil n/2 \rceil$ where $\lceil c \rceil$ denotes the smallest integer that is equal to or bigger than c. The problem of finding function f, such that $AI(f) = \lceil n/2 \rceil$, was examined in [8,2]. It is easy to observe that $AI(f)$ is never higher than its degree, i.e. $AI(f) \leq deg(f)$. This fact is true because $(1 \oplus f)f = 0$. In general, for any boolean function f of n variables, we have $AI(f) \leq \min\{deg(f), \lceil n/2 \rceil\}$. Very recently, Armknecht et al [1] presented a method by which the algebraic immunity of a random boolean function with n variables and degree d can be computed in $\mathcal{O}(D^2)$ steps where $D = \sum_{i=0}^{d} \binom{n}{i}$. This is an improvement on the previous best result $\mathcal{O}(D^3)$. This method is efficient for many classes of boolean functions including boolean functions of low degree. However D^2 will be as large as $\mathcal{O}(2^n)$ for random functions when d is larger than or close to $\frac{1}{2}n$.

2 Introduction to This Work

This work is composed of two parts. In the first part, we review the current definition of algebraic immunity and extend the concept. For a boolean function f, we create its algebraic complement f^c and define *extended algebraic immunity* of the function f as $AI^*(f) = \min\{AI(f), AI(f^c)\}$. We next prove that $0 \leq AI(f) - AI^*(f) \leq 1$. Since $AI(f) - AI^*(f) = 1$ holds for a large number of cases, the difference between $AI(f)$ and $AI^*(f)$ cannot be ignored in algebraic attacks. $AI^*(f)$ is applicable not only to LFSR-based stream ciphers but also to other ciphers whenever attackers can replace the original function f by f^c. In the second part, we apply the hypergraph theory to study annihilators. This new approach enables us to examine the relation among boolean functions f,

$1 \oplus f$, f^c and $1 \oplus f^c$. The main tool we use here is the concept of transversals in the hypergraph theory. We can produce annihilators of a function f, and its related functions $1 \oplus f$, f^c and $1 \oplus f^c$ and obtain an upper bound on $AI^*(f)$ in a fast and straightforward way. We also prove that the functions obtained in our approach must be annihilators, although they may not be optimal. Further we argue that the transversal number can be smaller than both $deg(f)$ and $\lceil n/2 \rceil$. This means the transversal number gives a new upper bound on $AI^*(f)$.

The rest of the paper is organised as follows. We review the definition of algebraic immunity $AI(f)$ and present the extended algebraic immunity $AI^*(f)$ in Section 3. We briefly introduce the hypergraph theory in Section 4. We describe the connection between boolean functions and hypergraphs in Section 5. In Sections 6 we show how to convert the problem of finding annihilators into the related problem of finding transversals in a hypergraph. Then in Section 7 we derive an upper-bound on $AI^*(f)$. In Section 8 we study boolean functions and their transversal numbers. In Section 9 we apply the well-known greedy algorithm in order to find annihilators for boolean functions in an efficient and straightforward way. Section 10 concludes the work. In the Appendix we elaborate how to use the greedy algorithm to obtain better annihilators.

3 Extended Algebraic Immunity

In this section we present the concept of extended algebraic immunity. Throughout the paper we are going to use the following notations. The vector space of n-tuples of elements from $GF(2)$ is denoted by $(GF(2))^n$. 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 $GF(2)$ or simply, a function f on $(GF(2))^n$. The *Hamming weight* of f, denoted by $HW(f)$, is defined as $HW(f) = \#\{\alpha \in (GF(2))^n,\ f(\alpha) = 1\}$, where $\#$ denotes the cardinality of a set. We express f as $f(x) = f(x_1,\ldots,x_n)$ where $x = (x_1,\ldots,x_n) \in (GF(2))^n$. The function f can be uniquely represented by a polynomial $f(x_1,\ldots,x_n) = \bigoplus_{\alpha \in (GF(2))^n} g(a_1,\ldots,a_n) x_1^{a_1} \cdots x_n^{a_n}$ where $\alpha = (a_1,\ldots,a_n)$, and g is also a function on $(GF(2))^n$. The polynomial representation of f is called the *algebraic normal form* (ANF) of the function and each $x_1^{a_1} \cdots x_n^{a_n}$ is called a *monomial (term)* in ANF of f. The *algebraic degree*, or simply *degree*, of f, denoted by $deg(f)$, is defined as the number of variables in the longest monomial of f, i.e., $deg(f) = \max\{HW(\alpha) \mid g(\alpha) = 1,\ \alpha \in (GF(2))^n\}$.

As an example, we consider stream ciphers based on LFSRs (Linear Feedback Shift Registers [10]). A such stream cipher is composed of two parts: a single LFSR defined by a *connection function* L and a *nonlinear filter* (boolean function) f on $(GF(2))^n$, where both L and f are known. A secret vector state K is also called the *initial state*. The stream cipher generates a sequence of keystream bits b_i as follows:

$$b_i = f(L^i(K)),\ \ i = 0, 1, \ldots \tag{1}$$

In a typical attack, adversaries wish to find the initial state K knowing the structure of the cipher (i.e. functions L and f) and a sequence of keystream bits b_i for some (not necessarily consecutive) clocks i. Since L is a linear transformation, $L^i(0) = 0$, $i = 0, 1, \ldots$. Therefore K must NOT be the all-zero state.

Notation 1. *Set* $\Delta(x) = (1 \oplus x_1) \cdots (1 \oplus x_n)$ *where* $x = (x_1, \ldots, x_n) \in (GF(2))^n$.

It is easy to prove the following lemma.

Lemma 1. *The function* $\Delta(x)$ *has the following properties. (i)* $\Delta(\alpha) \neq 0$ *if and only if* $\alpha = 0$, *(ii)* $h(x)\Delta(x)$ *is identical with the constant zero for any boolean function* $g \in (GF(2))^n$ *with* $h(0) = 0$, *(iii)* $h(x)\Delta(x)$ *is identical with* $\Delta(x)$ *for any boolean function* $g \in (GF(2))^n$ *with* $h(0) = 1$.

Notation 2. *Given a function* f *on* $(GF(2))^n$. *We define an algebraic complement of* f, *denoted by* f^c, *as the function that contains all monomials* $x_1^{a_1} \cdots x_n^{a_n}$, *where each* $a_j \in \{0, 1\}$, *that are not in ANF of the function* f.

The following properties of the algebraic complement are obvious: (1) $(f^c)^c = f$ for any function f; (2) any pair of functions (f, f^c) does not have any monomials in common.

Lemma 2. *Let* f *be a function on* $(GF(2))^n$. *Then (i)* $f^c(x) = \Delta(x) \oplus f(x)$ *for all* $x \in (GF(2))^n$, *(ii)* $f^c(x) = f(x)$ *for all nonzero* $x \in (GF(2))^n$.

Proof. It is easy to verify that ANF of $\Delta(x)$ contains all $2^n - 1$ possible monomials $x_1^{a_1} \cdots x_n^{a_n}$. Thus the statement (i) is true. Using Lemma 1, we can say that the statement (ii) holds.

Due to (ii) of Lemma 2, f can be replaced by f^c. This leads us to the following theorem.

Theorem 1. *Let the connection function* L *be nonsingular (i.e.* $L(\alpha) \neq L(\alpha')$ *if* $\alpha \neq \alpha'$*). Then Equation (1) is true if and only if*

$$b_i = f^c(L^i(K)), \quad i = 0, 1, \ldots \tag{2}$$

holds.

Proof. It is noted that the secret K must be nonzero. Since L is linear and nonsingular, $L^i(K) \neq 0$, $i = 0, 1, \ldots$. According to Lemma 2, we have proved the theorem. □

Note that there exists no guarantee that $AI(f)$ and $AI(f^c)$ are equal. This can be seen from a large number of evidences, for instance

Example 1. Let $f(x_1, x_2, x_3) = x_2x_3 \oplus x_2 \oplus x_3 \oplus x_1 \oplus 1$. Then its algebraic complement is $f^c(x_1, x_2, x_3) = x_1x_2x_3 \oplus x_1x_2 \oplus x_1x_3$. It is easy to check that $AI(f) = 2$ but $AI(f^c) = 1$. □

Clearly, in an algebraic attack, adversaries are going to compute both $AI(f)$ and $AI(f^c)$ and find annihilators for both functions f and f^c. Obviously, they can apply the annihilator whose degree is lowest. This is the reason why we need to revise the concept of algebraic immunity.

Definition 1. *Given a function f on $(GF(2))^n$. The* extended algebraic immunity *of f, denoted by $AI^*(f)$, is the minimum degree of nonzero boolean functions in $AN(f) \cup AN(1 \oplus f) \cup AN(f^c) \cup AN(1 \oplus f^c)$, or in other words, $AI^*(f) = \min\{AI(f), AI(f^c)\}$.*

Example 2. In Example 1, the extended algebraic immunity $AI^*(f) = 1$ but the algebraic immunity $AI(f) = 2$. □

Theorem 2. *Let f a function on $(GF(2))^n$. Then*

(i) $|AI(f) - AI(f^c)| \leq 1$,
(ii) $0 \leq AI(f) - AI^*(f) \leq 1$ *and* $0 \leq AI(f^c) - AI^*(f) \leq 1$.

Proof. Let $g \in AN(f) \cup AN(1 \oplus f)$ such that $deg(g) = AI(f)$. It is easy to see that there exists some i_0 with $1 \leq i_0 \leq n$ such that $1 \oplus x_{i_0}$ is not a factor of g, or in other words, g cannot be expressed as $g(x) = (1 \oplus x_{i_0})g'(y)$ where g' is a boolean function on $(GF(2))^{n-1}$. Hence $x_{i_0}g(x)$ is a nonzero function. Due to Lemma 1, $x_{i_0}\Delta(x)$ is identical with the constant zero. There exist two cases to be considered: $g \in AN(f)$ and $g \in AN(1 \oplus f)$. Consider the first case: $g \in AN(f)$. The function gf is identical with the constant zero. Therefore $x_{i_0}g(x)f^c(x)$ or $x_{i_0}g(x)(f(x) \oplus \Delta(x))$ is identical with the constant zero. This implies that $x_{i_0}g(x) \in AN(f^c)$ and thus $AI(f^c) \leq 1 + AI(f)$. We next consider the second case: $g \in AN(1 \oplus f)$. The function $g(1 \oplus f)$ is identical with the constant zero. Therefore $x_{i_0}g(x)(1 \oplus f^c(x))$ or $x_{i_0}g(x)(1 \oplus f(x) \oplus \Delta(x))$ is identical with the constant zero. This implies that $x_{i_0}g(x) \in AN(1 \oplus f^c)$ and thus $AI(f^c) \leq 1 + AI(f)$. We then have proved that $AI(f^c) \leq 1 + AI(f)$ in both cases. Since $(f^c)^c = f$, we know that $AI(f) \leq 1 + AI(f^c)$. $AI(f^c) \leq 1 + AI(f)$ and $AI(f) \leq 1 + AI(f^c)$ together imply that $-1 + AI(f) \leq AI(f^c) \leq 1 + AI(f)$, i.e., $|AI(f) - AI(f^c)| \leq 1$. Thus we have proved the relation (i) of the theorem. The relation (ii) is true due to (i) and the definition of $AI^*(f)$. □

Theorem 3. *Let f be a function on $(GF(2))^n$. Then $AI^*(f) = AI(f)$ if there exists some h in $AN(f) \cup AN(1 \oplus f)$ with $deg(h) = AI(f)$ and $h(0) = 0$, and, there exists some g in $AN(f^c) \cup AN(1 \oplus f^c)$ with $deg(g) = AI(f^c)$ and $g(0) = 0$.*

Proof. Let $h \in AN(f) \cup AN(1 \oplus f)$ with $deg(h) = AI(f)$ and $h(0) = 0$. Due to Lemma 1, the function $h(x)\Delta(x)$ is identical with the constant zero. Thus $h(x)f^c(x) = h(x)(f(x) \oplus \Delta(x)) = h(x)f(x)$. Similarly, $h(x)(1 \oplus f^c(x)) = h(x)(1 \oplus f(x) \oplus \Delta(x)) = h(x)(1 \oplus f(x))$. Consequently, h is either an annihilator of f^c or an annihilator of $1 \oplus f^c$ and then $AI(f^c) \leq AI(f)$. Symmetrically, $AI(f) \leq AI(f^c)$. Thus $AI(f^c) = AI(f)$ and thus $AI(f^*) = AI(f)$. □

Due to Theorem 3, $AI(f) - AI^*(f) = 0$ may hold sometimes. However the next example indicates that $AI(f) - AI^*(f) = 1$ can also hold.

Example 3. Let $\Delta(y)$ be the function on $(GF(2))^p$ defined in Notation 1 and $\beta_j \in (GF(2))^p$ be the binary representation of positive integer j, $j = 1, \ldots, 2^p-1$. Let $q \geq 2^p - 1$ be another integer. Thus there exist $2^p - 1$ linearly independent linear functions $\psi_1, \ldots, \psi_{2^p-1}$ on $(GF(2))^q$. Define a function on $(GF(2))^{p+q}$: $f(x) = \bigoplus_{j=1}^{2^p-1} \Delta(y \oplus \beta_j)\psi_j(z) \oplus \Pi_{i=1}^{q+p}(1 \oplus x_i)$ where $y = (x_1, \ldots, x_p)$, $z = (x_{p+1}, \ldots, x_{p+q})$ and $x = (y, z)$. Then $f^c(x) = \bigoplus_{j=1}^{2^p-1} \Delta(y \oplus \beta_j)\psi_j(z)$. It is not hard to verify that $AI(f) \geq p + 1$ and $AI(f^c) \geq p$. Since $\Delta(y)\Delta(y \oplus \beta)$ is identical with the constant zero for any nonzero $\beta \in (GF(2))^p$, $\Delta(y)$ is an annihilator of f^c. Thus $AI(f^c) \leq p$. $AI(f^c) \geq p$ and $AI(f^c) \leq p$ together imply that $AI(f^c) = p$. Due to $AI(f^c) = p$, $AI(f) \geq p + 1$ and Theorem 2, we have $AI(f) = p + 1$. Hence we have proved that $AI(f) = p + 1$ but $AI^*(f) = p$. □

Due to Example 3, $AI(f) - AI^*(f) = 1$ holds for a large number of boolean functions. Therefore the difference between $AI(f)$ and $AI^*(f)$ cannot be ignored in algebraic attacks. Observe that the extended algebraic immunity $AI^*(f)$ is not only relevant to LFSR-based stream ciphers but in general, to any ciphers whose initial states do not contain the zero vector.

4 Brief Introduction to Hypergraph

Hypergraph theory is a part of combinatorics. The word "hypergraph" was introduced in 1966. Let $X = \{x_1, \ldots, x_n\}$ be a finite set. Set $E = \{e_1, \ldots, e_m\}$, where each e_j is a subset of X. A *hypergraph*, denoted by $\aleph$, is the pair $\aleph = (X, E)$. Each x_j is called a *vertex*, $j = 1, \ldots, n$ and each e_j is called an *edge*; $j = 1, \ldots, m$. It should be noted that repeated edges are permitted. An edge $e \in E$ is called a *loop* if $\#e = 1$. The *rank* of $\aleph$ is defined as $\max\{\#e | e \in E\}$. In particular, the hypergraph $\aleph$ is called a *graph* if the rank of $\aleph$ is less or equal to 2. Graph theory was formed much earlier than hypergraph theory. Let X' be a subset of X and E' be a subset of E. If there exists some $e_j \in E'$ such that $X' \cap e_j \neq \emptyset$, where $\emptyset$ denotes the empty set, we simply say that X' and E' are *associated*. A star centered at a vertex x_j is a family of edges of $\aleph$ associated with x_j. The *degree* of vertex x_j, denoted by $\Delta_\aleph(x_j)$, is the size of the star centered at x_j. The maximum value of degrees of vertices is denoted by $\Delta(\aleph)$. Let $X' \subseteq X$, define $\aleph - X'$ as a hypergraph whose vertex set is $X - X'$ and whose edge set consists of all edges in E with all vertices in $X - X'$. A sequence $x_1e_1x_2e_2 \cdots x_pe_px_{p+1}$ is called a *path* of length p joining x_1 to x_{p+1}, where $p > 1$, all the e_j are distinct, x_j with $1 \leq j \leq p$ are distinct, and $x_j, x_{j+1} \in e_j$, $j = 1, \ldots, p$. In particular, if $x_1 = x_p$ then the path is called a *cycle* of length p. A subset of X, say S, is a *stable set* of $\aleph$, if $e_j \not\subseteq S$ for each $j = 1, \ldots, m$. The maximum cardinality of a stable set is called the *stability number* of $\aleph$ and it is denoted by $\varsigma(\aleph)$. A subset of X, say T, is a *transversal* of $\aleph$, if $T \cap e_j \neq \emptyset$ for each $j = 1, \ldots, m$. The minimum cardinality of a transversal is called the *transversal number* of $\aleph$ and it is denoted by $\tau(\aleph)$. A subset of E, say $M = \{e_{j_1}, \ldots, e_{j_q}\}$, is a *matching* of $\aleph$, if $e_{j_u} \cap e_{j_v} = \emptyset$, for $u \neq v$. The maximum number of edges in a matching is called the *matching number* of $\aleph$, denoted by $\nu(\aleph)$.

5 Relating Hypergraphs to Boolean Functions

Definition 2. *Let f be a function on $(GF(2))^n$. If the constant monomial in the ANF of f is zero (one) we say f to be* 0-CM *(*1-CM*).*

Definition 3. *Let $f(x)$ or $f(x_1, \ldots, x_n)$ be a 0-CM boolean function on $(GF(2))^n$, where $x = (x_1, \ldots, x_n)$. We now define a hypergraph $\aleph(f)$ associated with the function f as follows. The vertex set $X(f)$ of $\aleph(f)$ consists of all variables of the function f, i.e. $X(f) = \{x_1, \ldots, x_n\}$. A subset $e = \{x_{j_1}, \ldots, x_{j_s}\}$ over $X(f)$ is an edge of $\aleph(f)$ if and only if $x_{j_1} \cdots x_{j_s}$ is a monomial in ANF of f. Denote the collection of edges of $\aleph(f)$ by $E(f)$. The hypergraph $\aleph(f) = (X(f), E(f))$ is called the* hypergraph of the 0-CM boolean function f. *We define the* hypergraph of the 1-CM boolean function f *as the hypergraph of $1 \oplus f$ and use the same notation $\aleph(f) = (X(f), E(f))$.*

According to Definition 3, for any boolean function f, there uniquely exists a hypergraph $\aleph$ such that $\aleph = \aleph(f)$, but, for any hypergraph $\aleph$, there are precisely two boolean functions f and $1 \oplus f$ whose hypergraphs are identical, i.e. $\aleph = \aleph(f) = \aleph(1 \oplus f)$. Denote the stability number, the transversal number and the matching number of $\aleph(f)$ simply by $\varsigma(f)$, $\tau(f)$ and $\nu(f)$ respectively. In this way we can apply the known results in the hypergraph theory in our study of annihilators. The relation between boolean functions and hypergraphs was first introduced by Zheng et al in [20]. Note, however, that the authors of [20] used hypergraphs to examine the nonlinearity of boolean functions while in this work we use hypergraphs to study annihilators and extended algebraic immunity. It should also be noted that the relation between boolean functions and hypergraphs established in [20] contains a minor inaccuracy because 1-CM boolean functions do not correspond to any hypergraph. Note also that 0-CM and 1-CM can be united by the definition of algebraic immunity based on the scenarios S3a and S3b: let h have a lower degree than f, then h is an annihilator of f if and only if $h(1 \oplus f) = h$, while, h is an annihilator of $1 \oplus f$ if and only if $hf = h$.

6 Annihilators Versus Transversals

In this section we relate transversals to annihilators.

Lemma 3. *For a given 0-CM function f on $(GF(2))^n$, let $T = \{x_{j_1}, \ldots, x_{j_t}\}$ be a subset of $X(f)$. Then the following equation holds*

$$(1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t}) \cdot f = (1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t}) \cdot f|_{x_{j_1}=0,\ldots,x_{j_t}=0}.$$

Proof. Note that $a(1 \oplus a) = 0$ holds for any $a \in GF(2)$. Let $x_{i_1} \cdots x_{i_v}$ be a monomial in ANF of f. If $\{x_{i_1}, \ldots, x_{i_v}\} \cap \{x_{j_1}, \ldots, x_{j_t}\} \neq \emptyset$ then $x_{i_1} \cdots x_{i_v} \cdot (1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t})$ turns out to be the zero boolean function. If $\{x_{i_1}, \ldots, x_{i_v}\} \cap \{x_{j_1}, \ldots, x_{j_t}\} = \emptyset$ then $x_{i_1} \cdots x_{i_v} \cdot (1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t})$ will be different from zero. Note that the monomials of f that have empty intersection with T are uniquely identified by $f|_{x_{j_1}=0,\ldots,x_{j_t}=0}$. So the result follows. □

Note that Lemma 3 is relevant to the proof of Proposition 1 of [8] or the proof of Proposition 2 of [2]. The authors of [2,8] indicated that the algebraic immunity of a boolean function will be low if it has a sub-function of low degree. Since the authors of [2,8] did not determine how many variables or which variables are involved in such a sub-function, their claims need more investigation.

Lemma 4. *Let f be a 0-CM function on $(GF(2))^n$. Let $T = \{x_{j_1}, \ldots, x_{j_t}\}$ be a subset of $X(f)$. Then the following statements are equivalent: (i) T is a transversal of $\aleph(f)$, (ii) $(1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t})$ is an annihilator of the function f, (iii) $f|_{x_{j_1}=0,\ldots,x_{j_t}=0}$ vanishes or is identical with the constant zero.*

This lemma establishes a relation between annihilators of f and transversals of $\aleph(f)$. Due to Lemma 4, we can introduce the following equivalence.

Definition 4. *If $T = \{x_{j_1}, \ldots, x_{j_t}\}$ is a transversal of a 0-CM boolean function f then $(1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t})$ is called the annihilator of the function f,* corresponding to *the transversal T.*

7 Upper-Bound on Extended Algebraic Immunity

Theorem 4. *For any boolean function f on $(GF(2))^n$, the extended algebraic immunity of f is upper-bounded by its transversal number, i.e.,*

$$AI^*(f) \leq \min\{\tau(f), \tau(f^c)\}.$$

Proof. According to Lemma 4, $AI(f) \leq \tau(f)$ and $AI(f^c) \leq \tau(f^c)$. Then $AI^*(f) \leq \min\{\tau(f), \tau(f^c)\}$. □

In the hypergraph theory (see Section 3 of [11]), $\tau(\aleph) + \varsigma(\aleph) = n$, where $\varsigma(\aleph)$ is the stability number of $\aleph$. This equality and Theorem 4 imply that the following statement is true.

Corollary 1. *For any boolean function f on $(GF(2))^n$, the following upper bound on extended algebraic immunity holds:*

$$AI^*(f) \leq \min\{\lceil n/2 \rceil, deg(f), deg(f^c), \tau(f) = n - \varsigma(f), \tau(f^c) = n - \varsigma(f^c)\}.$$

According to Corollary 1, a large transversal number $\min\{\tau(f), \tau(f^c)\}$ is necessary for resistance against algebraic attacks. In the next section, we show a large number of boolean functions whose transversal numbers are less than both $deg(f)$ and $\lceil n/2 \rceil$. Therefore the new bound in Theorem 4 or Corollary 1 is non-trivial.

8 Boolean Functions with Low Transversal Number

Throughout this section, we discuss only f however we can do the same for f^c and then study $AI^*(f)$. We indicate that there exist a large number of boolean functions with small transversal number. It is known that the inequality $\nu(\aleph) \leq \tau(\aleph)$ holds for every hypergraph [11] where $\nu(\aleph)$ is the matching number of $\aleph$. The hypergraph $\aleph$ is said to satisfy the *König property* if $\nu(\aleph) = \tau(\aleph)$. We say that a boolean function f satisfies the König property if its hypergraph does.

Theorem 5. *Let f be a 0-CM boolean function on $(GF(2))^n$ satisfying the König property. Let M be a matching of $\aleph(f)$ such that $\#M = \nu(f)$. Let us denote $\lambda_M = \frac{1}{\nu(f)} \sum_{e \in M} \#e$. Then $AI(f) \leq \lfloor n/\lambda_M \rfloor$, where $\lfloor c \rfloor$ denotes the maximum integer less than or equal to c.*

Proof. It is noted that any two distinct $e \in M$ and $e' \in M$ are disjoint because M is a matching of $\aleph(f)$. Thus $\lambda_M \nu(\aleph) = \sum_{e \in M} \#e \leq n$. It follows that $\nu(\aleph) \leq n/\lambda_M$. Due to the König Property $\tau(f) = \mu(f)$, we know that $\tau(f) \leq n/\lambda_M$. Since $\tau(f)$ is an integer, $\tau(f) \leq \lfloor n/\lambda_M \rfloor$. We have proved the theorem. □

The following is a consequence of Theorem 5.

Corollary 2. *Let f be a 0-CM boolean function on $(GF(2))^n$ satisfying the König property. Let M be a matching of $\aleph(f)$ such that $\#M = \nu(f)$. Let $m_0 = \min\{\#e \mid e \in M\}$. Then $AI(f) \leq \lfloor n/m_0 \rfloor$.*

In Corollary 2, if $m_0 > \min\{2, n/deg(f)\}$ then $AI(f) < \min\{n/2, deg(f)\}$. Therefore the König property of a function may result in a lower algebraic immunity.

Notation 3. *Let f be a 0-CM function on $(GF(2))^n$ Let $f^{[i]}$, where $i = 1, \ldots, n$, denote the 0-CM function composed of all terms of f with degree at least i and $f_{[i]}$ denote the 0-CM function on $(GF(2))^n$ composed of all terms of f with degree at most $i-1$. Clearly $f = f^{[i]} \oplus f_{[i]}$.*

Lemma 5. *Let f be a 0-CM boolean function on $(GF(2))^n$ and $\aleph(f^{[i_0]})$ satisfy the König property for an integer i_0 with $2 \leq i_0 \leq n-1$. Then there exists a transversal $T = \{x_{j_1}, \ldots, x_{j_t}\}$ of $\aleph(f^{[i_0]})$ such that $t \leq \lfloor n/i_0 \rfloor$ and*

$$(1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t}) \cdot f = (1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t}) \cdot f_{[i_0]}|_{x_{j_1}=0,\ldots,x_{j_t}=0} \qquad (3)$$

where the degree of $(1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t}) \cdot f_{[i_0]}|_{x_{j_1}=0,\ldots,x_{j_t}=0}$ is at most $\lfloor n/i_0 \rfloor + i_0 - 1$, or, $(1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t}) \cdot f_{[i_0]}|_{x_{j_1}=0,\ldots,x_{j_t}=0}$ is identical with the constant zero.

Proof. Applying the proof of Theorem 5 to $\aleph(f^{[i_0]})$, we know that $\tau(f^{[i_0]}) \leq \lfloor n/\lambda_M \rfloor$, where λ_M is defined for $f^{[i_0]}$. Since $i_0 \leq \lambda_M$, we know that $\tau(f^{[i_0]}) \leq \lfloor n/i_0 \rfloor$. Thus there exists a transversal $T = \{x_{j_1}, \ldots, x_{j_t}\}$ of $f^{[i_0]}$ such that $\#T = t = \tau(f^{[i_0]}) \leq \lfloor n/i_0 \rfloor$. Therefore, from $f = f^{[i_0]} \oplus f_{[i_0]}$, we know that the equality (3) holds. If T is also a transversal of $f_{[i_0]}|_{x_{j_1}=0,\ldots,x_{j_t}=0}$ then $(1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t}) \cdot f_{[i_0]}|_{x_{j_1}=0,\ldots,x_{j_t}=0}$ will be identical with the constant zero. If T is not a transversal of $f_{[i_0]}|_{x_{j_1}=0,\ldots,x_{j_t}=0}$ then the degree of $(1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t}) \cdot f_{[i_0]}|_{x_{j_1}=0,\ldots,x_{j_t}=0}$ is at most $t + i_0 - 1 = \tau(f^{[i_0]}) + i_0 - 1 \leq \lfloor n/i_0 \rfloor + i_0 - 1$. We have proved the lemma. □

Corollary 3. *Let f be a 0-CM boolean function on $(GF(2))^n$. Let there be a subset X' of $X = \{x_1, \ldots, x_n\}$ such that $\aleph(f) - X'$ satisfies the König property. Let M be a matching of $\aleph(f) - X'$ such that $\#M = \nu(\aleph(f) - X')$. Denote $\lambda_M = \frac{1}{\nu(f)} \sum_{e \in M} \#e$, then $AI(f) \leq \#X' + \lfloor (n - \#X')/\lambda_M \rfloor$.*

Proof. Applying Theorem 5 to the hypergraph $\aleph(f) - X'$, we know that there exists a transversal $T' = \{x_{j_1}, \ldots, x_{j_t}\}$ of $\aleph(f) - X'$ such that $\#T' \leq \lfloor (n - \#X')/\lambda_M \rfloor$. Denote $T = X' \cup T'$. Clearly T is a transversal $\aleph(f)$ and $\#T = \#X' + \lfloor (n - \#X')/\lambda_M \rfloor$. Then the corollary holds. □

Corollary 3 shows that if a hypergraph does not satisfy the König property but its a sub-hypergraph obtained by removing some vertices does, then the transversal number τ can be small.

Example 4. In Lemma 5, if $\#i_0 \approx \sqrt{n}$ then the degree of $(1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t}) \cdot f_{[i_0]}|_{x_{j_1}=0,\ldots,x_{j_t}=0}$ is approximately $2\sqrt{n} - 1$. This shows a possible lower algebraic immunity. In fact, it is easy to verify that $AI(f) < \min\{n/2, deg(f)\}$ when $n \geq 12$ and $2\sqrt{n} - 1 < deg(f)$. It is noted that the real-valued function $\varphi(t) = n/t + t - 1$ reaches its minimum value $\varphi(\sqrt{n}) = 2\sqrt{n} - 1$. □

There exist many sufficient conditions for the König property. For example, a hypergraph $\aleph$ will satisfy the König property if it does not have a cycle of odd length [11].

9 Annihilators by Greedy Algorithm

Throughout this section, we discuss only the original function f and symmetrically we can do the same for the algebraic complement f^c. The *greedy algorithm* [11] is widely used in combinatorial optimisation. It is based on the natural principle of building up a solution from best choices that are made locally.

9.1 Annihilators of 0-CM Boolean Functions by Greedy Algorithm

Let f be a 0-CM boolean function over the set $X = \{x_1, \ldots, x_n\}$ of variables and its hypergraph $\aleph(f)$. We would like to find the transversal T of $\aleph(f)$. Below we give the description of such algorithm.

greedy algorithm (finds a transversal T of $\aleph(f)$)

1. Set $T_0 = \emptyset$.
2. For $k = 1, 2, \cdots$ do {
   - choose a vertex $x_{j_k} \in \aleph(f) - T_{k-1}$ where

$$\Delta_{\aleph(f)-T_{k-1}}(x_{j_k}) = \Delta(\aleph(f) - T_{k-1}),$$

   - set $T_k = T_{k-1} \cup \{x_{j_k}\}$,
   - if $\aleph(f) - T_k$ is empty return the transversal T_k and exit. }

Let $T = \{x_{j_1}, \ldots, x_{j_t}\}$ be a transversal obtained from the greedy algorithm. According to Lemma 4, we know that $(1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t})$ is an annihilator of f, i.e., $f \cdot (1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t})$ is identical with the constant zero.

Note that the greedy algorithm does not guarantee that the resulting transversal T is optimal. An optimal transversal should satisfy $\#T = \tau(f)$. However, we

still use the greedy algorithm to obtain a "reasonable" solution, and the greedy algorithm is often used in practice. We also note that there may exist two or more resulting transversals of $\aleph(f)$ by the greedy algorithm because, for example, there may exist two or more monomials whose degrees are equal to the $deg(f)$. Using the results from [12,19,13], the following statement can be proved.

Theorem 6. *Let f be a 0-CM boolean function with n variables. Then for any transversal T of $\aleph(f)$ obtained by the greedy algorithm, there is an upper bound on the cardinality of T and*

$$\#T \leq \tau(f)(1 + 1/2 + \cdots + 1/deg(f)).$$

Our considerations are illustrated on a boolean function f that was used as the filter function in the LILI-128 stream cipher [18]. Although the function f in the next example was studied in [6], in this work we use it to illustrate the greedy algorithm.

Example 5. Let f be the out filter function of LILI-128 (called f_d in [18]) that is a balanced, highly nonlinear and 3rd correlation immune boolean function of degree 6 on $(GF(2))^{10}$ constructed using design criteria given in [17]. ANF of f is taken from [6]. We next list all the monomials of f. They are

$x_2, x_3, x_4, x_5, x_6x_7, x_1x_8, x_2x_8, x_1x_9, x_3x_9, x_4x_{10}, x_6x_{10}, x_3x_7x_9, x_4x_7x_9, x_6x_7x_9,$
$x_3x_8x_9, x_6x_8x_9, x_4x_7x_{10}, x_5x_7x_{10}, x_6x_7x_{10}, x_3x_8x_{10}, x_4x_8x_{10}, x_2x_9x_{10}, x_3x_9x_{10},$
$x_4x_9x_{10}, x_5x_9x_{10}, x_3x_7x_8x_{10}, x_5x_7x_8x_{10}, x_2x_7x_9x_{10}, x_4x_7x_9x_{10}, x_6x_7x_9x_{10},$
$x_1x_8x_9x_{10}, x_3x_8x_9x_{10}, x_4x_8x_9x_{10}, x_6x_8x_9x_{10}, x_4x_6x_7x_9, x_5x_6x_7x_9, x_2x_7x_8x_9,$
$x_4x_7x_8x_9, x_4x_6x_7x_9x_{10}, x_5x_6x_7x_9x_{10}, x_3x_7x_8x_9x_{10}, x_4x_7x_8x_9x_{10}, x_4x_6x_7x_8x_9,$
$x_5x_6x_7x_8x_9, x_4x_6x_7x_8x_9x_{10}, x_5x_6x_7x_8x_9x_{10}.$

We apply the greedy algorithm to $\aleph(f)$. Since $\Delta_{\aleph(f)}(x_9) = \Delta(\aleph(f)) = 30$, we set $T_1 = \{x_9\}$. We then have $\aleph(f) - T_1 = \{x_2,\ x_3,\ x_4,\ x_5,\ x_6x_7,\ x_1x_8,\ x_2x_8,\ x_4x_{10},$ $x_6x_{10},\ x_4x_7x_{10},\ x_5x_7x_{10},\ x_6x_7x_{10},\ x_3x_8x_{10},\ x_4x_8x_{10},\ x_3x_7x_8x_{10},\ x_5x_7x_8x_{10}\}$. As $\Delta_{\aleph(f)-T_1}(x_{10}) = \Delta_{\aleph(f)-T_1} = 9$, we set $T_2 = T_1 \cup \{x_{10}\}$. Observe that $\aleph(f) - T_2 = \{x_2,\ x_3,\ x_4,\ x_5,\ x_6x_7,\ x_1x_8,\ x_2x_8\}$. Although we can continue the greedy algorithm until we find a transversal of f, we now stop the algorithm and multiple f by $(1 \oplus x_9)(1 \oplus x_{10})$. According to Lemma 3, we get

$$\begin{aligned}&(1 \oplus x_9)(1 \oplus x_{10}) \cdot f\\ = &(1 \oplus x_9)(1 \oplus x_{10}) \cdot (x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6x_7 \oplus x_1x_8 \oplus x_2x_8)\end{aligned}$$

Thus multiplying the equation $f(x_1, \ldots, x_{10}) = 1$ by $(1 \oplus x_9)(1 \oplus x_{10})$, we have

$$(1 \oplus x_9)(1 \oplus x_{10}) \cdot (1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6x_7 \oplus x_1x_8 \oplus x_2x_8) = 0$$

Similarly, multiplying the equation $f(x_1, \ldots, x_{10}) = 0$ by $(1 \oplus x_9)(1 \oplus x_{10})$, we receive

$$(1 \oplus x_9)(1 \oplus x_{10}) \cdot (x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6x_7 \oplus x_1x_8 \oplus x_2x_8) = 0$$

Therefore we have reduced degree of the equations from 6 to 4. □

9.2 Greedy Algorithm on 1-CM Boolean Functions

Let f be a 1-CM function on $(GF(2))^n$. Then $1 \oplus f$ is 0-CM. We apply the greedy algorithm to $\aleph(1 \oplus f)$ and obtain a transversal $T = \{x_{j_1}, \ldots, x_{j_t}\}$. According to Lemma 4, we know that $(1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t})$ is an annihilator of $1 \oplus f$, i.e., $(1 \oplus f) \cdot (1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t})$ is identical with the constant zero, or in other words, $f \cdot (1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t}) = (1 \oplus x_{j_1}) \cdots (1 \oplus x_{j_t})$.

9.3 Complexity of the Greedy Algorithm for Annihilators

We next investigate the complexity of the greedy algorithm for an annihilator.

Theorem 7. *For any function f on $(GF(2))^n$, by using the greedy algorithm, we can obtain an annihilator of f or $1 \oplus f$ in $n(n+1)$ steps.*

Proof. For the case that f is 0-CM, we first compute $\Delta_{\aleph(f)}(x_j) = d_j$, $j = 1, \ldots, n$. Thus it takes n steps to obtain $d_1, \ldots, d_n$. Set $p_1 = d_1$. Assume we have had p_k. Set p_{k+1}=max$\{p_k, d_{k+1}\}$. We then get p_n. Clearly $p_n = \max\{d_1, \ldots, d_n\}$. Thus we only need n steps to find p_n or x_{j_0} such that $\Delta_{\aleph(f)}(x_{j_0}) = \Delta_{\aleph(f)}$. Concluding, the computation takes at most $2n$ steps on $\aleph(f)$ to find $d_1, \ldots, d_n$, and p_n. Similarly, we compute the degree of each vertex of $\Delta_{\aleph(f)-\{x_{j_0}\}}$, and then find x_{j_1} such that $\Delta_{\aleph(f)-\{x_{j_0}\}}(x_{j_1}) = \Delta_{\aleph(f)-\{x_{j_0}\}}$. The computation takes at most $2(n-1)$ steps on $\aleph(f) - \{x_{j_0}\}$. By using the greedy algorithm, we can find an annihilator of a 0-CM function f with n variables in at most $2n+2(n-1)+\cdots + \leq n(n+1)$ steps. Since we can apply the greedy algorithm to $1 \oplus f$ when f is 1-CM, we then have proved the theorem. □

According to Theorem 7, the greedy algorithm is always fast. The algorithm guarantees the resulting function must be an annihilator although it may not be best (with minimum degree). The greedy algorithm will be refined in the Appendix.

10 Conclusions

We have argued that in algebraic attacks, boolean functions f may be replaced by their algebraic complements f^c. We then have introduced the extended algebraic immunity $AI^*(f) = \min\{AI(f), AI(f^c)\}$. We prove that $0 \leq AI(f) - AI^*(f) \leq 1$. We have also indicated that $AI(f) - AI^*(f) = 1$ holds for a large number of boolean functions. Therefore the difference between $AI(f)$ and $AI^*(f)$ cannot be ignored in algebraic attacks. We have established a relation between annihilators of boolean functions and traversals of hypergraphs. The relation allows us to find annihilators in a fast and effective way provided ANF of the function is known. In addition, we establish a new upper-bound on $AI^*(f)$. The new upper-bound and the algorithms together show that the new approach is helpful in analysis of the extended algebraic immunity $AI^*(f)$ and in finding annihilators.

Acknowledgment

The first two authors were supported by Australian Research Council grants DP0345366, DP0451484 and DP0663452. We would like to thank the referees for helpful suggestions.

References

1. F. Armknecht, C. Carlet, P. Gaborit, S. Künzli, W. Meier, and O. Ruatta. Efficient computation of algebraic immunity for algebraic and fast algebraic attacks. In *Advances in Cryptology - Eurocrypt'06*, volume 4004 of *Lecture Notes in Computer Science*, pages 147–164. Springer-Verlag, Berlin, Heidelberg, New York, 2006.
2. C. Carlet, D. Dalai, K. Gupta, and S. Maitra. Algebraic immunity for cryptographically significant boolean functions: Analysis and construction. *IEEE Transactions on Information Theory*, IT-xx No. x:xxx–xxx, 2006.
3. 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.
4. N. Courtois. Higher order correlation attacks, XL algorithm and cryptanalysis of Toyocrypt. In *The 5th International Conference on Information Security and Cryptology (ICISC'02), Seoul, Korea*, volume 2587 of *Lecture Notes in Computer Science*, pages 182–199. Springer-Verlag, Berlin, Heidelberg, New York, 2003.
5. 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.
6. N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. (`http://www.nicolascourtois.net/toyolili.pdf`), 2003.
7. D. Dalai, K. Gupta, and S. Maitra. Results on algebraic immunity for cryptographically significant boolean functions. In *Proceedings of Indocrypt 2004*, volume 3348 of *Lecture Notes in Computer Science*, pages 92–106. Springer-Verlag, Berlin, Heidelberg, New York, 2004.
8. D. Dalai, K. Gupta, and S. Maitra. Cryptographically significant boolean functions: Construction and analysis in term of algebraic immunity. In *Proceedings of Fast Encryption 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 98–111. Springer-Verlag, Berlin, Heidelberg, New York, 2005.
9. F. Didier and J. Tillich. Computing the algebraic immunity efficiently. In *Proceedings of Fast Encryption 2006*, volume xxxx of *Lecture Notes in Computer Science*, pages xxx–xxx. Springer-Verlag, Berlin, Heidelberg, New York, 2006.
10. S. W. Golomb. *Shift Register Sequences*. Laguna Hills, CA: Aegean Park, 1982.
11. R. L. Graham, M. Grötschel, and L. Lovász. *Handbook of Combinatorics*, volume I. Elsevier Science B. V., 1995.
12. D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System. Sci.*, 9:256–298, 1974.
13. L. Lovász. On the ratio of optimal fractional and integral covers. *Discrete Math.*, 13:383–390, 1975.
14. 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.

15. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, London, New York, Washington, D.C., 1997.
16. Y. Nawaz, G. Gong, and K. Gupta. Upper bounds on algebraic immunity of power functions. In *Proceedings of Fast Encryption 2006*, volume xxxx of *Lecture Notes in Computer Science*, pages xxx–xxx. Springer-Verlag, Berlin, Heidelberg, New York, 2006.
17. P. Sarkar and S. Maitra. Nonlinearity bounds and constructions of resilient boolean functions. In *Advances in Cryptology - CRYPTO2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 515–532. Springer-Verlag, Berlin, Heidelberg, New York, 2000.
18. L. Simpson, E. Dawson, J. Golic, and W. Millan. LILI keystream generator. In *Selected Areas in Cryptography, 7th Annual International Workshop, SAC2000*, volume 2012 of *Lecture Notes in Computer Science*, pages 248–261. Springer-Verlag, Berlin, Heidelberg, New York, 2001.
19. S. K. Stein. Two combinatorial covering theorems. *Journal of Combinatorial Theory A*, 16:391–397, 1974.
20. Y. Zheng, X. M. Zhang, and Hideki Imai. Restrictions, terms and nonlinearity of boolean functions. *Theoretical Computer Science*, 226:207–223, 1999.

Appendix: Multiple Greedy Algorithms for Annihilators

Throughout this section, we discuss only f and symmetrically we can do the same for f^c. It is noted that $1 \oplus f$ is an annihilator of f where f is any boolean function. Thus, if $\tau(f) > deg(f)$ then the greedy algorithm will fail. For this reason, in this section we strengthen the greedy algorithm in order to obtain better annihilators. By the improved algorithm, we may obtain a better annihilator of f even $\tau(f) > deg(f), \lceil n/2 \rceil$.

Let f boolean function on $(GF(2))^n$ (0-CM or 1-CM), $X = \{x_1, \ldots, x_n\}$ be the set of variables. Applying greedy algorithm, described in Section 9 to f or $1 \oplus f$, according to f is 0-CM or 1-CM, we obtain a transversal $T = \{x_{j_1}, \ldots, x_{j_t}\}$ of $\aleph(f)$ or $\aleph(1 \oplus f)$, where x_{j_1} is produced earliest in the algorithm, x_{j_2} is produced second earliest, ..., x_{j_t} is produced last. Based on the transversal T, we next present the *Multiple greedy algorithm* in a series of notations.

Notation 4. *We define a function D_β on $(GF(2))^k$, where $1 \leq k \leq r = \min\{\frac{1}{4}n, t-2\}$, as follows: $D_\beta(y) = (1 \oplus b_1 \oplus x_{j_1}) \cdots (1 \oplus b_k \oplus x_{j_k})$ where $y = (x_{j_1}, \ldots, x_{j_k})$, $\beta = (b_1, \ldots, b_k)$, $\{x_{j_1}, \ldots, x_{j_k}\} \subseteq T = \{x_{j_1}, \ldots, x_{j_t}\}$. We define $f_\beta(z) = f(x)|_{x_{j_1}=b_1,\ldots,x_{j_k}=b_k}$ where $z = (x_{i_1}, \ldots, x_{i_{n-k}})$ satisfying $\{x_{j_1}, \ldots, x_{j_k}\} \cup \{x_{i_1}, \ldots, x_{i_{n-k}}\} = \{x_1, \ldots, x_n\}$ with $i_1 < \cdots < i_{n-k}$.*

It is easy to see that

$$f(x) = \bigoplus_{\beta \in (GF(2))^k} D_\beta(y) f_\beta(z) \tag{4}$$

Due to the greedy algorithm, it should be noted that $y = (x_{j_1}, \ldots, x_{j_k})$ does not necessarily imply $j_1 < \cdots < j_k$.

Definition 5. *(4) is called* the kth greedy decomposition *of f with respect the transversal $T = \{x_{j_1}, \ldots, x_{j_t}\}$ of $\aleph(f)$. Each $f_\beta(z)$ is called a* subfunction *of f in the greedy decomposition (4).*

Notation 5. *Let k be a fixed integer with $1 \le k \le r$, where $r = \min\{\frac{1}{4}n, t-2\}$ and $t = \#T$. We write the kth greedy decomposition of f in the form (4). If $f_\beta(z)$ is a non-constant function, we apply the greedy algorithm to $f_\beta(z)$ (when $f_\beta(z)$ is 0-CM) or $1 \oplus f_\beta(z)$ (when $f_\beta(z)$ is 1-CM), and then we obtain the transversal $T_{k,\beta}$ of $f_\beta(z)$ or $1 \oplus f_\beta(z)$. Clearly $T_{k,\beta}$ is a subset of $\{x_{i_1}, \ldots, x_{i_{n-k}}\}$. We define an integer $\rho_{k,\beta} = \begin{cases} 0 & \text{if } f_\beta(z) \text{ is the constant one or zero} \\ \min\{deg(f_\beta(z)), \#T_{k,\beta}\} & \text{otherwise} \end{cases}$. We also define an integer $\rho_k = \min\{\rho_{k,\beta} | \beta \in (GF(2))^k\}$.*

Notation 6. *If exists some k such that $\rho_k = 0$, we define $k^* = \min\{k | \rho_k = 0\}$. In this case, by definition, there exists some $\beta_{j^*} \in (GF(2))^{k^*}$ such that $f_{\beta_{j^*}}(z)$ is the constant zero or one. Otherwise, $\rho_k > 0$, $k = 1, \ldots, r$, we define $k^{**} + \rho_{k^{**}} = \min\{k + \rho_k | 1 \le k \le r\}$, where $r = \min\{\frac{1}{4}n, t-2\}$ and $t = \#T$. In this case, by definition, there exists some $\beta_{j^{**}} \in (GF(2))^{k^{**}}$ such that $\rho_{k^{**},\beta_{j^{**}}} = \rho_{k^{**}}$.*

Theorem 8. *Let f be a function on $(GF(2))^n$ (0-CM or 1-CM).*

(i) if the first case (in Notation 6) occurs then $D_{\beta_{j^*}}(y)$ is an annihilator of f or $1 \oplus f$, where $deg(D_{\beta_{j^*}}) = k^*$,
(ii) if the second case (in Notation 6) occurs then there exists a function g on $(GF(2))^{n-k^{**}}$ such that $D_{\beta_{j^{**}}}(y)g(z)$ is an annihilator of f or $1 \oplus f$, where $deg(g) = \min\{deg(f_{\beta_{j^{**}}}(z)), \#T_{k,\beta_{j^{**}}}\}$,
(iii) both annihilators in (i) and (ii) have a degree is less than or equal to $t = \#T$.

Proof. We first prove (i) of the theorem. It is noted that $D_{\beta'}(y) \cdot D_{\beta''}(y)$ is identical with the constant zero when $\beta' \neq \beta''$. Therefore, according to (4), we know that $D_{\beta_{j^*}}(y)f(x) = D_{\beta_{j^*}}(y)f_{\beta_{j^*}}(z)$. Thus, if $f_{\beta_{j^*}}(z)$ is the constant zero then $D_{\beta_{j^*}}(y)$ is an annihilator of f, and, if $f_{\beta_{j^*}}(z)$ is the constant one then $D_{\beta_{j^*}}(y)$ is an annihilator of $1 \oplus f$. We next prove (ii). Similarly to the proof of (i), we have $D_{\beta_{j^{**}}}(y)f(x) = D_{\beta_{j^{**}}}(y)f_{\beta_{j^{**}}}(z)$. When $\#T_{k^{**},\beta_{j^{**}}} < deg(f_{\beta_{j^{**}}}(z))$, there exists an annihilator g of $f_{\beta_{j^{**}}}(z)$ or $1 \oplus f_{\beta_{j^{**}}}(z)$, where the annihilator g is corresponding to (see Definition 4) the transversal $T_{k^{**},\beta_{j^{**}}}$. Therefore $D_{\beta_{j^{**}}}(y)g(z)f(x) = D_{\beta_{j^{**}}}(y)g(z)f_{\beta j^{**}}(z) = 0$ if $f_{\beta_{j^{**}}}(z)$ is 0-CM, or, $D_{\beta_{j^{**}}}(y)g(z)f(x) = D_{\beta_{j^{**}}}(y)g(z)f_{\beta j^{**}}(z) = D_{\beta_{j^{**}}}(y)g(z)$, i.e., $D_{\beta_{j^{**}}}(y)g(z)(1 \oplus f(x)) = 0$ if $f_{\beta_{j^{**}}}(z)$ is 1-CM. This proves that $D_{\beta_{j^{**}}}(y)g(z)$ is an annihilator of f or $1 \oplus f$. When $\#T_{k^{**},\beta_{j^{**}}} \ge deg(f_{\beta_{j^{**}}}(z))$, we set $g = 1 \oplus f_{\beta_{j^{**}}}(z)$. Therefore $D_{\beta_{j^{**}}}(y)g(z)f(x) = D_{\beta_{j^{**}}}(y)g(z)f_{\beta j^{**}}(z)$ that is identical with the constant zero. This proves that $D_{\beta_{j^{**}}}(y)g(z)$ is an annihilator of f. We have completed the proof of (ii). We finally prove (iii). The degree of annihilator in (i) is equal to k^*. According the the multiple greedy algorithm, $k \le r$ where $t = \#T$. The degree of annihilator in (ii) is equal to $k^{**} + \rho_{k^{**},\beta_{j^{**}}} \le k^{**} + \#T_{k^{**},\beta_{j^{**}}} \le k^{**} + \#T_{k^{**},0}$. Recall that T is the transversal of $\aleph(f)$. Therefore $k^{**} + \#T_{k^{**},0} = \#T$. We have proved (iii). □

Definition 6. *We call the algorithm in this section the* multiple greedy algorithm. *To avoid confusion, we call the algorithm in Section 9 the* single greedy algorithm.

Theorem 9. *Let f be a function on $(GF(2))^n$ (0-CM or 1-CM). Let T with $\#T = t$ be a transversal of f by the Greedy Algorithm. Then an annihilator of f or $1 \oplus f$ can be computed by using the multiple greedy algorithm in $2^{r+1} \cdot n^2$ computing operations, where $r = \min\{\frac{1}{4}n, t-2\}$ and $t = \#T$.*

Proof. Due to the multiple greedy algorithm, for each k with $1 \leq k \leq r$, we do single greedy algorithm for at most 2^k functions on $(GF(2))^{n-k}$. According to Theorem 7, the computing operations is at most $\sum_{k=1}^{r} 2^k \cdot (n-k)(n-k+1) \leq n^2 \sum_{k=1}^{r} 2^k \leq n^2 \cdot 2^{r+1}$. □

The following statement is obvious.

Corollary 4. *In the multiple greedy algorithm, for any k with $1 \leq k \leq r$, where $r = \min\{\frac{1}{4}n, t-2\}$ and $t = \#T$, and any $\beta \in (GF(2))^k$, we have $AI(f) \leq k + AI(f_\beta) \leq k + \tau(f_\beta)$.*

According to Corollary 4, any degenerate subfunction is not desirable.

The main difference between the multiple and single greedy algorithms is that the multiple greedy algorithm contains many single greedy algorithms. It is noted that many subfunctions f_β are involved in the algorithm. This is helpful for the algebraic attacks because the subfunctions have less variables than original function f and some subfunctions may have a low degree or a small transversal number or satisfy the König property. Of course, The multiple greedy algorithm needs more computing times than the single greedy algorithm, but it results in better annihilators.

Note that Proposition 1 of [8] or Proposition 2 of [2] previously indicated that the algebraic immunity of a boolean function will be low if it has a subfunction of low degree. The main difference between the multiple greedy algorithm and the previous result is that the formula (4) is based on a transversal T of f, produced by the single greedy algorithm. Also the single Greedy Algorithm is further applied to each subfunction f_β in (4).

By the same reasoning, we can apply the multiple Greedy Algorithm to f^c and obtain an annihilator of f^c. Comparing the degree of the annihilator of f^c by the multiple greedy algorithm, to the degree of the annihilator of f by the same algorithm, we choose one with smaller degree between the two annihilators as the final result.

High-Speed RSA Crypto-processor with Radix-4 Modular Multiplication and Chinese Remainder Theorem

Bonseok Koo[1], Dongwook Lee[1], Gwonho Ryu[1],
Taejoo Chang[1], and Sangjin Lee[2]

[1] National Security Research Institute
161, Gajeong-dong, Yuseong-gu, Daejeon 305-350, Korea
{bskoo, dwlee, jude, tchang}@etri.re.kr
http://www.nsri.re.kr
[2] Graduate School of Information Security, Korea University
1, 5-Ka, Anam-dong, Sungbuk-ku, Seoul 136-701 Korea
sangjin@korea.ac.kr

Abstract. Today, RSA is one of the most popular public-key cryptosystem in various applications. In this paper, we present a high-speed RSA crypto-processor with modified radix-4 Montgomery multiplication algorithm and Chinese Remainder Theorem (CRT). Our design takes 0.84M clock cycles for a 1024-bit modular exponentiation and 0.25M clock cycles for two 512-bit exponentiations. Using 0.18 um standard cell library, the processor achieves 365Kbps for a 1024-bit exponentiation and 1,233Kbps for two 512-bit exponentiations at a 300MHz clock rate. For the high performance RSA crypto-system, the processor can also execute modular reduction, which is essential for calculating the Montgomery mapping constant and the modularly reduced ciphertext in CRT technique.

Keywords: RSA, Montgomery multiplication, Booth's algorithm, Carry Save Adder, Chinese Remainder Theorem.

1 Introduction

RSA [1] was, and still is the most widely used public-key cryptosystem. In the RSA cryptosystem, both encryption and decryption use modular exponentiation, which can be done by a sequence of modular multiplications. Therefore, the modular multiplication is the major factor which determines the performance of the RSA crypto-system. Many modular multiplication algorithms have been proposed in the past [2,3,4]. The Montgomery multiplication algorithm [4,5] is one of the most efficient algorithms. Montgomery's algorithm replaces trial division by the modulus with a series of additions and divisions by a power of two, thus it is well suited for hardware implementation. However, the key size in a RSA crypto-system is generally 512∼1024 bits to provide adequate security,

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 81–93, 2006.
© Springer-Verlag Berlin Heidelberg 2006

which means a high throughput is difficult to achieve. To improve the performance of modular multiplication, numerous high-radix modular multiplication algorithms have been proposed [6,7,8,9]. In high-radix algorithms, multi-bits of multiplicand are processed at one clock cycle, and thus the number of iterations can be reduced. Another way to speed up the RSA crypto-system is to use the Chinese Remainder Theorem (CRT) technique for decryption [10,11,12]. CRT is a divide-and-conquer method, which makes the RSA decryption almost 4 times faster, if two half-size exponentiations are calculated in parallel. Usually there are two implementation styles for Montgomery multiplication algorithm. One is to use Carry Save Adder(CSA) to avoid carry propagation [13,14,15] and the other is using the systolic-array element which mainly consists of 1-bit full adder [16,17,18,19]. For radix-2 multiplication, both methods are adequate for high-speed clock implementation, because the first method does not suffer carry propagation and the second method uses the pipelining structure based on 1-bit full adders. However, for high-radix multiplication, in CSA style implementation, it is not possible to directly determine the least significant bits of the intermediate result in carry redundant form, and also in systolic-array implementation, it is difficult to control multi-bits in 1-bit based pipelining structure and avoid latency problem.

In this paper, we introduce an architecture of a high-speed RSA crypto-processor based on modified radix-4 Montgomery multiplication algorithm and CRT. Using CSA, we design a radix-4 modular multiplier which is also suitable for CRT, and a modular reducer which can be used to compute the Montgomery domain mapping constant and the modularly reduced ciphertext in CRT. We also present implementation results and performance analysis of our results.

The rest of this paper is organized as follows. In section 2, we describe some algorithms for our crypto-processor. Section 3 describes the hardware architectures of our RSA cryptosystem. In section 4, we provide the implementation results and some comparisons with previously reported cases. Section 5 concludes the paper with some final remarks.

2 Algorithms

2.1 Montgomery's Modular Multiplication

Clearly, modular exponentiation is the main operation of the RSA algorithm. For modular exponentiation, a sequence of modular multiplications can be used. If the n-bit exponent $e{=}\sum_{i=0}^{n-1} e_i 2^i$, $e_i \in 0, 1$, exponentiation M^e is then,

$$M^e = M^{\sum_{i=0}^{n-1} e_i 2^i} = M^{2^{n-1} e_{n-1}} \cdots M^{2e_1} M^{e_0}. \tag{1}$$

Therefore, if we use multiplication instead of squaring, this can be done by a sequence of multiplications. This is known as the binary exponentiation algorithm and there are two binary methods; Left-to-Right(LR) and Right-to-Left(RL)

method according to the scanning direction of the bits of e. Both of them require n iterations for an n-bit exponent and each iteration contains two modular multiplications.

For three n-bit inputs A, B and N, radix-2 Montgomery's modular multiplication [4,5] is shown as follows.

Algorithm 1. Radix-2 Montgomery multiplication

input: $A = \sum_{i=0}^{n-1} a_i 2^i$, $B = \sum_{i=0}^{n-1} b_i 2^i$ and $N = \sum_{i=0}^{n-1} n_i 2^i$
output: $ABR^{-1} \bmod N$, where $R = 2^n \bmod N$

1. $S = 0$
2. **for** $i = 0$ to $n - 1$ **do begin**
3. $\quad q_i = S + Ba_i \bmod 2$
4. $\quad S = (S + Ba_i + q_i N)/2$
5. **end**
6. **if** $(S \geq N)$ $S = S - N$
7. **return** S

In order for the algorithm 1 to operate correctly, the condition $gcd(N, R) = gcd(N, 2^n) = 1$ must be satisfied. As N is the product of two large prime numbers, N is an odd number and this condition is always true. The most time consuming part of the algorithm is the calculation of S given by the three input addition(step 4). It comes from the carry propagation of the very large operand additions and this can be avoided by using CSA for additions. The function of CSA is to add three n-bit inputs X, Y and Z to produce two outputs C and S as results such that $C + S = X + Y + Z$. The ith bit of sum s_i and the $i + 1$th bit of carry c_{i+1} is calculated by the following functions with ith bits of three inputs, x_i, y_i, z_i,

$$\begin{aligned} s_i &= x_i \oplus y_i \oplus z_i \\ c_{i+1} &= x_i y_i \vee y_i z_i \vee z_i x_i, c_0 = 0. \end{aligned} \tag{2}$$

To improve the performance of modular multiplication, numerous high-radix modular multiplication algorithms have been proposed [6,7,8,9]. Hong et al. proposed radix-4 Montgomery modular multiplication algorithm based on Booth's multiplier [9]. With Booth's Recoding scheme, their radix-4 algorithm uses the multiples of B and N, $\{\pm B, \pm 2B\}$ and $\{\pm N, 2N\}$, instead of $\{B, 2B, 3B\}$ and $\{N, 2N, 3N\}$ which need higher hardware cost. However, for the case of radix-4 multiplication, we can not do mod 4 or division by 4 before the actual addition of C and S. Therefore, using 2-bit additions, we rearranged Hong et al.'s algorithm for the CSA style implementation. Let $BR((a_{2i+1}, a_{2i}, a_{2i-1}), B)$ denote the radix-4 Booth's recoding with inputs $(a_{2i+1}, a_{2i}, a_{2i-1})$ and B, which output neg_B and B_i as table 1. The modified radix-4 Montgomery multiplication algorithm with CSA is shown in algorithm 2.

Table 1. Radix-4 Booth's recoding rules in algorithm 2

a_{i+1}	a_i	a_{i-1}	neg_B	B_i	a_{i+1}	a_i	a_{i-1}	neg_B	B_i
0	0	0	0	0	1	0	0	1	$\neg 2B$
0	0	1	0	B	1	0	1	1	$\neg B$
0	1	0	0	B	1	1	0	1	$\neg B$
0	1	1	0	$2B$	1	1	1	0	0

Algorithm 2. Modified Radix-4 Booth Montgomery Multiplication using CSA, $R4MM(A, B, N)$

input: $A = \sum_{i=0}^{n} a_i 2^i$, $B = \sum_{i=0}^{n} b_i 2^i$, $N = \sum_{i=0}^{n-1} n_i 2^i$, $a_i, b_i, n_i \in \{0, 1\}$
output: $ABR^{-1} \bmod N$, where $R = 2^{\lceil (n+3)/2 \rceil}$

1. $C = 0$, $S = 0$, $a_{-1} = 0$, $c_{in} = 0$
2. **for** $i = 0, 1, 2, 3, ..., \lceil (n+3)/2 \rceil - 1$ **do begin**
3. $(neg_B, B_i) = BR((a_{2i+1}, a_{2i}, a_{2i-1}), B)$
4. $(C, S) = CSA(S, C, B_i)$
5. $(t_{i1}, t_{i0}) = ((s_1, s_0) + (c_1, neg_B) + c_{in}) \bmod 4$
6. $neg_N = (t_{i1} = n_1)$
7. **if** $(t_{i0} = 0)$ **do begin**
8. **if** $(t_{i0} = 0)$ $N_i = 0$
9. **else** $N_i = 2N$
10. **end**
11. **else do begin**
12. **if** $(neg_N = 1)$ $N_i = \neg N$, $C = C + neg_N$
13. **else** $N_i = N$
14. **end**
15. $(C, S) = CSA(S, C, N_i)$
16. $c_{in} = ((s_1, s_0) + (c_1, neg_B) + c_{in})/4$
17. $(C, S) = (C/4, S/4)$
18. **end**
19. $S = C + S$
20. **return** S

In this algorithm, $(C, S) = CSA(X, Y, Z)$ represents a CSA addition of three inputs X, Y and Z, and the two outputs of the addition are saved in C and S respectively. Unlike Carry Propagation Adder (CPA), CSA does not have a separate carry-input, hence it is impossible to calculate negative multiples of 2's complement numbers, B and N with CSA directly. Therefore, as table 1, for positive Booth's recoding cases, neg_B is 0 and B_i is B or $2B$, while in negative cases, neg_B is 1 and B_i is $\neg 2B$ or $\neg B$, the bit-wise inverses of B and $2B$. In the same manner, N_i is also represented by neg_N and $\neg N$.

Note that a 2-bit addition of two 2-bit inputs and 1-bit carry input is performed in step 5, and operation of mod 4 is simply done by taking 2-bit sum output as a result for the value of (t_{i1}, t_{i0}) which is used for determining N_i. In

steps 6∼15, N_i is selected by the condition of the values, (t_{i1}, t_{i0}) , (n_1, n_0) and then added to the intermediate value, (C, S). The calculation of $C + neg_N$ in step 12, can be easily done by substituting the Least Significant Bit (LSB) of C, (which is always 0) by neg_N. Also, another 2-bit addition is used in step 16, and division by 4 is simply done by only taking the carry output result to update c_{in} for the next iteration. Now that the two least significant bits of the sum of the intermediate value, in step 17 are 0, division by 4 is just 2-bit right shift of C and S. As a result, using two 2-bit additions and carry update, we can easily implement radix-4 Booth Montgomery multiplier based on CSA.

2.2 Modular Exponentiation and Chinese Remainder Theorem

As mentioned before, modular exponentiation can be done by a sequence of modular multiplications. L-R modular exponentiation algorithm using Montgomery multiplication is shown in algorithm 4, where $R4MM()$ denotes the radix-4 Montgomery Multiplication(algorithm 2).

Algorithm 3. L-R Montgomery Exponentiation

input: $M = \sum_{i=0}^{n-1} m_i 2^i$, $N = \sum_{i=0}^{n-1} n_i 2^i$, $e = \sum_{i=0}^{n-1} e_i 2^i$, $m_i, e_i, n_i \in \{0, 1\}$,
and $K = R^2 \bmod N = 4^{\lceil (n+3)/2 \rceil} \bmod N$
output: $M^e \bmod N$

1. $M = R4MM(M, K, N)$, $P = R4MM(K, 1, N)$
2. **for** $i = n-1, n-2, ..., 0$ **do begin**
3. $\quad P = R4MM(P, P, N)$
4. $\quad$ **if** $(e_i{=}1)$ $P = R4MM(P, M, N)$
5. **end**
6. $P = R4MM(P, 1, N)$
7. **if** $(P < 0)$ $P = P + N$
8. **return** P

By letting $K = R^2 \bmod N = 4^{\lceil (n+3)/2 \rceil} \bmod N$, the first phase (step 1) calculates residues with respect to R of initial values M and 1. Once the value of key N is given, the value K remains unchanged. It is thus necessary to pre-compute K before the modular exponentiation. The main part of the computation is a loop in which modular squares and multiplications are performed(steps 2∼5). In step 6, the result of the loop is switched back from the residue domain to the normal representation of numbers. Note that, since we use algorithm 3 for $R4MM()$, the intermediate results of M and P are in the range $[-N, N)$ during the entire exponentiation process, thus we only convert the final result in step 7 to the range $[0, N)$ by adding N to P if $P < 0$.

The RSA decryption and signature operation can be speeded up by using the CRT [11,12]. Since the modulus N is equivalent to $P \times Q$, the computation of $C^d \bmod N$ can be partitioned into two smaller parts and computed by CRT as follows.

Algorithm 4. RSA Decryption with CRT

input: C, P, Q, Q^{-1}, $d_P = d \bmod (P\text{-}1)$, $d_Q = d \bmod (Q\text{-}1)$
output: $M = C^d \bmod N$

1. $C_P = C \bmod P$
2. $C_Q = C \bmod Q$
3. $M_P = C_P^{d_P} \bmod P$
4. $M_Q = C_Q^{d_Q} \bmod Q$
5. $h = Q^{-1}(M_P - M_Q) \bmod P$
6. $M = M_Q + hQ$.
7. return M

It is obvious to see that the initial reductions of the ciphertext C(steps 1∼2) and the Chinese recombination(steps 5∼6) do not cause significant computational cost compared to the modular exponentiations in steps 3∼4. The two $n/2$-bit exponentiations are independent from each other, thus if calculated in parallel, the calculation time is about 4 times faster than that of the n-bit modular exponentiation which is used in non-CRT based decryption. This speedup comes from the half length of both, the exponent and the modulus.

2.3 Modular Reduction

For RSA computation based on Montgomery's modular multiplication and CRT technique, modular reduction is requested for calculating $K = R^2 = 4^{\lceil (n+3)/2 \rceil}$ in algorithm 3, and $C_P = C \bmod P$ and $C_Q = C \bmod Q$ in algorithm 4. The mapping constant K can be pre-computed for every public key N, however this requires a lot of spaces of memory. Moreover the modularly reduced ciphertexts C_P and C_Q can not be determined and pre-computed until the ciphertext C is received. Thus it is desirable to implement a modular reduction hardware for the high speed RSA crypto-system.

In 1998, Koc et al. proposed a fast modular reduction algorithm which uses CSA and the sign estimation technique [21]. This algorithm is very efficient for high speed hardware implementation because it uses CSA to avoid carry propagation and also only a 4-bit addition is required to estimate the sign of the intermediate result. However, Koc et al. assumed that the modulus N requires exactly n bits to represent it, i.e. $2^{n-1} \leq N < 2^n$. Hence, for arbitrary n-bit N, it is required to move the sign estimation position depending on the MSB position of N. This could be very inefficient in hardware implementation because the modulus N is usually of hundreds or thousands in bits in a RSA crypto-system.

Let us assume $X = \alpha N + \beta$ ($\alpha \geq 0$, $0 \leq \beta < N$), then

$$X2^l \bmod N2^l = (\alpha N + \beta)2^l \bmod N2^l = \beta 2^l. \tag{3}$$

Now, the reduction result $X \bmod N(=\beta)$ can be just obtained by l-bit right shift of $\beta 2^l$. Therefore, adding shift and count operations to the original algorithm, we can simply make the algorithm to work for arbitrary n-bit N as follows.

Algorithm 5. Modular Reduction with Sign Estimation for arbitrary n-bit N

input: $X = \sum_{i=0}^{k+n-1} x_i 2^i$, $N = \sum_{i=0}^{n-1} n_i 2^i$
output: X mod N
1. $S = 2^{-k-1}X$, $C = 0$, $l = 0$
2. **while** $(N_{n-1} = 0)$ **do begin**
3. $N = 2N$, $l = l + 1$
4. **end**
5. **for** $i = 0$ to $k + l$ **do begin**
6. **if** $ES(C, S)$=(+) $(C, S) = CSA(2S, 2C, -N)$
7. **elseif** $ES(C, S)$=(-) $(C, S) = CSA(2S, 2C, N)$
8. **else** $(C, S) = CSA(2S, 2C, 0)$
9. **end**
10. $S = C + S$
11. **if** $(S < 0)$ $S = S + N$
12. **while** $(l \neq 0)$ **do begin**
13. $S = S/2$, $l = l - 1$
14. **end**
20. **return** S

In this algorithm, $ES(C, S)$ denotes Koc et al.'s main idea, the estimated sign function which uses only 4-bit addition to estimate the sign of the intermediate result. For a more detailed explanation, please refer to [21]. Steps 2∼4 (shift left of N and count up of l operations) and steps 12∼14 (shift right of S and count down of l operations) are addintionally added to the original algorithm, and also the number of iterations has been changed to $k + l + 1$ from $k + 1$. The reduction iteration number $k + l + 1$ comes from the fact that the input number X is $(n + k)$-bits and the MSB l-bits of the n-bit number N are all '0's.

3 Architecture

3.1 Modular Multiplier

A fully parallel architecture has been designed to implement Montgomery modular multiplier using CSA and Booth's algorithm. Fig. 1 illustrates the architecture of the radix-4 Montgomery modular multiplier based on algorithm 2. It mainly consists of an $(n + 2)$-bit CSA, an $(n + 3)$-bit CSA, two 2-bit adders, a BR(Booth Recoding) block, a RT(Reduction Table) block, and three registers (A_reg, C_reg, S_Reg). The input A is stored in the register A_reg at first and right shifted by 2-bits in each clock cycle during the multiplication. The BR block generates the values neg_B and B_i from the inputs $(a_{2i+1}, a_{2i}, a_{2i-1})$ and B, and also the RT block generates the values neg_N and N_i from (t_{i1}, t_{i0}) and N. The two inputs of the multiplier, A and B are $(n + 1)$-bit size each because these are 2's complement values, and another input N is n-bit positive integer.

The value B_i generated by the BR block is one of $\{-2B, -B, 0, B, 2B\}$, thus the upper CSA is $(n+2)$-bit size and the lower CSA is 1-bit sign extended size i.e., $(n+3)$-bit. The adder_1 calculates the 2-bit value (t_{i1}, t_{i0}) which is used as the input of the RT block and the adder_2 generates the value c_{in} for the next iteration.

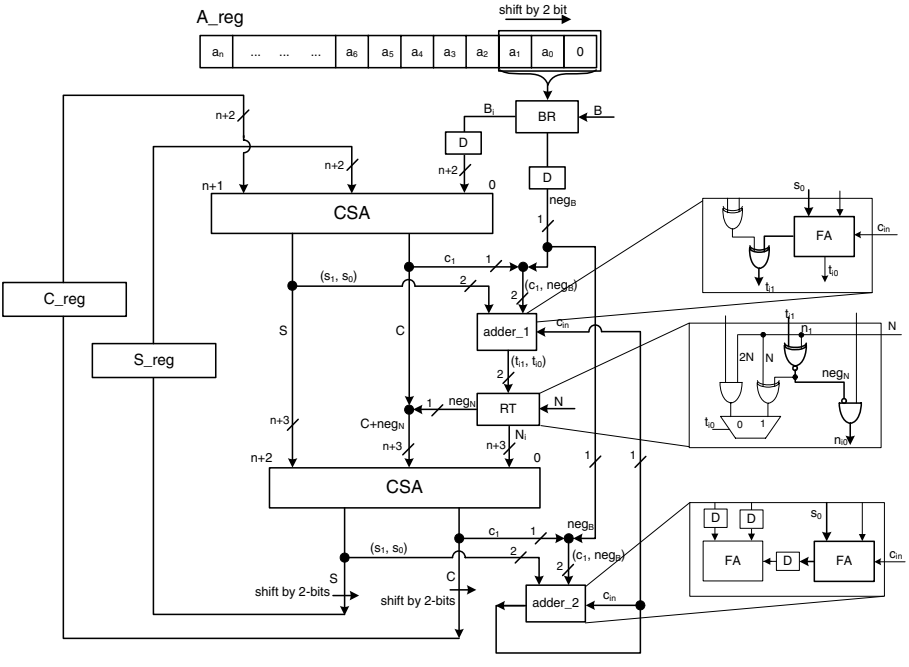


Fig. 1. The basic architecture of radix-4 modular multiplier

After $\lceil (n+3)/2 \rceil + 1$ clock cycles, the result of carry save representation, C and S are saved in C_reg and S_reg each. The w-bit adder can be used for summation of C and S in w-bits by w-bits to generate the final multiplication result. Therefore the radix-4 modular multiplier requires $\lceil (n+3)/2 \rceil + \lceil (n+2)/w \rceil + 1$ clock cycles for an n-bit multiplication.

As shown in Fig. 1, the critical path of the multiplier is "CSA → adder_1 → RT → CSA → adder_2". Because the adder_1 does not calculate carry output, the delay of that is only "Full Adder(FA)+XOR". Also, as N is an odd number, which means LSB is always 1, the LSB of N_i, n_{i0} can be directly determined by the values neg_N and t_{i0} i.e., $n_{i0} = \neg_N \wedge t_{i0}$, thus the delay of the RT block is "XOR+AND". Therefore the critical path delay of the proposed radix-4 multiplier is "4FAs+ 2XORs+ AND" with which we can expect tens or hundreds of MHz clock speed.

As explained above, two $n/2$-bit exponentiations are the heaviest computations in RSA decryption with CRT and if calculated in parallel, the decryption

time is about 4 times faster than that of the normal n-bit exponentiation. Because CSA has no carry propagation, it is easy to make a multiplier of our architecture to be able to calculate n-bit modular multiplication or two $n/2$-bit multiplications in parallel selectively. Therefore, we finally design a radix-4 multiplier which can support CRT technique. The designed multiplier treats the input $A(B)$ as an $(n+2)$-bit number or the concatenation of two independent $(n/2+1)$ numbers selectively. Also N can be treated as the concatenation of two $n/2$-bit numbers P and Q. As a result, our multiplier requires $\lceil (n+3)/2 \rceil + \lceil (n+2)/w \rceil + 1$ clock cycles for a n-bit multiplication and $\lceil (n/2+3)/2 \rceil + \lceil (n/2+2)/w \rceil + 1$ cycles for two $n/2$-bit multiplications.

3.2 Modular Reducer

We also present an architecture of a high speed modular reduction hardware based on algorithm 5. Fig. 2 illustrates the architecture of the modular reducer. For arbitrary $(n+k)$-bit and n-bit positive integers X and N, this hardware can calculate X mod N, and as depicted in Fig. 2, it consists of an $(n+2)$-bit CSA, a $\lceil \log_2 n \rceil$-bit up/down counter, a 4-bit adder, four registers(N_reg, C_reg, S_reg, L_reg) and some control logics.

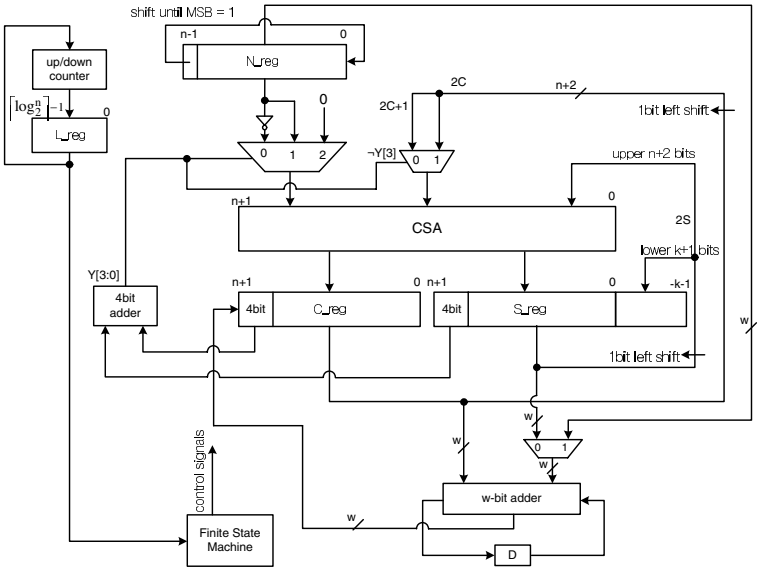


Fig. 2. The architecture of modular reducer

Once the operation starts, the N_reg register which stores the input N is left circular shifted until the MSB of the register is 1 and the $\lceil \log_2 n \rceil$-bit value which is stored in the L_reg register is up-counted from 0, simultaneously. The two registers S_reg and C_reg, store the values $2^{k-1}X$ and 0 each at first, and then

the intermediate values of CSA addition are saved during $(k + l + 1)$ iterations. Like the case of the modular multiplier, a w-bit adder is used for summation of C and S, and the result is saved in C_reg. Also if the MSB of the summation result is 1, then the addition of the value in register N_reg is done again. The final result is calculated and saved in C_reg register after the value in L_reg register is down-counted until it is 0, and C_reg is right shifted simultaneously. In addition, the 4-bit adder is used to calculate the 4-bit value Y which is used for sign estimation and the value $2C + 1$ is directly determined because the LSB of $2C$ is always 0. As a result, when X is $(n + k)$-bits, N is n-bits and the MSB l-bits of N are all 0's, the designed hardware requires $2l + (k + l + 1) + 2\lceil (n + 2)/w \rceil$ clock cycles for a modular reduction in the worst case.

3.3 RSA Crypto-processor

Fig. 3 illustrates the overall architecture of a high speed RSA cryptosystem we designed. It mainly consists of a radix-4 modular multiplier and a modular reducer(Fig. 2), thus can execute modular exponentiation or modular reduction.

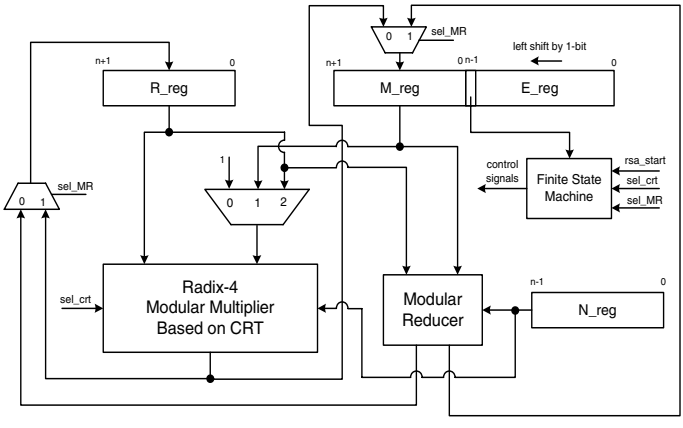


Fig. 3. The overall architecture of proposed RSA crypto-system

When the signal sel_MR is 0, the processor operates modular exponentiation based on L-R Montgomery modular exponentiation as algorithm 4. In this case, the registers N_reg, M_reg and E_reg store the values N, M, e respectively, and the register R_reg stores the mapping constant K at the start and the final result at the end of calculation. The processor uses the radix-4 modular multiplier which supports CRT explained above, thus it executes an n-bit exponentiation if the signal sel_CRT is 0, and two $n/2$-bit exponentiations in parallel when sel_CRT is 1. Because the L-R algorithm is used, the number of modular multiplication is about $2n$(or n) for the worst case and $3/2n$(or $3/4n$) for average cases. Hence the number of clock cycles of modular exponentiation is approximately $1.5(n^2/2 + n^2/w)$ for an n-bit operation and for $1.5(n^2/8 + n^2/4w)$ for $n/2$-bit operations on

average. Furthermore if sel_MR is 1, the processor can execute modular reduction with the modular reducer as in Fig. 2. In this case the two registers M_reg and E_reg store the value X for area efficiency, thus the size of X is $2n+2$ bits, and the result of calculation is saved in the register R_reg. When the MSB of N is $1(l=0)$, the mapping constant K is calculated in $n+3+2\lceil(n+2)/w\rceil$ clock cycles, and if the MSB of P(or Q) is $1(l=n/2)$, modularly reduced ciphertext C_p(or C_q) is calculated in $1.5n+3+2\lceil(n+2)/w\rceil$ clock cycles.

4 Result and Performance

Our RSA crypto-processor has been implemented with the Samsung 0.18um CMOS standard cell library. The synthesis result shows that the critical path delay is about 3.0ns, thus we can expect the processor to operate at a 300MHz clock rate. We use a 32-bit(w=32) Carry Look-ahead Adder(CLA) for high speed operation, and thus the processor needs about 0.84M clock cycles for one 1024-bit modular exponentiation and about 0.25M cycles for two 512-bit modular exponentiations. At a 300MHz clock, the throughput is 365Kbps and 1.233Mbps in each case.

Table 2. Performance comparison of 1024-bit RSA impelementations

	Tech (um)	Gate count	Freq (Mhz)	No. of clocks	Op. time (ms)	Baud rate (Kbps)	CRT	Mod Reduc.
Kwon [13]	0.5	156K	50	n^2	43	45	no	no
Blum [6]	FPGA	-	45	$n^2/2$	12	83	no	no
Cho [22]	-	230K	40	$n^2/2$	13	77	no	no
McIvor [23]	FPGA	-	97	$n^2/4$	2.73	375	yes	no
Ours	0.18	192K	300	$1.5n^2/8$	0.83	1,233	both	yes

Table 2 shows the performance result comparison with those recently reported in the literature. With the results of the table, our design is the fastest RSA crypto-processor published to date. The implementation by Kwon et al. [13] simply uses the basic radix-2 modular multiplier based on CSA, thus its performance is not outstanding. Blum et al. [6] proposed the radix-16 modular multiplier on FPGA to reduce the clock cycles, but they used the dedicated memory in FPGA to precompute and store the multiples of B and N. Cho et al. [22] used the radix-4 multiplier based on CSA and sign estimation technique. However their design uses four n-bit CSAs, thus requires large hardware area. McIvor et al. [23] proposed the modular multiplier based on carry redundant representation. Their design avoids conversion additions at each iteration by using three n-bit CSAs, thus also requires large hardware area, for example 165K for an 1024-bit modular multiplier. Actually, the performance of our implementation in table 2 is for the average case. For the worst case, the baud rate decreases to 274Kbps for non-CRT and 930Kbps for CRT, however our implementation

is still the fastest one. Moreover, if we use two multipliers to do the squaring and multiplication in parallel by R-L exponentiation algorithm, the performance will be improved by 50%. Furthermore, our design can execute both non-CRT and CRT exponentiations selectively, whereas others can execute only one of them. It can also execute modular reduction operation, X mod N for arbitrary 2050-bit and 1024-bit positive integers X and N. Using this function, the 1024-bit mapping constant K can be calculated in 1,059 clock cycles and the 512-bit modularly reduced ciphertext C_p(or C_q) calculated in 1,602 clock cycles if the MSBs of N and P(or Q) are all 1's. Therefore, our design is eminently suitable for high speed RSA cryptosystem.

5 Conclusions

We have presented an architecture of a high-speed RSA crypto-processor based on modified radix-4 Montgomery modular multiplication algorithm and CRT. The implementation of a 1024-bit RSA crypto-system with a 0.18um CMOS standard cell library has been done. The number of clock cycles is 0.84M for non-CRT and 0.25M for CRT style 1024-bit modular exponentiation. The comparison of our implementation with other designs in table 2 shows that the performance of our design is the fastest reported to date in the literature. Our design can also execute modular reduction operation, X mod N for arbitrary 2050-bit and 1024-bit positive integers X and N, which can be used to calculate the Montgomery mapping constant and the modular reduction cipertext in the CRT technique.

References

1. R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signature and public-key cryptosystems," *Comm. ACM*, Vol.21, pp. 120-126, 1978.
2. G.R. Blakley, "A computer algorithm for the product ab modulo m," *IEEE Trans. Comput.*, vol. C-32, pp. 497-500, 1983.
3. E.F. Brickell, "A fast modular multiplication algorithm with application to two-key cryptography," *Proc. CRYPTO'82 Advances Cryptology*, pp. 51-60, 1982.
4. P. L. Montgomery, "Modular multiplication without trial division," *Math. Computation*, Vol. 44, pp. 519-521, 1985.
5. Koc, C.K., Acar, T., Burton S. Kaliski Jr, "Analyzing and Comparing Montgomery Multiplication Algorithms," *IEEE Micro.*, 16(3), pp. 26-33, June 1996.
6. T Blum and C. Paar, "Montgomery modular exponentiation on reconfigurable hardware," *Proc. 14th IEEE Symp. on Comp. Arith.*, pp. 70-77, 1999.
7. N. Takagi, "A radix-4 modular multiplication hardware algorithm for modular exponentiation," *IEEE Trans. Comput.*, vol. 41, pp. 949-956, Aug. 1992.
8. P. Kornerup, "High-radix modular multiplication for cryptosystems," *Proc. 11th IEEE Symp. Comput. Arith.*, Windsor, ON, Canada, June 1993, pp. 277-283.
9. Jin-Hua Hong, Cheng-Wen Wu, "Cellular-Array Modular Multiplier for Fast RSA Public-Key Cryptosystem Based on Modified Booth's Algorithm" *IEEE Trans. on VLSI Systems*, Vol.11, No.3, pp. 474-484, June 2003.
10. C. Couvreur, J. J. Quisquater, "Fast decipherment algorithm for RSA public-key cryptosystem", *Electronics letters*, 18(21), pp. 905-907, 1982.

11. Ciaran McIvor, Maire McLoone, John V McCanny, "A high-speed, low latency RSA decryption silicon core," *IEEE Inter. Symp. On Circuits and Systems(ISCAS)*, vol. 4, pp. 25-28, May 2003.
12. Chung-Hsien Wu, Jin-Hua Hong, Cheng-Wen Wu, "RSA cryptosystem design based on Chinese remainder theorem," *IEEE Proc. Asia and South Pacific Design Automation Conf.(ASP-DA)*, pp. 391-395, 2001.
13. T. W. Kwon, C. S. You, W. S. Heo, Y. K. Kang, and J. R. Choi, "Two Implementation Methods of a 1024-bit RSA Cryptoprocessor Based on Modified Montgomery Algorithm," *IEEE Inter. Symp. On Circuits and Systems(ISCAS)*, vol. 4, pp. 650-53, 2001.
14. Ciaran McIvor, Maire McLoone, John V McCanny, "Fast Montgomery Modular Multiplication and RSA Cryptographic Processor Architectures," *37th Asilomar Conference on Signals, Systems and Computers*, Vol.1, No.7, pp. 379-384, Nov. 2003.
15. A. Cilardo, A.Mazzeo, L. Romano, G.P. Saggese, "Carry-Save Montgomery Modular Exponentiation on Reconfigurable Hardware," *IEEE Proc. on DATE'04*, vol. 03, no. 3, pp. 206-211, 2004.
16. T. Blum and C. Paar, "High-radix Montgomery modular exponentiation on reconfigurable hardware," *IEEE Trans. Comput.*, vol. 50, issue 7, pp. 70-77, July 2001.
17. Peter Kornerup, "Systolic, Linear-Array Multiplier for a Class of Right-Shift Algorithms," *IEEE Trans. on Comp.*, vol. 43, issue 8, pp. 892-898, Aug. 1994.
18. C. D. Walter, "Systolic modular multiplication," *IEEE Trans. on Comp.*, vol. 42, issue 3, pp. 376-378, Mar. 1993.
19. S. E. Eldridge and C. D. Walter, "Hardware implementation of Montgomery's modular multiplication algorithm," *IEEE Trans. on Comp.*, vol. 42, issue 6, pp. 693-699, June 1993.
20. A. D. Booth, "A signed binary multiplication technique," *Q. J. Mech. Appl. Math.*, vol. 4, issue 2, pp. 236-240, 1951.
21. C. K. Koc, C. Y. Hung, "Fast algorithm for modular reduction," *IEE Proc-Comput. Digit. Tech.*, vol. 145, no. 4, pp. 265-271, July 1998.
22. Koon-Shik Cho, Je-Hyuk Ryu, Jun-Dong Cho, "High-speed modular multiplication algorithm for RSA cryptosystem," *IECON'01*, pp. 479-483, 2001.
23. Ciaran McIvor, Maire McLoone, John V McCanny, "Modified Montgomery modular multiplication and RSA exponentiation techniques," *IEE Proc-Comput. Digit. Tech.*, vol. 151, issue 6, pp. 402-408, Nov. 2004.

A High-Speed Square Root Algorithm in Extension Fields

Hidehiro Katou, Feng Wang, Yasuyuki Nogami, and Yoshitaka Morikawa

Graduate School of Natural Science and Technology,
Okayama University, Okayama-shi, 700-8530, Japan
{katou, wangfeng, nogami, morikawa}@trans.cne.okayama-u.ac.jp

Abstract. A square root (SQRT) algorithm in $GF(p^m)$ ($m = r_0 r_1 \cdots r_{n-1} 2^d$, r_i: odd prime, $d > 0$: integer) is proposed in this paper. First, the Tonelli-Shanks algorithm is modified to compute the inverse SQRT in $GF(p^{2^d})$, where most of the computations are performed in the corresponding subfields $GF(p^{2^i})$ for $0 \leqslant i \leqslant d-1$. Then the Frobenius mappings with an addition chain are adopted for the proposed SQRT algorithm, in which a lot of computations in a given extension field $GF(p^m)$ are also reduce to those in a proper subfield by the norm computations. Those reductions of the field degree increase efficiency in the SQRT implementation. More specifically the Tonelli-Shanks algorithm and the proposed algorithm in $GF(p^{22})$, $GF(p^{44})$ and $GF(p^{88})$ were implemented on a Pentium4 (2.6 GHz) computer using the C++ programming language. The computer simulations showed that, on average, the proposed algorithm accelerates the SQRT computation by 25 times in $GF(p^{22})$, by 45 times in $GF(p^{44})$, and by 70 times in $GF(p^{88})$, compared to the Tonelli-Shanks algorithm, which is supported by the evaluation of the number of computations.

1 Introduction

The task of computing square roots (SQRTs) in a finite field $GF(p^m)$ is a problem of considerable importance. Why is the SQRT important? Of course, we may simply be interested in the problem because it's there, but there are also applications to cryptography [1], [2]. In this paper, an efficient algorithm is proposed to compute the SQRT in $GF(p^m)$. First, however, we give a short overview of several SQRT algorithms in a prime field $GF(p)$.

In order to find the values of z in $z^2 = x$ for a given square $x \in GF(p)$, most of known efficient methods are equivalent to or use the same basic ideas as either the Tonelli-Shanks [3] or the Cipolla-Lehmer algorithm [4]. The Cipolla-Lehmer algorithm has the disadvantage that one has to require a quadratic non-residue (QNR) that depends on both p and x, while the QNR needed by the Tonelli-Shanks algorithm can be reused for different x. This study is based on the Tonelli-Shanks algorithm.

The Tonelli-Shanks algorithm was originally devised in $GF(p)$, but can easily be applied in $GF(p^m)$ by simply replacing the operations in $GF(p)$ with those in

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 94–106, 2006.
© Springer-Verlag Berlin Heidelberg 2006

$GF(p^m)$. Note that the operations in $GF(p^m)$ are more expensive than those in $GF(p)$ for $m>1$. For example, a multiplication in $GF(p^m)$ at least requires m^2 multiplications in $GF(p)$ by using the schoolbook method. Therefore, it costs too much to directly apply the Tonelli-Shanks algorithm in extension fields. This is a big motivation of the authors to develop an efficient SQRT algorithm in extension fields $GF(p^m)$.

An extension field $GF(p^m)$ provides us flexibility to choose system parameters, such as characteristic p and extension degree m as well as modular polynomial. The authors have considered the cases of $m = 2^d$ [5] and $m = r2^d$ [6], where r is an odd number and $d \geqslant 0$ is an integer. However, for the application to elliptic curve cryptosystem, we especially restricted $d=1, 2$ in [6]. In this paper, we treat the general case i.e. p is an odd prime number and m has the form of $m=r_0 r_1 \cdots r_{n-1} 2^d$, where r_i is an odd prime number, and $r_i \geqslant r_j$ for $i \leqslant j$, and $d \geqslant 0$ is an integer. In what follows $m = r2^d$ $(r = r_0 r_1 \cdots r_{n-1})$ without further explanation.

The basic idea of the proposed SQRT algorithm is described as follows. If a given element $x \in GF(p^m)$ is not a quadratic residue (QR), i.e. x has no SQRT in the same field, then there is no need to compute its SQRTs. Before SQRT computations, we should thus use Euler's criterion $C^m(x)=x^{(p^m-1)/2}$ to identify whether x is a QR or not, which is called QR test. Since $m=r2^d$, the exponent in Euler's criterion can be factorized into

$$\frac{p^m-1}{2} = e \cdot \frac{p^{2^d}-1}{2}, \quad e = 1 + p^{2^d} + \cdots + (p^{2^d})^{r-1}. \tag{1.1}$$

So, $C^m(x)$ can be evaluated in the following steps:

$$C^m(x) = \bar{x}^{(p^{2^d}-1)/2}, \ \bar{x} = N^m_{2^d}(x) = x^e, \tag{1.2}$$

where $N^m_{2^d}(x)$, the norm of x with respect to the subfield $GF(p^{2^d})$, is always an element in $GF(p^{2^d})$. From Eq.(1.2), i.e. $\bar{x} = x^e$, we have

$$\sqrt{x} = x^{(e+1)/2} \bar{x}^{-\frac{1}{2}}. \tag{1.3}$$

Based on Eq.(1.3), an efficient SQRT algorithm can be presented. In the proposed algorithm, the SQRT algorithm presented in [5], i.e. the MW-ST algorithm, is first modified to compute the inverse SQRT $\bar{x}^{-\frac{1}{2}}$ in a given subfield $GF(p^{2^d})$, where most of the computations in $GF(p^{2^d})$ can be reduced to those in proper subfields $GF(p^{2^{d-i}})$ for $1 \leqslant i \leqslant d$. Therefore, the number of computations for $\bar{x}^{-\frac{1}{2}}$ is far fewer than that for $\sqrt{x}$. In addition, not only the binary method, but also the Frobenius mappings with an addition chain are adopted for $x^{(e+1)/2}$ in Eq.(1.3). For exponentiation, we usually resort to the binary method, however, the number of computations for the Frobenius mapping $\phi(x) = x^p$ is far fewer than that for the binary method in important finite fields, such as optimal extension fields (OEFs) [7], all one polynomial fields (AOPFs) [8] and successive extension fields (SEFs) [9]. Thus, the square root $\sqrt{x}$ can be efficiently computed using the proposed algorithm.

Throughout this paper, A_m, M_m, and ϕ_m denote additions, multiplications, and Frobenius mappings, respectively, in $GF(p^m)$, and $\#A_m$, $\#M_m$, and $\#\phi_m$ denote the respective numbers of these operations.

2 QR Test

Euler's criterion is usually used to identify whether or not a nonzero element $x \in GF(p^m)$ is a QR:

$$C^m(x) = x^{(p^m-1)/2} = \begin{cases} 1, & \text{if } x \text{ is a QR} \\ -1, & \text{if } x \text{ is a QNR} \end{cases}. \tag{2.1}$$

In Eq.(2.1), $x^{(p^m-1)/2}$ can be directly computed by the binary method. However, its complexity is very large for a large p and $m > 1$. Thus, we should present a fast implementation method for the QR test in $GF(p^m)$.

2.1 Fast Implementation of the QR Test

Knowledge prepared for the QR Test. For simplicity without loss of generality, assume $m = \bar{r}\bar{m}$ in this section, where $\bar{r}$ is an odd prime number and $\bar{m}$ is a composite number or 1, and then the exponent in Euler's criterion can be factorized as

$$\frac{p^m - 1}{2} = \left[1 + p^{\bar{m}} + \cdots + (p^{\bar{m}})^{\bar{r}-1}\right] \cdot \frac{(p^{\bar{m}} - 1)}{2}. \tag{2.2}$$

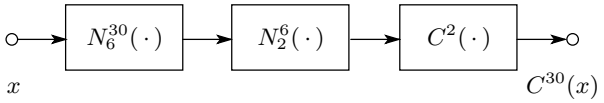
So, $C^m(x)$ can be evaluated in the following steps:

$$C^m(x) = C^{\bar{m}}(\bar{x}),\ \bar{x} = N^m_{\bar{m}}(x) = x^{1+p^{\bar{m}}+\cdots+(p^{\bar{m}})^{\bar{r}-1}}. \tag{2.3}$$

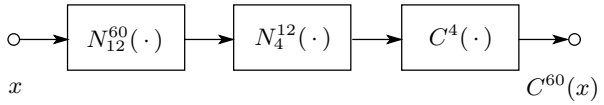
Eq.(2.3) shows that $C^m(x)$ will be reduced into $C^{\bar{m}}(\bar{x})$ by the norm computation of $N^m_{\bar{m}}(x)$. Since x is an element of an extension field $GF(p^m)$ and $\bar{x} = N^m_{\bar{m}}(x)$ is an element of a subfield $GF(p^{\bar{m}})$, the number of computations for $C^{\bar{m}}(\bar{x})$ is thus far fewer than that for $C^m(x)$ i.e. $C^{\bar{r}\bar{m}}(x)$. Note that, for the larger $\bar{r}$, the more reductions in the computations can be obtained. The similar procedures can be applied to the remaining factors of $\bar{m}$. For example, the computation structures of $C^m(x)$ for $m = 30,\ 60$ are shown in Fig.1, in which the symbol $N^{\cdot}_{\cdot}(\cdot)$ denotes the norm computation.

For the norm computation $N^m_{\bar{m}}(x)$, the binary method is usually used, the number of computations of which is huge. In the present study, we therefore adopt the Frobenius mapping for efficiency. Using the ith iteration of the Frobenius mappings $\phi^{[i]}(x) = x^{p^i}$, $N^m_{\bar{m}}(x)$ can be expressed as follows:

$$N^m_{\bar{m}}(x) = \prod_{i=0}^{\bar{r}-1} \phi^{[i\bar{m}]}(x). \tag{2.4}$$



(a) $m = 30 = 2 \cdot 3 \cdot 5$



(b) $m = 60 = 2^2 \cdot 3 \cdot 5$

Fig. 1. Some examples of the structure of $C^m(x)$

In what follows, for simplicity, the ith iteration of the Frobenius mapping is regarded as the Frobenius mapping because their properties are all the same. Since the Frobenius mapping $\phi^{[i\bar{m}]}(x) = x^{p^{i\bar{m}}}$ has the linearity

$$\phi^{[i\bar{m}]}(a\xi + b\zeta) = a\phi^{[i\bar{m}]}(\xi) + b\phi^{[i\bar{m}]}(\zeta), \tag{2.5}$$

for $\xi, \zeta \in GF(p^m)$ and $a, b \in GF(p)$, and since any element $x \in GF(p^m)$ is expressed as a linear combination of the basis, the Frobenius mapping of x can be computed with a small number of computations when the Frobenius mappings of the basis are easily obtained. In particular, the number of computations required for the Frobenius mapping of the basis is negligibly small in OEFs [7], AOPFs [8] and SEFs [9].

Moreover, in order to increase the computation efficiency of $N_{\bar{m}}^m(x)$, we can adopt an addition chain, which reuses the previously obtained values of the Frobenius mappings. In the proposed addition chain, we first compute $\Phi_{\bar{m}}^m$ and $\bar{\Phi}_{\bar{m}}^m$, and then multiply them together to obtain $N_{\bar{m}}^m(x)$, where

$$\Phi_{\bar{m}}^m = \prod_{i=1}^{(\bar{r}-1)/2} \phi^{[(2i-1)\bar{m}]}(x), \quad \bar{\Phi}_{\bar{m}}^m = \prod_{i=0}^{(\bar{r}-1)/2} \phi^{[(2i)\bar{m}]}(x). \tag{2.6}$$

In fact, several addition chains can be used to compute $N_{\bar{m}}^m(x)$. In the proposed addition chain, however, we obtain and save the value of $\Phi_{\bar{m}}^m$ that is necessary for the proposed SQRT algorithm. An example of the addition chain for computing $N_{\bar{m}}^m(x)$ for $\bar{r} = m/\bar{m} = 11$ is shown in Fig.2, where $\phi^{[i\bar{m}]}(\cdot)$ denotes the Frobenius mapping of the input $\cdot$ and $\otimes$ denotes the multiplication in $GF(p^m)$. If $N_{\bar{m}}^m(x)$ is computed directly using Eq.(2.4), then 10 multiplications and 10 Frobenius mappings are required in $GF(p^m)$. As opposed to this, using the addition chain, only 5 multiplications and 5 Frobenius mappings are required in $GF(p^m)$ as shown in Fig.2.

Using the Frobenius mappings with the addition chain, from analogous results in [10], $N_{\bar{m}}^m(x)$ requires that

$$\#M_m = \#\phi_m = \lfloor \log_2(\bar{r}) \rfloor + w(\bar{r}) - 1, \tag{2.7}$$

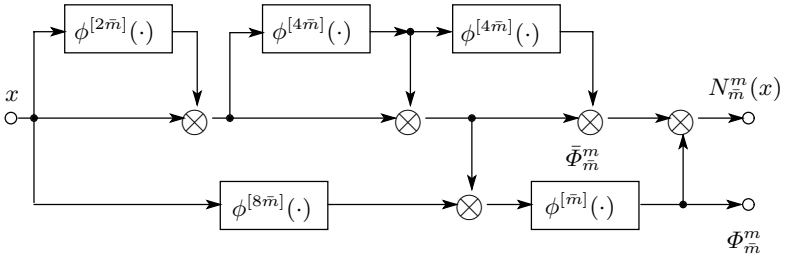


Fig. 2. An example of the addition chain to compute $N^m_{\bar{m}}(x)$ for $\bar{r}=11$

where $\lfloor\cdot\rfloor$ and $w(\cdot)$ denote, respectively, the maximum integer less than $\cdot$ and the Hamming weight of $\cdot$. In the following, the expression $\lfloor\log_2(\cdot)\rfloor + w(\cdot) - 1$ is denoted as $LW(\cdot)$ for convenience.

Degree reduction by odd prime factors in m. In the general case of $m = r_0 r_1 \cdots r_{n-1} 2^d$, let $m_n = 2^d$ and define

$$m_j = r_j \cdots r_{n-1} 2^d, \quad 0 \leqslant j \leqslant n-1, \tag{2.8}$$

and then we have $m_0 = m$. Referring to Fig.1, $C^m(x)$ can be reduced into $C^{2^d}(\bar{x}_n)$ by the following norm computations $\bar{x}_{j+1} := N^{m_j}_{m_{j+1}}(\bar{x}_j)$ with the Frobenius mappings

$$\bar{x}_{j+1} = N^{m_j}_{m_{j+1}}(\bar{x}_j) = \prod_{i=0}^{(r_j-1)} \phi^{[im_{j+1}]}(\bar{x}_j) \in GF(p^{m_{j+1}}), \tag{2.9}$$

where $0 \leqslant j \leqslant n-1$ and $\bar{x}_0 = x$. When implementing the above computations, we first use the proposed addition chain to compute $\Phi^{m_i}_{m_{j+1}}$ and $\bar{\Phi}^{m_i}_{m_{j+1}}$, respectively, and then multiply them together to get $\bar{x}_{j+1}$, where

$$\Phi^{m_j}_{m_{j+1}} = \prod_{i=1}^{(r_j-1)/2} \phi^{[(2i-1)m_{j+1}]}(\bar{x}_j), \ \bar{\Phi}^{m_j}_{m_{j+1}} = \prod_{i=0}^{(r_j-1)/2} \phi^{[(2i)m_{j+1}]}(\bar{x}_j). \tag{2.10}$$

In the QR test, all the values of $\Phi^{m_j}_{m_{j+1}}$ $(0 \leqslant j \leqslant n-1)$ will be saved as shown in Fig.3 because the SQRT algorithm requires them.

From Eq.(2.9), we can see that the computations in a given extension field $GF(p^m)$, i.e. $GF(p^{r_0 \cdots r_{n-1} 2^d})$, can be reduced to those in proper subfields $GF(p^{m_j})$ $(1 \leqslant j \leqslant n-1)$ by the above norm computations. When taking the reduction by the prime factors in m, the reason why the largest r_0 is first performed is that the largest computation reduction will be attained.

Degree reduction by factors 2 in m. Find an integer $T \geqslant 0$ and an odd number s such that

$$p^{2^d} = 2^T s + 1, \tag{2.11}$$

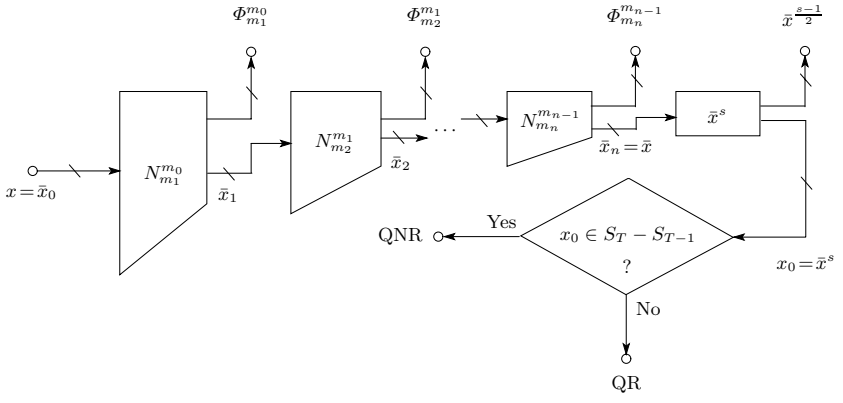


Fig. 3. Schematic diagram for the proposed QR test

and then the multiplicative group $GF^*(p^{2^d})$ is a cyclic group of order $2^T s$. Therefore, its Sylow 2-subgroup S_T is cyclic of order 2^T, and there is a descending chain of subgroups of S_T

$$S_T \supset S_{T-1} \supset \cdots \supset S_2 \supset S_1 = \{\pm 1\} \supset S_0 = \{1\}. \tag{2.12}$$

For any QNR c in $GF(p^m)$, $\bar{c} = c^e$ must be a QNR in $GF(p^{2^d})$, where e is given by Eq.(1.1). It follows that S_T of order 2^T is generated by $c_0 := (\bar{c})^s$, S_{T-1} of order 2^{T-1} is generated by $c_1 := (c_0)^2$, and in general, S_{T-k} of order 2^{T-k} is generated by $c_k := (c_{k-1})^2$ for $k = 1, \cdots, T$. For $1 \leqslant k \leqslant d-1$, we have

$$c_k \in S_{T-k} - S_{T-k-1} \subset GF(p^{2^{d-k}}). \tag{2.13a}$$

For $d \leqslant k \leqslant T-2$, we have

$$c_k \in S_{T-k} - S_{T-k-1} \subset GF(p), \text{ when } 4 \,|\, (\mathrm{p}-1), \tag{2.13b}$$

or

$$c_k \in S_{T-k} - S_{T-k-1} \subset GF(p^2), \text{ when } 4 \nmid (\mathrm{p}-1). \tag{2.13c}$$

For $k = T-1$, we have

$$c_{T-1} \in S_1 - S_0 = \{-1\} \subset GF(p). \tag{2.13d}$$

Decision of QR in $GF(p^m)$. In the conventional QR test, for a nonzero element $x \in GF(p^m)$, we should directly compute $C^m(x) = x^{(p^m-1)/2}$ and identify its result is 1 or -1. In the proposed QR test, the computation of $C^m(x)$ will be reduced into that of $C^{2^d}(\bar{x})$ by a series of norm computations as described above,

where $\bar{x} \in GF(p^{2^d})$. If $\bar{x}$ is a QR in $GF(p^{2^d})$ then x is also a QR in $GF(p^m)$. If $\bar{x}$ is a QNR in $GF(p^{2^d})$ then x is also a QNR in $GF(p^m)$. Since any element in $S_T - S_{T-1}$ must always be a QNR in $GF(p^{2^d})$, and since $x_0 = \bar{x}^s$ must belong to one of sets $S_k - S_{k-1}$ for $1 \leqslant k \leqslant T$, we only need to identify whether $x_0 \in S_T - S_{T-1}$ as shown in Fig.3. In OEF and AOPF, we can know whether $x_0 \in S_T - S_{T-1}$ from the form of x_0. This almost does not need any computation. For example, if an element in an OEF $GF(p^4)$ $(4 \,|\, (p-1))$ has the form of $(0, a, 0, 0)$ or $(0, 0, 0, a)$ $(a \neq 0 \in GF(p))$, then x is a QNR.

2.2 Complexity of the QR Test

In the conventional QR test, $x^{(p^m-1)/2}$ is directly computed using the binary method, which requires that

$$\#M_m = LW\left(\frac{p^m - 1}{2}\right). \tag{2.14}$$

In the proposed QR test, we first implement a series of norms $N_{m_j}^{m_{j+1}}(\cdot)$ for $0 \leqslant j \leqslant n-1$ to get $\bar{x}$ using the Frobenius mappings with the addition chain. From Eq.(2.7), the norm computations $N_{m_j}^{m_{j+1}}(\cdot)$ for $0 \leqslant j \leqslant n-1$ in total require the sum of

$$\#\phi_{m_j} = \#M_{m_j} = LW(r_j), \quad 0 \leqslant j \leqslant n-1. \tag{2.15}$$

Then, we compute $\bar{x}^s$. In order to avoid the overlapping computations in the QR test and the SQRT computation, we first compute $\bar{x}^{(s-1)/2}$, and then take its square to get $\bar{x}^{s-1}$, finally multiply $\bar{x}^{s-1}$ by $\bar{x}$ to get $\bar{x}^s$ (*see* Fig.3). When $4 \,|\, (p-1)$, the computation of $\bar{x}^s$ requires (*see* Appendix G of [11])

$$\#M_{2^d} = LW(s_1) + LW\left(\frac{s_1 - 1}{2}\right) + \frac{(d+4)(d-1)}{2} + (d-1)LW\left(\frac{p-1}{4}\right), \tag{2.16a}$$

$$\#\phi_{2^d} = \frac{d(d-1)}{2}. \tag{2.16b}$$

When $4 \nmid (p-1)$, the computation of $\bar{x}^s$ requires (*see* Appendix G of [11])

$$\#M_{2^d} = LW(s_1) + LW\left(\frac{s_1 - 1}{2}\right) + \frac{(d+6)(d-1)}{2} + (d-1)\left[LW\left(\frac{p-3}{4}\right) + LW\left(\frac{p+1}{2}\right)\right], \tag{2.17a}$$

$$\#\phi_{2^d} = \frac{d(d-1)}{2}. \tag{2.17b}$$

where the nonnegative integer T_1 and the odd integer s_1 are satisfied with $p^2 = 2^{T_1} s_1 + 1$.

Finally, we identify whether $\bar{x}^s = x_0 \in S_T - S_{T-1} \subset GF(p^{2^d})$ from the form of x_0, which almost do not need any compuation as described in Section 2.1.

3 The Proposed SQRT Algorithm in $GF(p^m)$

3.1 The Proposed SQRT Algorithm in $GF(p^m)$

From the norm definition Eq.(2.3) and (2.9), we have

$$\bar{x}_{j+1} = N^{m_j}_{m_{j+1}}(\bar{x}_j) = \bar{x}_j^{(p^{m_{j+1}})^{r_j-1}+\cdots+p^{m_{j+1}}+1}. \tag{3.1}$$

Dividing the both sides by $\bar{x}_j\bar{x}_{j+1}$ and then taking the SQRT, we have

$$(\bar{x}_j)^{-\frac{1}{2}} = (\Phi^{m_j}_{m_{j+1}})^{\frac{1+p^{m_{j+1}}}{2}} \cdot (\bar{x}_{j+1})^{-\frac{1}{2}},\ 0 \leqslant j \leqslant n-1, \tag{3.2}$$

where $\Phi^{m_j}_{m_{j+1}}$ is given by Eq.(2.10). It follows that

$$(\bar{x}_0)^{-\frac{1}{2}} = (\bar{x}_n)^{-\frac{1}{2}} \cdot \prod_{j=0}^{n-1} (\Phi^{m_j}_{m_{j+1}})^{\frac{1+p^{m_{j+1}}}{2}}, \tag{3.3}$$

where $\bar{x}_0 = x$ and $\bar{x}_n = \bar{x}$. Therefore, we have

$$x^{\frac{1}{2}} = \bar{x}^{-\frac{1}{2}} \cdot x \cdot \prod_{j=0}^{n-1} (\Phi^{m_j}_{m_{j+1}})^{\frac{1+p^{m_{j+1}}}{2}}. \tag{3.4}$$

When implementing the exponentiation of $\frac{1+p^{m_{j+1}}}{2}$, which can be expressed as

$$\frac{p^{m_{j+1}}+1}{2} = \left(1 + p + \cdots + p^{(m_{j+1})-1}\right) \cdot \frac{p-1}{2} + 1, \tag{3.5}$$

we apply the Frobenius mappings with the addition chain to the part in parenthesis, and apply the binary method to $(p-1)/2$ followed by a multiplication.

Based on Eqs.(3.4), the SQRT in $GF(p^m)$ can be efficiently computed using the following algorithm via the Frobenius mappings with the addition chain:

The Proposed SQRT Algorithm over $GF(p^m)$ c

INPUT: An odd prime number p and an integer m and a random nonzero element $x \in GF(p^m)$, where $m = r_0 r_1 \cdots r_{n-1} 2^d$ (r_i: odd prime, $d \geqslant 0$: integer).

OUTPUT: A SQRT $z = x^{\frac{1}{2}} \in GF(p^m)$ such that $z^2 \equiv x \pmod{p}$, or "UNSOLVABLE", if no such solution exists.

PRECOMPUTATION FOR GIVEN p AND m: Factorize the order of the multiplicative group in $GF(p^{2^d})$ as shown in Eq.(2.11).

1. If $x = 1$ then return 1. Otherwise, execute the proposed QR test as described in Section 2.1, if the input x is a QR then save the values of $\bar{x}$, $\Phi^{m_j}_{m_{j+1}}$ for $0 \leqslant j \leqslant n-1$, else if the input x is a QNR then return "UNSOLVABLE". From Eq.(2.9), we know $\bar{x} = \bar{x}_n = x^e \in GF(p^{2^d})$, where e is given by Eq.(1.1).
2. $\alpha_j \leftarrow \Phi^{m_j}_{m_{j+1}}$ for $0 \leqslant j \leqslant n-1$.
3. $z \leftarrow \bar{x}^{-\frac{1}{2}}$.
4. About the computations for $x \cdot \prod_{j=0}^{n-1} (\Phi^{m_j}_{m_{j+1}})^{\frac{1+p^{m_{j+1}}}{2}}$ in Eq.(3.4):
   (a) $\beta_j \leftarrow \alpha_j^{\frac{1+p^{m_{j+1}}}{2}}$ for each $0 \leqslant j \leqslant n-1$, $\gamma \leftarrow \prod_{j=0}^{n-1} \beta_j$.
5. $z \leftarrow zx\gamma$.
6. Return(z).

3.2 The Complexity of the Proposed SQRT Algorithm

In the proposed SQRT algorithm, step 1 is just the QR test whose complexity has been evaluated in Section 2.2. Since α_j has been computed in the QR test, we do not evaluate the complexity of step 2.

In step 3, the value of $\bar{x} \in GF(p^{2^d})$ has been computed in the QR test, recomputation by step 3 is thus not necessary. We only need to modify the MW-ST algorithm [5] to compute $\bar{x}^{-\frac{1}{2}}$ in $GF(p^{2^d})$. As described in Section 4.5.4 of [11], in the case of $4 \mid (p-1)$ and $d>1$, step 3 on average requires that

$$\#M_1 = \frac{T^2 - T + d^2 + 5d}{4} - \frac{2^{d-1}(d^2+3d)}{2^{T-1}}, \tag{3.6a}$$

$$\#M_{2^{d+1-n}} = \frac{n^2+3n-2}{4}, \quad n = 1,\ 2,\ \cdots,\ d. \tag{3.6b}$$

In the case of $4 \nmid (p-1)$ and $d>2$, step 3 on average requires that

$$\#M_2 = \frac{T^2 - T + d^2 + 3d - 4}{4} - \frac{2^{d-2}(d^2+d-2)}{2^{T-1}}, \tag{3.7a}$$

$$\#M_{2^{d+1-n}} = \frac{n^2+3n-2}{4},\ n = 1,\ 2,\ \cdots,\ d-1. \tag{3.7b}$$

In step 4, we compute the exponentiation of $\alpha_j^{\frac{1+p^{m_{j+1}}}{2}}$ for each $0 \leqslant j \leqslant n-1$ and multiply them together. Using binary method and the Frobenius mappings with the addition chain, step 4 requires the sum of the following equations:

$$\#\phi_{m_j} = \#M_{m_j} = LW(m_{j+1}),\ 0 \leqslant j \leqslant n-1, \tag{3.8a}$$

$$\#M_{m_j} = LW(\frac{p-1}{2}) + 1,\ 0 \leqslant j \leqslant n-1, \tag{3.8b}$$

$$\#M_{m_j} = r_{j-1},\ 1 \leqslant j \leqslant n-1. \tag{3.8c}$$

Since $x,\ \gamma \in GF(p^m)$ and $z \in GF(p^{2^d})$, step 5 requires that

$$\#M_m = 1, \tag{3.9a}$$

$$\#M_{2^d} = r_0 r_1 \cdots r_{n-1}. \tag{3.9b}$$

3.3 Complexity of the Tonelli-Shanks Algorithm

Assuming $p^m = 2^{\bar{T}}\bar{s} + 1$, from Eq.(2.11), it follows that

$$\bar{T} = T,\ \bar{s} = s \cdot \left[1 + (p^{2^d}) + \cdots + (p^{2^d})^{r_0 r_1 \cdots r_{n-1} - 1}\right]. \tag{3.10}$$

Based on the result in Section 6.3.3 of [11], the average complexity of the Tonelli-Shanks algorithm over given $GF(p^m)$ is that

$$\#M_m = \frac{1}{4}(\bar{T}^2 + 7\bar{T} - 16) + \frac{1}{2^{\bar{T}-1}} + LW\left(\frac{\bar{s}-1}{2}\right) + 2. \tag{3.11}$$

4 Computer Simulations

In this section, we set the characteristic p and the extension degree m of $GF(p^m)$ as follows:

$$p = 2^{28} + 625, \quad m = 22 = 11 \times 2; \tag{4.1a}$$

$$p = 11969, \quad m = 44 = 11 \times 2^2; \tag{4.1b}$$

$$p = 89, \quad m = 88 = 11 \times 2^3. \tag{4.1c}$$

The conventional QR test, the proposed QR test, the Tonelli-Shanks algorithm and the proposed SQRT algorithm over the extension fields $GF(p^{22})$, $GF(p^{44})$ and $GF(p^{88})$ are implemented on a Pentium4 (2.6 GHz) computer using the C++ programming language, where $GF(p^{22})$, $GF(p^{44})$ and $GF(p^{88})$ are constructed as SEFs that are viewed as extension fields of degree 2, 4, and 8 over $GF(p^{11})$, respectively, and $GF(p^2)$, $GF(p^4)$, $GF(p^8)$ and $GF(p^{11})$ are constructed as OEFs.

Based on Eqs.(2.11) and (4.1), we can get the values of T, s, and then from Eq.(3.10), we can know the values of $\bar{T}$ and $\bar{s}$. Inputting p, m, T, s, $\bar{T}$ and $\bar{s}$ to Eqs.(2.14), (2.15), (2.16), (3.6), (3.8), (3.9), and (3.11), we explicitly evaluate the complexity of the two QR tests and the two SQRT algorithms over $GF(p^{22})$, $GF(p^{44})$ and $GF(p^{88})$, such as ϕ_4, ϕ_8, $\#M_1$, $\#M_2$ and so on as shown in the column A of Table 1. Table 3 shows the number of algebraic operations required for ϕ_i and M_j, where $i = 4,\ 8,\ 22,\ 44,\ 88$ and $j = 1,\ 2,\ 4,\ 8,\ 22,\ 44,\ 88$. According to the data in Table 3, we can get $\#A_1$ and $\#M_1$ in the column A of Table 2.

Table 1. Computational Complexity

Field	Method	A. Complexity							
$GF(p^m)$		$\#\phi_4$	$\#\phi_8$	$\#\phi_m$	$\#M_1$	$\#M_2$	$\#M_4$	$\#M_8$	$\#M_m$
$m = 22$ $p = 2^{28} + 625$	Conventional QR test	–	–	–	–	–	–	–	921
	Proposed QR test	–	–	5	–	132	–	–	5
	Tonelli-Shanks algorithm	–	–	–	–	–	–	–	927
	Proposed SQRT algorithm	–	–	1	6	23	–	–	34
$m = 44$ $p = 11969$	Conventional QR test	–	–	–	–	–	–	–	910
	Proposed QR test	1	–	5	–	–	144	–	5
	Tonelli-Shanks algorithm	–	–	–	–	–	–	–	928
	Proposed SQRT algorithm	–	–	2	16	1	23	–	21
$m = 88$ $p = 89$	Conventional QR test	–	–	–	–	–	–	–	833
	Proposed QR test	–	3	5	–	–	–	149	5
	Tonelli-Shanks algorithm	–	–	–	–	–	–	–	843
	Proposed SQRT algorithm	–	–	3	11	2	1	23	12

Table 2. Computational Amount and Running Time (CPU: Pentium4, (2.6GHz))

Field $GF(p^m)$	Method	A. Amount $\#A_1$	A. Amount $\#M_1$	B. Time [μs]
$m = 22$ $p = 2^{28} + 625$	Conventional QR test	425502	465105	3.4×10^4
	Proposed QR test	2574	3255	2.6×10^2
	Legendre-Kronecker-Jacobi symbol	-	-	2.8×10^2
	Tonelli-Shanks algorithm	428274	468135	3.5×10^4
	Proposed algorithm	15754	17311	1.2×10^3
$m = 44$ $p = 11969$	Conventional QR test	1721720	1800890	1.3×10^5
	Proposed QR test	11188	12834	0.9×10^3
	Tonelli-Shanks algorithm	1755776	1836512	1.4×10^5
	Proposed algorithm	40010	42097	3.1×10^3
$m = 88$ $p = 89$	Conventional QR test	6377448	6523223	4.8×10^5
	Proposed QR test	46624	50155	3.2×10^3
	Tonelli-Shanks algorithm	6454008	6601533	4.9×10^5
	Proposed algorithm	93176	95885	0.7×10^4

Table 3. The Number of Algebraic Operations Required for ϕ_i and M_j, where $i =$ 4, 8, 22, 44, 88 and $j = 1$, 2, 4, 8, 22, 44, 88

	ϕ_4	ϕ_8	ϕ_{22}	ϕ_{44}	ϕ_{88}	M_2	M_4	M_8	M_{22}	M_{44}	M_{88}
$\#A_1$	–	–	–	–	–	2	12	56	462	1892	7656
$\#M_1$	3	7	20	40	80	5	19	71	505	1979	7831

From the column A of Table 1, we see that $\#M_m$ required in the proposed SQRT algorithm is much smaller than that in the Tonelli-Shanks algorithm, primarily because in the proposed SQRT algorithm, most of multiplications in a given extension field are replaced by those in its proper subfields.

Inputting 560,000 random QRs, the running time for the two QR tests and the two SQRT algorithms were measured in the column B of Table 2. The column A of Table 2 shows that, using the proposed QR test, the numbers of computations in $GF(p^{22})$, $GF(p^{44})$ and $GF(p^{88})$ show about 130-fold reductions, compared to the conventional QR test. Using the proposed SQRT algorithm, the numbers of computations in $GF(p^{22})$, $GF(p^{44})$ and $GF(p^{88})$ show 25-fold, 45-fold and 70-fold reductions for $p = 2^{28} + 625$, 11969 and 89, respectively, compared to the Tonelli-Shanks algorithm, where the number of computations is the number of the algebraic operations required in $GF(p)$. The computer simulations show that, on average, the proposed QR test accelerates the QR test by about 130 times in $GF(p^{22})$, $GF(p^{44})$ and $GF(p^{88})$, compared to the conventional QR test. The computer simulations also show that, on average, the proposed algorithm accelerates the SQRT computation by 25 times in $GF(p^{22})$, by 45 times in $GF(p^{44})$,

and by 70 times in $GF(p^{88})$, compared to the Tonelli-Shanks algorithm, which is supported by the evaluation of the number of computations.

5 Conclusion

We have presented an efficient SQRT algorithm over $GF(p^m)$ based on Eqs.(3.4). Although the main idea of the proposed SQRT algorithm is based on the Tonelli-Shanks algorithm, in the proposed SQRT algorithm over $GF(p^m)$, most of the computations required in extension fields $GF(p^m)$ can be reduced to those in proper subfields $GF(p^{m_j})$ for $1 \leqslant j \leqslant n$ and $GF(p^{2^{d-i}})$ for $1 \leqslant i \leqslant d$. To the contrary, all the computations required for the Tonelli-Shanks algorithm over $GF(p^m)$ must be executed in extension fields $GF(p^m)$. In addition, the proposed SQRT algorithm can reuse the intermediate data of the QR test, such as $\Phi_{m_{j+1}}^{m_j}$ for $0 \leqslant j \leqslant n-1$. The computer simulations showed that, on average, the proposed algorithm accelerates the SQRT computation by 25 times in $GF(p^{22})$, by 45 times in $GF(p^{44})$, and by 70 times in $GF(p^{88})$, compared to the Tonelli-Shanks algorithm, which is supported by the evaluation of the number of computations. Therefore, we can conclude that the proposed SQRT algorithm over $GF(p^m)$ is very efficient.

References

1. D. Hankerson, A. Menezes, and S. Vanstone, Guide to Elliptic Curve Cryptography, Springer, 2003.
2. K. Kurosawa, T. Ito, and M. Takeuchi, "Public key cryptosystem using a reciprocal number with the same intractability as factoring a large number," Cryptologia, vol. 12, no. 4, pp. 225-233, 1988.
3. A. Tonelli, "Bemerkung über die Auflösung quadratischer Congruenzen," Göttinger Nachrichten, pp. 344-346, 1891.
4. M. Cipolla, "Un metodo per la risolutione della congruenza di secondo grado," Rendiconto dell'Accademia Scienze Fisiche e Matematiche Napoli, Ser. 3, vol. IX, pp. 154-163, 1903.
5. F. Wang, Y. Nogami, and Y. Morikawa, "An efficient square root computation in finite fields," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E88-A, No. 10, pp. 2792-2799, 2005.
6. F. Wang, Y. Nogami, and Y. Morikawa, "A fast square root computation using the Frobenius mapping," Proc. ICICS 2003, LNCS 2836, pp. 1-10, 2003.
7. D. V. Bailey and C. Paar, "Optimal extension fields for fast arithmetic in public-key algorithms," Proc. Crypto. 1998, pp. 472-4854, 1998.
8. Y. Nogami, A. Saito, and Y. Morikawa, "Finite extension field with modulus of all-one polynomial and representation of its elements for fast arithmetic operations," Trans. IEICE vol. E86-A, no. 9, pp. 2376-2387, 2003.
9. J.L. Fan and C. Paar, "On efficient inversion in tower fields of characteristic two," Proc. ISIT 1997, pp. 20, 1997.
10. D. V. Bailey, Computation in optimal extension fields, A thesis submitted to the Faculty of the Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science, 2000.

11. F.Wang, "Efficient square root algorithms over extension fields $GF(p^m)$, A thesis submitted to the Graduate School of Natural Science and Technology of Okayama University for the Degree of Doctor in Engineer, 2005.

Appendix: Legendre-Kronecker-Jacobi Symbol

We can use Legendre-Kronecker-Jacobi symbol for a QR test. Table 4 shows the simulation result. As shown in Table 4, the calculation time when we use Legendre-Kronecker-Jacobi symbol is faster than that of our proposed QR test. Our proposed method can use the calculation results of QR test in the following SQRT calculation; however, the case of Legendre-Kronecker-Jacobi symbol cannot. Therefore, we can carry out the QR test with Legendre-Kronecker-Jacobi symbol, but the QR test dose not contribute to the following SQRT calculation.

Table 4. Calculation Time over $GF(p^{22})$. (CPU: Pentium4, (2.6GHz)).

Field	Method	Time [μs]
$m = 22$	Conventional QR test	3.4×10^4
	Proposed QR test	2.6×10^2
	Legendre-Kronecker-Jacobi symbol	1.8×10^2
$p = 2^{28} + 625$	Tonelli-Shanks algorithm	3.5×10^4
	Proposed algorithm	1.2×10^3

The Smallest ARIA Module with 16-Bit Architecture

Sangwoon Yang[1], Jinsub Park[2], and Younggap You[2]

[1] National Security Research Institute,
161 Gajeong-dong, Yuseong-gu, Daejeon, South Korea
sangwoon@etri.re.kr
[2] Chungbuk National University,
San 12, Gaeshin-dong, Cheongju, Chungbuk, South Korea
{jspark, yy}@hbt.chungbuk.ac.kr

Abstract. This paper presented the smallest hardware architecture of the ARIA block cipher algorithm. A 128-bit data block was divided into eight 16-bit blocks to reduce the hardware size. The 16-bit architecture allowed two S-Boxes and 16-bit diffusion operation. We proposed a design for the substitution layer and the memory block. The proposed round key generator processed a 16-bit block of a 128-bit round key for three cycles. The proposed ARIA module with a 128-bit key comprised 6,076 equivalent gates using a 0.18-μm CMOS standard cell library. It took 88 clock cycles to generate four initial values for a round key and 400 clock cycles to en/decrypt 128-bit block data. The power consumption of 16-bit ARIA was only 5.02 μW at 100 kHz 1.8V.

Keywords: Cryptography, ARIA, Low Power Design.

1 Introduction

The notion of ubiquitous computing is becoming more general in human beings' lives. Radio Frequency Identification (RFID) products are used to identify products at sup-ply chains, to enter buildings, and to pay bus fares. Also, many experts expect that a ubiquitous sensor network will cover various fields from military to nuclear power plant applications. These ubiquitous devices will address heavier information transfers. It will need increasingly better protection on information from eavesdroppers.

The block cipher algorithm named Academy, Research Institute, Agency (ARIA) was announced as the Korean standard block cipher algorithm in Dec. 2004 [1]. The ARIA is a 128-bit block cipher with an involution Substitution Permutation Network (SPN). Since 2004, just few studies have been published on the hardware architecture for the ARIA in contrast to Advanced Encryption Standard (AES). The 128-bit architecture with one round looping structure was proposed for high-throughput [4]. It used 1,491 slices and 16 BlockRAMs. It yielded throughput of 496 Mbps at 46.5 MHz. Because it was optimized for high-throughput, its applications were limited such as server systems for network security. A compact 32-bit design has been implemented and simulated

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 107–117, 2006.
© Springer-Verlag Berlin Heidelberg 2006

with 0.25-μm CMOS standard cell library [6]. It used 13,893 gates and power consumption was 61.46 mW at 71MHz. However, it was yet small enough for practical use.

In this paper, we propose a 16-bit architecture for ARIA block cipher which hardware size is under 10,000 gates. We focus on a power restricted system such as RFID tags. A circuit design for cryptography depends on a target system. Especially, the power restricted system needs lots of consideration to design the circuit due to many constraints which the proposed compact architecture should meet such as area and power consumption. This paper is organized as follows. In section II, ARIA algorithm is briefly described. In section III, the proposed ARIA implementation is presented. Section IV explains simulation results of the proposed ARIA design. Finally, concluding remarks are in Section V.

2 ARIA Algorithm

The ARIA algorithm is a symmetric cipher which is designed for the same specification of the AES. It encrypts and decrypts 128-bit blocks of data using a secret key. A key size can be one of 128, 192, or 256 bit. The ARIA comprises two blocks: a round function for en/decryption of incoming data and a key scheduler including an initialization and round key generation.

The round function is based on the involution SPN that a round function iteratively operates 12, 14 and 16 rounds in accordance to the key width. It consists of the Substitution layer, the Diffusion layer and the AddRoundKey layer except the last round. In the last round, the AddRoundKey layer replaces the Diffusion layer as shown Figure 1. The output value of each layer is defined as a state which consists of a 128-bit memory. The AddRoundKey layer is to perform an exclusive-OR operation that adds the 128-bit round key to the state, where the round key is generated each round. The Substitution layer uses four S-boxes: S_1, S_2, S_1^{-1}, and S_2^{-1}. An irreducible polynomial of Equation (1) is used to define the field. The S-box S_1 is defined to be an affine transformation of the multiplicative inverse, x^{-1} on $GF(2^8)$ as shown in Equation (2) of Figure 1. The S-box S_2 uses the affine transformation of x^{247} on $GF(2^8)$ as shown in Equation (3) of Figure 1. The Diffusion layer is an involutional 16×16 binary matrix of Equation (4) in Figure 1.

$$m(x) = x^8 + x^4 + x^3 + x + 1 \tag{1}$$

The key scheduler performs two processes: the key initialization and the round key generation. In the key initialization process, four 128-bit values, W_0, W_1, W_2 and W_3 are generated from the master key through a three-round 256-bit Feistel cipher. In the round key generation process, the key generation block issues round keys combining the four Wi values. The round keys for encryption differ from the round keys of decryption. The round key sequences are strictly in order; all keys but the first and the last are sent to the Diffusion layer.

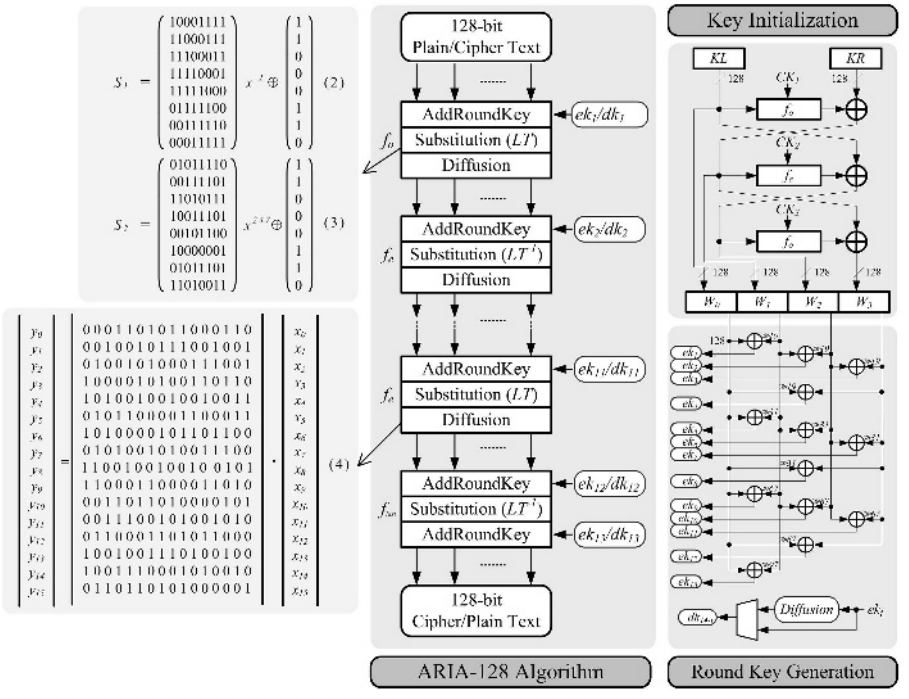


Fig. 1. ARIA-128 algorithm

3 Compact Architecture for ARIA Algorithm

The ARIA hardware can be tailored to a smaller die-size because the algorithm can be implemented on various platforms with different bus widths. It can be restructured suitable for 16-bit, 32-bit and 128-bit platforms. The Diffusion layer of ARIA is hard to be designed without registers to store an intermediate value for the operation. Registers used in the implementation make hardware size bigger, power consumption higher, and process-ing time increase.

The goal of our design of the ARIA is to minimize the size of hardware and average power consumption. This paper presents a 16-bit ARIA design with compact architecture shown in Figure 2. We modify the data path employing a dual addressing in [7], making a structure with a single addressing scheme. A memory based on the static RAM is modified by using the gated clock method. The round function consists of the Upper/Lower AddRoundKey layer, the Substitution layer employing a multiplicative inverter on the composite field $GF(((2^2)^2)^2)$, and the Diffusion layer designing a 16×16 binary matrix. The round key generator of [7] is also modified in here.

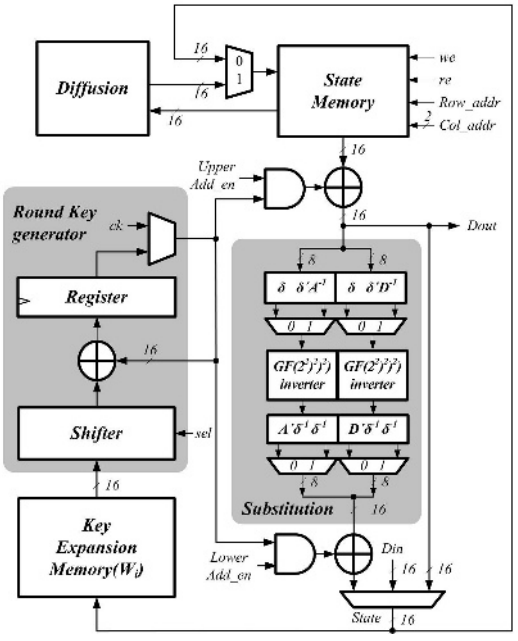


Fig. 2. Block diagram comprising a data-path, a round key generator, a memory block, a substitution layer, and a diffusion layer are modified for the 16-bit operation of ARIA

3.1 Data Path

The ARIA is an involution SPN where an encryption and decryption use the identical data path as shown in Figure 1. We modify a data path of round function and key scheduler to reduce a processing time. In our design with 16-bit-width data path, the round function takes 32 cycles except the last round of 48 cycles: 24 cycles of the round key generation and 8 cycles of the diffusion. In the case that the round function was designed in a 32-bit data path in [6], it had taken 16 cycles instead of 24 cycles.

In the round function, the forms of the data path are a AddRoundKey, Substitution, and Diffusion flow for encryption and a Diffusion, Substitution, and AddRoundKey flow for decryption. Although this idea had been adopted in [7], we improve the data path as shown in Figure 2. AddRoundKey layer and Substitution layer are operated simultaneously. AddRoundKey layer can be placed selectively at the both sides of the Substitution layer. For encryption, the AddRoundKey block is enabled in the upper side and is masked in the lower one. The operation for decryption is in the reverse order.

The previous round key generator in [7] takes 4 cycles: 3 cycles for reading and 1 cycle for initializing the register. We eliminate an initialization process of the register. The combination of the four Wi values yields round keys. The proposed key scheduler generates the round keys by using a shifter for rotating a

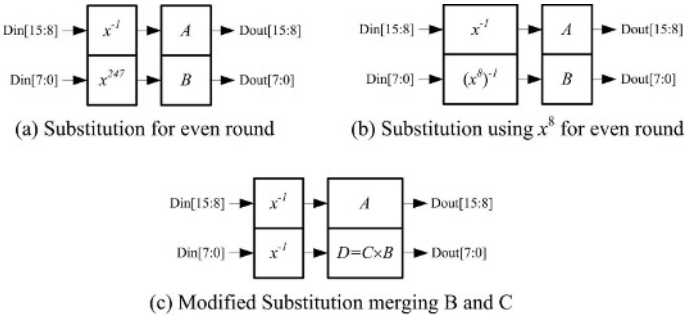


Fig. 3. Representation conversion process of x^{247} with x^{-1}

word, a 16-bit exclusive-OR gate, and registers as shown in the left side of Figure 2. The shifter rotates data from Key Expansion Memory by the predetermined number of positions. The rotated data goes through the exclusive-OR gate with the previous output value stored in the register.

3.2 Substitution Layer

The S-Box is the most critical component for controlling the size and the speed of the hardware implementation. The Substitution layer of ARIA comprises four S-Boxes, S_1, S_1^{-1}, S_2, and S_2^{-1}. We introduce two methods for the optimization of S-boxes. In the first method, only two tables are used for the optimization of S-boxes instead of the general one using 4 tables. S-boxes, S_1 and S_2, are pre-calculated with the terms, x^{-1} and x^{247}, two tables and two affine transformations in the specification of ARIA.

In $GF(2^8)$ with the polynomial $f(x) = x^8 + x^4 + x^3 + x + 1$, the term x^{247} can be represented as an-other form with the multiplicative inverse, x^{-1}. The term x^{247} is equivalent to x^{-8} in the field. If we introduce a new table C, x^{-8} can also be represented as $C \cdot x^{-1}$ because of the linearity of the field. Table C is a 8×8 binary matrix as shown in Equation (5). As a result, the equation of S-box S_2, $B \cdot x^{247} \oplus b$, becomes $(B \cdot C) \cdot x^{-1} \oplus b$. Here, a new table, $D = B \cdot C$, is computed by Equation (6). We can redefine S-box S_2: $x \rightarrowtail D \cdot x^{-1} \oplus b$. Figure 3 shows the process of the representation conversion of x^{247} with x^{-1}.

The second method employs the compact implementation of x^{-1} using a composite field proposed by Satoh [12]. For the optimization of a multiplicative inverse, a composite field $GF(((2^2)^2)^2)$ is used [2]. The isomorphism functions δ and δ^{-1} are located at the both sides of the S-Boxes, and can be merged with an affine transformation [9]. ARIA has two kinds of substitutions. The substitution used in even round functions consists of S_1, S_2, S_1^{-1}, and S_2^{-1} as shown in Figure 4a. The substitution used in even round functions consists of S_1^{-1}, S_2^{-1}, S_1, and S_2 as shown in Figure 4b. The two substitution blocks can use the common multiplicative inverter with multiplexers as shown in Figure 4c. Sharing the common factors of the isomorphism function and the affine transformation reduces the number of exclusive-or gates and the critical path delay as shown in Figure 4d.

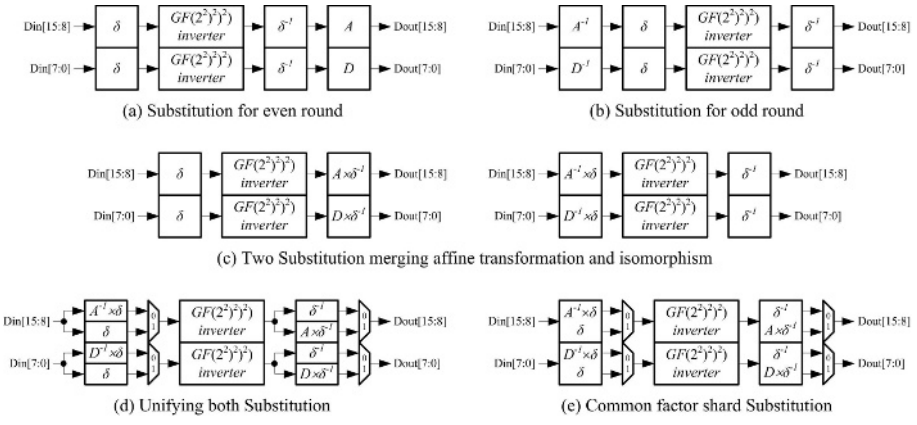


Fig. 4. Unified substitution layer architecture

Table 1. Equivalent gates of Substitution layer

Block	Equivalent gates
Substitution in this paper	740
Lookup table scheme [7]	3,493

The hardware size of the proposed Substitution layer is 79 % smaller than that of the lookup table scheme of [7]. The comparison between the proposed memory and the dual addressing scheme is described in Table 1. The Substitution layer in this paper is implemented manually using a 0.18-μm CMOS standard cell library in the gate level. One gate is equivalent to a 2-way NAND.

3.3 Diffusion Layer

In the proposed design, we employ a 16×16 binary matrix which diffuses 16-bit data at one time. It takes 8 cycles for diffusing a 128-bit block. This architecture can re-duce the number of gates compared to the previous one with a 4×16 binary matrix in [7] dividing the original 16×16 matrix by 4. It is because the 4×16 binary matrix in [7] needs an additional 128-bit register with intermediate values for the diffusion operation.

We delve into the diffusion logic in Equation (4) and find that each bit, $S_{0,c}$, in 16 8-bit blocks in State Memory can be processed in a bit serial fashion with small logics. It can be replaced with the result $S'_{0,c}$, in the original position without an immediate value. Additional registers in Diffusion layer in [7] can be eliminated. In this design, the Substitution layer makes a progress 8 cycles after a complete 128-bit state is made in the Diffusion layer. State Memory for diffusion operation is modified and re-quires 96 2-input exclusive-or gates. A size of 197 EGs is obtained through an automatic synthesis, and it is about 93 %

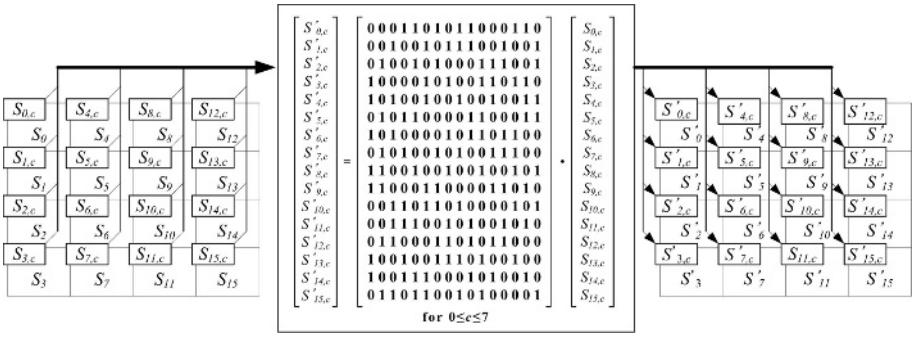


Fig. 5. The operation of proposed diffusion layer

smaller than that of the 4×16 binary matrix. Eventually, this feature of the modified Diffusion layer makes the design more efficient in power.

3.4 Memory Block

The proposed ARIA design has a Key Expansion Memory block storing four 128-bit values of W_0, W_1, W_2, and W_3 for the round key generator and a State Memory block for the state of the round function. Because the memory consumes the power in the system mainly due to the clock, it makes an ARIA design difficult to be efficient in power.

Memory blocks designed in this paper are represented as a static single port memory. Clock gating is applied to reduce power consumption. In this design, clock gating is an efficient technique because only one 16-bit register in the memory block can be processed at a time and the others are disabled to save power. The memory used in State Memory and Key Expansion Memory is modified to save power as shown in Figure 6.

Key Expansion Memory is the biggest of the compact ARIA design. The operation of Key Expansion Memory stores four values in initialization, and outputs the saved data in encryption and decryption. Because data in memory only changes in initialization, we employ a latch for lower power consumption and smaller size than a flip-flop.

4 Results of Simulation

This section reports the simulation results of a 16-bit version: hardware size, throughput, and power consumption. Circuit were described and verified in Verilog-HDL. The simulation is based on the transistor level using a 0.18-μm and 0.25-μm CMOS standard cell library from MagnaChip Semiconductors. A synthesis is per-formed on Design Compiler from Synopsys. The power consumption is an important factor of the design for a restricted power supply.

We explain how to measure the overall consumption of power, and then analyze each block of the design. The power consumption is measured at 100 kHz

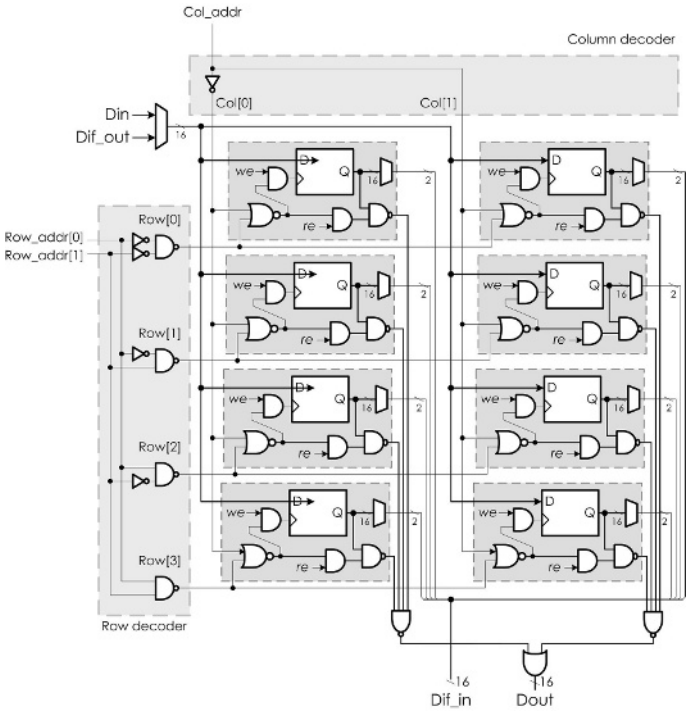


Fig. 6. The block diagram of the State Memory

and 2.5 V power supply. It is an average power dissipated in initializing the round key and encrypting the 128-bit plain-text. All the values in the paper are from the simulations on the transistor level estimation, which was performed by using NanoSim from Synopsys Inc.

A total hardware size is estimated to be 6,076 equivalent gates. Table 2 shows the number of equivalent gates of each block of the 16-bit ARIA design. The proposed 16-bit ARIA is about 83 % smaller than the 128-bit system [5] and about 45 % smaller than 32-bit one [7]. The biggest circuit is the memory block: Key Expansion Memory requires 2,605 EGs (42.8 %) and State Memory 1,280 EGs (21.0 %). Although memory blocks have a big size, we can reduce the size by employing a latch than a flip flop. The Substitution layer requires 740 EGs about 79 % smaller compared to the previous one. The Substitution layer using a composite field inverter is more efficient than one using LUT [7]. It is effective to merge the isomorphism function and affine transformation. Sharing the common factor is another feature to enhance effectiveness. The diffusion block which uses one 16×16 binary matrix generated by synthesis tools requires only 192 EGs. The control block has 722 EGs. The complexity of the control block in-creases as the word size decreases.

The power consumption of 16-bit ARIA is 5.02 μW at 100 kHz. The Diffusion layer in the proposed design consumes 0.16μW. The number of data transitions

Table 2. Equivalent gates and power consumption of each component in 16-bit ARIA

Block	Equivalent Gates (%)	Power consumption μW (%)
Key Expansion Memory	2,605 (42.8)	0.26 (05.17)
State Memory	1,280 (21.0)	0.65 (13.01)
Substitution	740 (12.1)	2.38 (47.39)
Control circuit	722 (12.3)	0.95 (18.54)
Round Key Generator	421 (06.9)	0.42 (08.36)
Diffusion	192 (03.1)	0.16 (03.23)
Upper AddRoundKey	58 (00.9)	0.09 (01.89)
Lower AddRoundKey	58 (00.9)	0.12 (02.41)
Total	6,076 (100)	5.03 (100)

Table 3. Performance comparison for ASIC implementations

Algorithm	Cycles /Init.	Cycles /Round.	Total Cycles	Process	Equiv. Gate (# of NAND)	Area (mm^2)	Aver. Power Consumption (μW@100kHz)
Proposed 16-bit architecture							
ARIA-128	88	32	400	0.18-μm	6,076	0.18-μm	5.02
				0.25-μm	6,840	0.25-μm	12.17
[7] 32-bit architecture							
ARIA-128	63	28	356	0.25-μm	13,893	0.25-μm	86.5 (estimated)
[5] 128-bit architecture							
ARIA-128	6	1	12	Xilinx VirtexE-1600	1,491 Slices 16 BRAM	-	-
Re-synthesized 128-bit architecture [5]							
ARIA-128	6	1	12	0.25-μm	47,669	0.82	-
ARIA-192	6	1	14				-
ARIA-256	6	1	16				-

of a 128-bit register in the diffusion block of the dual addressing scheme is 48 for 12 rounds. We eliminate the immediate register, and make the Diffusion layer operate on low power. 47.39% of power is consumed only in Substitution layer's data transitions in the design. If we use the low power S-Box circuit of Satoh [12], the power consumption can decrease further. Only in 8 cycles of 24 cycles for the AddRoundKey and Substitution, the data transition is effective. Masking of input data in the Substitution layer for the other cycles is to reduce the number of data transitions in the logic. The power consumption of the control block takes 18.54 % of the total. Why we do not concern about energy consumption is that the power consumption per clock cycle is limited although the total energy consumption of an operation might be larger [11].

The throughput is not the most important factor in a low power environment. Most applications in the low power environment use a simple value only. An

RFID uses an ID, which is from one to hundred bits. A sensor node does not require a high-throughput in a ubiquitous sensor network. The throughput of this design at 100 kHz is 32 Kbps. According to a synthesis report, the 16-bit ARIA design allows 15 MHz as the maximum frequency, and the maximum throughput is 4.8 Mbps after initialization. The proposed ARIA needs 88 clock cycles for initialization and 400 clock cycles for en-cryption/decryption. The round key generator takes 24 clock cycles because generating a 16-bit word of one round key needs 3 cycles. A concurrent operation of AddRoundKey and Substitution requires 24 clock cycles in encryption. The Diffusion layer takes 8 clock cycles. One round needs 32 clock cycles except the last round which uses 48 clock cycles. It is because a AddRoundKey layer is more needed in the last round. Although an 8-bit architecture has a bit small size, the 16-bit one meeting those constraints can support a higher data throughput.

5 Conclusion

This paper presents a compact architecture for ARIA. We modify the data path eliminating the diffusion function to generate a decryption round key. A 16-bit bus-width was employed in order to reduce the hardware size. It requires only an quarter size of 128-bit processing. The S-Box is very small due to the composite field inverter and the representation conversion of x^{247} of S-Box using x^{-1}. The memory module employing a gated clock reduces the power consumption effectively.

The proposed design of ARIA meets the constraints of low-power environment. The 16-bit ARIA comprises 6,076 equivalent gates. The power consumption of the 16-bit ARIA is 5.02-μW at 100 kHz 1.8V. Measurements in this work are estimated by simulation using a 0.18-μm CMOS process. It needs 88 clock cycles to generate initial values for a round key and 400 clock cycles to en/decrypt a 128-bit block data. Its maximum throughput is 4.8 Mbps at 15 MHz. The proposed architecture is the small-est among ARIA design published so far as shown in Table 3.

Acknowledgments. This work was supported by the Regional Research Centers Program of the Ministry of Education & Human Resources Development in Korea.

References

1. NSRI: ARIA Algorithm Specification, http://www.nsri.re.kr/ARIA/doc/ARIA-specification.pdf, 2004 (in Korean)
2. A. Satoh, S. Morioka, K. Takano and S. Munetoh, "A compact Rijndael hardware architec-ture with S-Box optimization," *Proceeding Advances in Cryptology ASIACRYPT 2001*, pp. 239–254, Dec. 2001.
3. A. Biryukov, C. D. Canni‘ere, J. Lano, S. B. Ors, and B. Preneel: Security and performance analysis of ARIA, http://www.nsri.re.kr/ARIA/doc/ARIA-COSICreport.pdf, Jan. 2003

4. D. Kwon , J. Kim, S. Park, S. Sung, Y. Sohn, J. Song, Y. Yeom, E. Yoon, S. Lee, J. Lee, S. Chee, D. Han and J. Hong, "New block cipher: ARIA," *Proceeding ICISC2003*, pp. 432–445, Nov. 2003
5. J. Park, Y. Yun, Y.-D. Kim, S. Yang, T. Chang and Y. You, "Design and implementation of ARIA cryptic algorithm," *Journal of IEEK*, vol. 42-SD, no. 4, pp. 22–36, Apr. 2004 (in Korean)
6. J. Park, Y. Yun, S. Kim, Y.-D. Kim, and Y. You, "A crypto-processor for security PDA systems," *Proceeding ISOCC2005*, pp. 11–14, Oct. 2005
7. J. Park, Y.-D. Kim, S. Yang, and Y. You, "Low power compact design of ARIA block cipher," *Proceeding ISCAS2006*, pp. 313–316, May. 2006
8. M. Feldhofer, S. Dominikus and J. Wolkerstorfer, "Strong authentication for RFID systems using the AES algorithm," *Proceeding CHES2004*, LNCS 3156, pp 357–370, Aug. 2004
9. A. Satoh and S. Morioka, "Unified hardware architecture for 128-bit block cipher AES and Camellia," *Proceeding CHES 2003*, LNCS 2779, pp 304–318, Sep. 2003
10. W. Nebel and J. Mermet, *Low Power Design in Deep Submicron Electronics*, Kluwer Aca-demic Publishers, 1997
11. M. Feldhofer, J. Wolkerstorfer and V. Rijmen, "AES implementation on a grain of sand," *IEE Proceeding, Information Security*, vol. 152, pp. 13–20, Oct. 2005
12. S. Morioka and A. Satoh, "An optimized S-Box circuit architecture for low power AES design," *Proceeding CHES 2002*, LNCS 2523, pp. 172–186, Aug. 2002.
13. A. Satoh and S. Morioka, "Hardware-focused performance comparison for the standard block cipher AES, Camellia, and Triple-DES," *Proceeding ISC 2003*, LNCS 2851, pp. 252-266, Oct. 2003.

A Simpler Sieving Device: Combining ECM and TWIRL

Willi Geiselmann[1], Fabian Januszewski[2], Hubert Köpfer[1], Jan Pelzl[3], and Rainer Steinwandt[4]

[1] Institut für Algorithmen und Kognitive Systeme, Fakultät für Informatik, Am Fasanengarten 5, Universität Karlsruhe (TH), 76128 Karlsruhe, Germany
geiselma@ira.uka.de
[2] Mathematisches Institut II, Fakultät für Mathematik, Englerstraße 2, Universität Karlsruhe (TH), 76128 Karlsruhe, Germany
fabian.januszewski@math.uni-karlsruhe.de
[3] Horst Görtz Institute for IT-Security, Ruhr University of Bochum, Universitätsstraße 150, 44780 Bochum, Germany
pelzl@crypto.rub.de
[4] Department of Mathematical Sciences, Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33431, USA
rsteinwa@fau.edu

Abstract. A main obstacle in manufacturing the TWIRL device for realizing the sieving step of the Number Field Sieve is the sophisticated chip layout. Especially the logic for logging and recovering large prime factors found during sieving adds significantly to the layout complexity. We describe a device building on the Elliptic Curve Method (ECM) that for parameters of interest enables the replacement of the complete logging part in TWIRL by an off-wafer postprocessing. The postprocessing is done in real time, leaving the total sieving time basically unchanged.

The proposed device is an optimized ECM implementation building on curves chosen to cope with factor sizes as expected in the output of TWIRL. According to our preliminary analysis, for the relation collection step expected for a 1024-bit factorization our design is realizable with current fab technology at very moderate cost. The proposed ECM engine also finds the vast majority of the needed cofactor factorizations. In summary, we think the proposed device to enable a significant decrease of TWIRL's layout complexity and therewith its cost.

Keywords: RSA, NFS, ECM, cryptanalytic hardware.

1 Introduction

Lacking paramount theoretical progress in the design of algorithms for factoring integers, in recent years significant efforts have been invested into designing special purpose hardware for factoring. Having in mind a record factorization of a 1024-bit RSA-modulus, at the moment the (General) Number Field Sieve (NFS) seems to be the algorithm of choice, and consequently several proposals

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 118–135, 2006.
© Springer-Verlag Berlin Heidelberg 2006

for using dedicated hardware to speed up the time-dominating steps of the NFS have been put forward. In particular, for the NFS' linear algebra step significant progress has been achieved [2,14,8,5,6]—for the most recent designs thinking of a practical implementation for the 1024-bit case does not seem too far-fetched.

On the other hand, even the most recent designs that have been proposed for implementing the sieving step of the NFS—the other time-dominating part of the NFS—rely on highly non-trivial technological assumptions: for mesh-based devices along the lines of [2,7,9,10] no practically promising parameter set for the 1024-bit case has been proposed so far, the TWIRL device [23,15] involves rather large chip sizes along with a non-trivial layout, and for SHARK [4] the actual implementation of the underlying butterfly transport system is technologically challenging. From a practical point of view, finding modifications or alternatives to the existing proposals that are of comparable performance but closer to existing fab technology is highly desirable.

In this contribution we describe a device that enables the removal of the complete "diary circuitry" from TWIRL (see [23, Appendix A.7]). This part of TWIRL adds significantly to the layout complexity, but unfortunately seems vital for the recovery of *largish* prime factors found during sieving. While our approach does not solve the problem of TWIRL being a wafer-scale design, it enables a significant reduction of the complexity of the layout. We show that for relevant parameter choices it is feasible to omit the recording of prime factors during sieving, and to recover them in a postprocessing phase with an optimized implementation of the Elliptic Curve Method (ECM): Our design builds on elliptic curves chosen to cope with factor sizes as expected in the output of TWIRL. Specifically, for the parameters currently considered as realistic for the 1024-bit case, according to our preliminary analysis the proposed ECM engine can be implemented on chips of standard size at very moderate cost. Additionally, the suggested device computes almost all the required cofactor factorizations. As the required computations can be performed in real time, the overall sieving time remains basically unchanged. In summary, we think the proposed ECM engine to enable a significant decrease of TWIRL's layout complexity and therewith its implementation cost. Clearly, the major issue that TWIRL is a wafer-scale design is not solved by our contribution. However, we think that simplifying the structure of this wafer-scale circuitry is a significant contribution towards a prototype of TWIRL. The chips proposed for our design are fairly standard ASIC chips, whose production should encounter no major obstacles. For other sieving designs like SHARK [4] it seems worthwhile, too, to explore—e. g., bandwidth—savings that can be achieved through the use of a high performance ECM engine as described here. The idea of not having to store prime factors found during sieving seems to be more than of pure conceptual relevance. Due to the very modest hardware requirements, our approach may in fact be of independent interest for NFS implementations on "classical" hardware platforms.

Further related work. We do not claim the idea of using a dedicated ECM circuitry within the relation collection step of the NFS to be a novel one, e. g., the idea of a parallel ECM engine for smoothness testing is mentioned in [2,15]

already, and [20] proposes a dedicated ECM engine in connection with the SHARK device. The design presented below differs significantly from that in [20] resp. SHARK, however. To the best of our knowledge our proposal is the first one that aims at coping with the performance requirements needed to substitute parts of the TWIRL architecture through an ECM engine and has been explored at this level of detail. Our discussion includes, e. g., an issue like efficient primality testing within ECM, and we are not aware of a "simple reparametrization" of the design in [20] that meets our performance requirements. However, we deem it an interesting question for future work to what extent the ideas below can also facilitate an implementation of SHARK—and vice versa how to integrate ideas from [20] with TWIRL.

Should we do better? It may look tempting to expand the role taken by ECM in the relation collection, and we considered a hybrid design where an ECM engine replaces the algebraic TWIRL device altogether. However, so far we could not identify a concrete design, where this approach yields a device that is easier to implement than the known proposals.

2 Preliminaries and Parameters

Providing an introduction to the NFS is beyond the scope of this contribution, and we refer to [22] for a survey and to the standard reference [13] for details of the NFS. Section 2.1 gives a brief summary of those aspects of the (relation collection step of the) NFS, that are crucial for our design. Similarly, for an introduction to ECM we refer to [16], but Section 2.2 recalls those aspects that are relevant for describing our design.

2.1 Choice of NFS Parameters

In this paper we deal with the NFS' *relation collection* step only. At this we are given two univariate polynomials $f(x), g(x) \in \mathbb{Z}[x]$ sharing a common root M modulo the integer N to be factored:

$$f(M) \equiv g(M) \equiv 0 \pmod{N}.$$

Everything related to $f(x)$ is usually referred to as belonging to the *algebraic side*, and analogously for everything related to $g(x)$ we use the term *rational side*. We are specifically interested in the case of a 1024-bit factorization with $N = 1350\ldots7563$ being the number *RSA-1024* as specified in [12]. To this aim we assume the polynomials $f(x)$ and $g(x)$ to be chosen of degree 5 and 1, respectively, as proposed in [23, Appendix B.2] resp. [15, Appendix B]. Both of these specific polynomials are non-monic, and accordingly we define two homogeneous polynomials $N_{\mathbf{a}}(a,b) := |b^5 \cdot f(a/b)|$ (of degree 5) and $N_{\mathbf{r}}(a,b) := |b \cdot g(a/b)|$ (of degree 1). Now the goal of the relation collection step can be phrased as finding pairs of coprime integers (a,b) with $b > 0$ such that both $N_{\mathbf{a}}(a,b)$ and $N_{\mathbf{r}}(a,b)$ split into a product of primes smaller than a smoothness bound $y_{\mathbf{a}}$ resp. $y_{\mathbf{r}}$. At

this, we do not require a "full splitting", but allow that $N_{\mathbf{a}}(a,b)$ resp. $N_{\mathbf{r}}(a,b)$ contain up to $\ell_{\mathbf{a}}$ resp. $\ell_{\mathbf{r}}$ "large" prime factors below some semi-smoothness bound $z_{\mathbf{a}}$ resp. $z_{\mathbf{r}}$.

The question of the *concrete* choice of these parameters and of the required *sieving range* for the pairs (a,b) obviously arises. As our device aims at an integration with TWIRL, for the analysis of the 1024-bit case we adopt the parameters from [23] resp. [15, Table 10]:

- On the rational side, the smoothness and semi-smoothness bounds are chosen as $y_{\mathbf{r}} = 3.5 \cdot 10^9$ and $z_{\mathbf{r}} = 4 \cdot 10^{11}$, respectively.
- On the algebraic side, the smoothness and semi-smoothness bounds are chosen as $y_{\mathbf{a}} = 2.6 \cdot 10^{10}$ and $z_{\mathbf{a}} = 6 \cdot 10^{11}$, respectively.
- $2 + 2$ large primes are used, i. e., both on the rational and on the algebraic side we allow $\ell_{\mathbf{r}} = \ell_{\mathbf{a}} = 2$ large prime factors.
- For the sieving region $-A < a \leq A$, $0 < b \leq B$ we choose $A = 5.5 \cdot 10^{14}$ and $B = 2.7 \cdot 10^8$.

Another figure that is important for analyzing the 1024-bit case in more detail, is the rate at which (a,b)-candidates are output by TWIRL: To be of practical interest, the required test of simultaneous smoothness of $N_{\mathbf{r}}(a,b)$ and $N_{\mathbf{a}}(a,b)$ should be completed in real-time and not require extensive buffering. Following [23, Appendix A.7], we can expect that a fraction of $\gamma := 2 \cdot 10^{-11}$ of the sieve locations will be output by TWIRL as interesting (a,b)-candidate. With the 1024-bit parameters in [23], TWIRL handles $s_{\mathbf{a}} = 2^{15}$ sieve locations per clock cycle, running at a clock speed of $f := 1$ GHz. If the candidates were output in regular time intervals, we thus had to handle about

$$\Delta' := f \cdot \gamma \cdot s_{\mathbf{a}} \approx 655$$

sieve locations per second with our ECM engine. At the beginning of the sieving phase more candidates are to be expected, but at this early stage we can simply store the candidates in some buffer, and it seems safe to design our ECM engine to handle $\Delta := 1000$ candidate (a,b)-pairs/second. The unlikely case of a buffer overflow can be tolerated and is compensated by simply performing slightly more sieving. For each of these candidates we have to determine the norms $N_{\mathbf{r}}(a,b), N_{\mathbf{a}}(a,b)$ and compute the prime factorizations hereof if the smoothness conditions are met. With the mentioned choices for $N_{\mathbf{r}}$, $N_{\mathbf{a}}$, A and B the numbers to be factored can be expected to have no more than 216 bit on the rational and 350 bit on the algebraic side.

It is certainly acceptable to allow the ECM engine to fail for a small fraction of the "good" (a,b)-candidates. Based on software simulations, for the NFS parameters considered here, we decided to use 84 curves on each side, and we estimate that $< 0.5\%$ of the "good" (a,b)-candidates get lost. To further decrease the error, one could implement a version of our design applying more curves and pay for this with an increased chip area. E. g., allowing 128 curves, on the algebraic side, we encountered only 75 integers out of 10^6 that would be semi-smooth on the algebraic side, but could not be factored by the curves used in our design. The chosen number of 84 curves seems to be an acceptable trade-off between error rate and chip area.

2.2 The Elliptic Curve Method (ECM)

To factor a composite $n \in \mathbb{N}$ with Lenstra's ECM method, one starts by choosing a random point $P = (x : y : 1)$ on a random elliptic curve in Weierstrass normal form. Assuming $\gcd(6, n) = 1$, in affine coordinates that is a solution of a *Weierstrass equation*

$$y^2 = x^3 + ax + b \tag{1}$$

modulo n, where $\gcd(4a^3 + 27b^2, n) = 1$. Then the solutions of (1) modulo a prime divisor q of n constitute a finite abelian group E_q if we adjoin a point $O = (0 : 1 : 0)$ at infinity (serving as identity element). Lenstra's idea was to apply Pollard's "$p - 1$"-method of factorization [21] but to work in E_q instead of $(\mathbb{Z}/q\mathbb{Z})^*$. The key point here is that the order of E_q depends on a, b and q, i. e., not only on q. This allows multiple tries with different curves for the same composite n. Formulae for the group law for curves in Weierstrass form are given in [16]. One can expect that with a certain positive probability—depending on the number of B-smooth integers in $(q + 1 - 2\sqrt{q}; q + 1 + 2\sqrt{q})$—the order $\#E_q$ is B-smooth for a fixed bound B (and $\#E_{q'}$ is not B-smooth for prime divisors $q' \neq q$ of n). Then the computation of

$$\prod_{p<B} p^{e_p} \cdot P, \qquad e_p = \lfloor \log_p(q + 1 + 2\sqrt{q}) \rfloor, \tag{2}$$

yields the prime divisor q of n.

After the computation of (2) we can run a so-called *continuation* as in the case of Pollard's method of factorization. This resembles Shanks' Baby step-Giant step method. If we choose a second bound $C > B$, a sufficient condition for a continuation of ECM to find q is then that $\#E_q$ be divisible by a prime p, $B \leq p < C$, and $\#E_q/p$ be B-smooth. We will explain the choice of the continuation in Section 3.2 below.

3 Splitting the Norms in Real Time

The goal of our device is to check whether the candidates (a, b) received (from a diary-free TWIRL) satisfy the rational and algebraic semi-smoothness conditions. If yes, the factorizations of $N_{\mathbf{r}}(a, b)$ and $N_{\mathbf{a}}(a, b)$ are to be computed. These computations are to be done in real time, so that no excessive buffering or a significant increase of the time spent for relation collection becomes necessary. The proposed design naturally splits into two parts which process the received values in a pipelined manner: a *Rational Factorization Unit* and an *Algebraic Factorization Unit*.

3.1 Basic Components

The Rational Factorization Unit. It is distributed on four identical chips. Each of these chips consists of four parts: The first part, a controller, handles the I/O, distributes the tasks to the other parts on the chip and stores the results. The

second part receives the (a, b) pairs from TWIRL through the controller and calculates the rational norm $N_{\mathbf{r}}(a, b)$, where b remains constant during the sieving of one line. After some preprocessing for each line, calculating the rational norm (which can be expected to have no more than 216 bit) therefore reduces to the evaluation of an affine polynomial, i. e., one multiplication and one addition. These operations are performed using a 16 bit adder, some logic for the multiplication and four registers up to a length of 216 bit. Some 10,000 transistors should be sufficient to realize this part. The norm $N_{\mathbf{r}}(a, b)$ is forwarded to the trial division pipeline that performs the divisions with all primes/prime powers $\leq 100,000$ and reports the factors found to the controller. The remaining factor of $N_{\mathbf{r}}(a, b)$ (with ≈ 200 bit on average) is then forwarded, through the controller, to the fourth and largest part of the chip, the ECM engine. Details of the trial division pipeline and the ECM engine which factors the remaining part, if the semi-smoothness conditions are met, will be discussed in Section 4.2. If an (a, b)-pair turns out to satisfy the rational semi-smoothness bounds, it is—along with a report encoding the factors found on the rational side—forwarded to the subsequent algebraic factorization unit.

The Algebraic Factorization Unit. It is realized with five chips and has the same structure as its rational counterpart, but it considers only those (a, b)-pairs where the rational semi-smoothness conditions turned out to be fulfilled already. As on the rational side, first the norm $N_{\mathbf{a}}(a, b)$ has to be computed, which for a constant b amounts to 5 multiplications and 5 additions (using Horner's rule). On the algebraic side we have to deal with larger integers than on the rational side, and we can expect $N_{\mathbf{a}}(a, b)$ to have no more than 350 bit. However only those (a, b)-pairs fulfilling the rational semi-smoothness conditions are forwarded to this device. Therefore on average no more than 75 inputs are expected per chip and second. This reduced number of inputs compensates the larger multiplication/division time. It should be possible to realize this part with 14,000 transistors. Again, the trial division pipeline and an ECM engine are used to split $N_{\mathbf{a}}(a, b)$ into prime factors.

3.2 Design of the ECM Engine

From a mathematical point of view, our use of ECM on the algebraic and the rational side is the same: We build on an identical set of 84 curves over $\mathbb{Q}$, and for the second phase we use the same improved standard continuation. Due to the different operand sizes, however, the hardware implementation on the algebraic and the rational side is different. In the remaining part of this section we focus on theoretical parameter choices underlying our design. Section 4 discusses aspects related to a hardware implementation.

Choice of Curves. Any elliptic curve over a field K with $\operatorname{char} K \nmid 6$ is isomorphic to a curve in Weierstrass form (1). Referring to ideas of Suyama, in [19] Montgomery suggests to use different families of curves to speed up ECM. In fact, it is possible to choose a random elliptic curve E_q over $\mathbb{F}_q$ and to guarantee

$d \mid \#E_q$ for any fixed $d \in \{4, 12, 16\}$[1]. Basing on software experiments of one of the authors [11], for our purposes the family proposed by Atkin and Morain in [1] with $d = 16$ seems to perform best.

In [19] Montgomery proposed to use elliptic curves of the form

$$sy^2 = x^3 + tx^2 + x, \qquad \gcd(s(t^2 - 4), n) = 1. \tag{3}$$

Equation (3) usually is referred to as *Montgomery form* or *Chudnovsky form*, and [19] gives efficient formulae for the computation on curves of this form. One major advantage is that these formulae enable the evaluation of the product (2) without the use of inversions. Additionally the order of the torsion subgroup of a curve of this type is known a priori to be divisible by 4. Besides being useful for factoring this also implies that not all elliptic curves can be isomorphically transformed into this form. The family with $d = 16$ proposed by Atkin and Morain may be transformed into Montgomery form. Generation of random curves of this family involves computations on an elliptic curve which cannot be transformed into Montgomery form (due to the reason mentioned above). The aspects of curve generation are discussed in the following paragraph.

Generation of Curves. For a composite n we generate a random elliptic curve modulo n and a point on it as follows. According to [1], $S := (12 : 40 : 1)$ is a point of infinite order on $E : Y^2 = X^3 - 8X - 32$ over $\mathbb{Q}$. Therefore we get for every $r \in \mathbb{N}$ a different point $(x : y : 1) := r \cdot S \in E$. According to [1] and [11] every (x, y) yields an elliptic curve E_r in Montgomery form for which we can guarantee $16 \mid \#E_r$ as follows:

Define $\alpha := (x - 9)/(x + y + 16)$ and $\beta := 4\alpha + 1$. Then $\tilde{x} := 4\beta - (\beta^2 - 1)(\beta + 1)^2$, $\tilde{z} := 4(\beta + 1)^2$ yield a point $(\tilde{x} : \tilde{y} : \tilde{z})$ of infinite order on the elliptic curve E_r given by $sY^2 = X^3 + tX^2 + X$ with $s, t, \tilde{y}$ also depending on α. For actual computations we do not need to know the values of $\tilde{y}, s, t$ but the value of

$$\frac{t+2}{4} = \frac{(\beta^2+1)^4}{16\beta^2(\beta^2-1)^2}.$$

This term is the only one, despite of $\tilde{x}$ and $\tilde{z}$, that is needed by the arithmetic described by Montgomery in [19] for computations on E_r. For our design it seems practical to precompute the values of β over $\mathbb{Q}$ for $r \in \{1, \ldots, 84\}$ and to proceed then modulo n to compute $\tilde{x}, \tilde{z}$ and $(t + 2)/4$. Numerators and denominators of the coordinates (x, y) grow rather quickly over $\mathbb{Q}$. Since we restricted our setting to the "first" 84 curves of this family, the numerators and denominators of the values of β we need to handle are bounded in size by 17 kbit.

Modular reduction of such large numbers takes time. Therefore we partition the set of the values of α as follows. The i-th partition consists of the curves for $r \in \{i, i + 14, i + 28, i + 42, i + 56, i + 70\}$. To factor n we then choose randomly[2]

[1] By a celebrated result of Mazur we cannot expect more, because of 16 being the maximal order arising for a torsion subgroup of an elliptic curve over $\mathbb{Q}$.

[2] In a hardware implementation the needed random value can be determined, e. g., using an LFSR.

$i \in \{1, \ldots, 14\}$ and apply the curves $i, i+14, i+28, i+42, i+56, i+70$ to n (in that order, because the absolute values of the numerator and denominator of β are smaller for r small). During the computation on an elliptic curve we may precompute the modular reduction needed for the next curve.

First Phase. We choose $B = 402$ so that there are 79 primes $p_1, \ldots, p_{79} < B$. Additionally we choose $v = 530$, $e_i := \lfloor \log_{p_i} v \rfloor$ and compute $k = \prod_{i=1}^{79} p_i^{e_i}$. Then the first phase consists in computing $Q = k \cdot P$. This can be done efficiently without inversions using Montgomery's formulae [19]. Finally we do a gcd computation to check if this computation already yields a nontrivial divisor d of n (this is necessary for our primality test, see below). If this does yield a nontrivial divisor of n we can continue the computation modulo $n' := n/d$ with $Q' := Q \pmod{n'}$.

We restricted v to the value of 530 instead of $v = (q+1+2\sqrt{q})/16$ corresponding to Equation (2) with $y_{\mathbf{r}} \leq q \leq z_{\mathbf{a}}$ since the probability that a greater power of $p_i^{e_i}$ divides $\mathrm{ord}(P)$ was seen to be very low in simulations for candidates in our setting, see [11]. So the reduction of v leads to a considerable speedup of ECM.

Continuation. As second phase for ECM we choose the improved standard continuation as described in [3, Section 3.2] and [11] which may be realized using Montgomery's arithmetic. We choose $C = 9680$ and let $Q = k \cdot P$ denote the result of the first phase. In the continuation we compute sequentially the points $2 \cdot Q, 4 \cdot Q, \ldots, 2t \cdot Q$. Then we can write every prime q with $B \leq q < C$ in the form $q = 2(st \pm r) + 1$, with $1 \leq r \leq t$. This enables us to test whether $q \cdot Q = O$ holds modulo a divisor d of n by checking whether $2r \cdot Q = \pm(2st+1) \cdot Q$ holds modulo d. If $l \cdot Q = (X_l : - : Z_l)$ for $l \in \mathbb{N}$ this can be done by checking if $d \mid X_{2r} Z_{2st+1} - X_{2st+1} Z_{2r}$ which in an implementation amounts to computing $\gcd(n, X_{2r} Z_{2st+1} - X_{2st+1} Z_{2r})$. Instead of computing every gcd separately we may compute $d_0 = \gcd(n, T_{0,0})$ where $T_{0,0} = \prod_{r,s} (X_{2r} Z_{2st+1} - X_{2st+1} Z_{2r})$ for all relevant pairs (r, s) which correspond to a prime q (or a pair of primes) as above.

If d_0 is composite this implies that several prime divisors were found at once. In that case we may try to recover those prime divisors by splitting the product $T_{0,0}$ into subproducts $T_{1,0} = \prod_{\text{some } r,s} (X_{2r} Z_{2st+1} - X_{2st+1} Z_{2r})$, $T_{1,1} = T_{0,0}/T_{1,0}$ followed by a computation of $d_1 = \gcd(d_0, T_{1,0})$, $d_2 = \gcd(d_0, T_{1,1}) = d_0/d_1$. If $1 \neq d_1 \neq d_0$ we successfully reconstructed a finer factorization $d_0 = d_1 d_2$. This process may be repeated recursively to eventually compute a decomposition of d_0 into prime factors. We refer subsequently to this structure as the *tree structure* and call the $T_{i,j}$ the *product tree*.

In our design we compute first $T_{1,0}$ and $T_{1,1}$ and then $T_{0,0} = T_{1,0} T_{1,1}$. To have a chance of 50% to split a composite d_0 into nontrivial d_1, d_2 we (once for all) choose the (r, s)-pairs which contribute to $T_{1,0}$ randomly from all pairs. Due to the parameters chosen, there are 1116 primes in the continuation so that there are at most $\lceil \log_2 1116 \rceil = 11$ levels of recursion or equivalently, a product tree of *height* 11, if we want to recover in all cases all information that the continuation may provide. We limit us to a height of 3 so that there are 15

values $T_{0,0}, T_{1,0}, T_{1,1}, \ldots, T_{3,7}$ to be stored. Since we use that tree structure also as a basis for our primality test we compute $d_0 = \gcd(n, T_{0,0})$ and proceed to compute $d_1 = \gcd(d_0, T_{1,0})$ if $d_0 > 1$. Then we compute d_2/d_1 and depending on whether $d_1 > d_2$ or not we continue by computing $d_3 = \gcd(d_1, T_{2,0})$ or $d_3 = \gcd(d_2, T_{2,2})$. Likewise we continue to compute $d_4 = \gcd(d_3, T_{3,i})$ for an $i \in \{0, 2, 4, 6\}$. Consequently we end up with precisely 4 gcd computations after the continuation. See [11] for more details.

We choose $t = 30$ so that there are only 16 different values for r which occur in a representation of a prime q considered in the continuation. Therefore we only need to keep 16 points of the form $2r \cdot Q$ in memory while s runs from 1 to 162. Furthermore a pair (r, s) may represent two primes at once. Our choice of t reduces the number of 1116 primes to precisely 1024 pairs.

Primality Test. We know that smooth cofactors $n \geq z_{\mathbf{a}}$ are composite and that composite numbers $n < z_{\mathbf{a}}$ are of the form $n = p \cdot q$ with p, q prime, because we assume that the trial division removed all small prime factors less than 100,000. Assuming $p < q$ this leads to $p \leq 774{,}593$. Then for every elliptic curve E_p over $\mathbb{F}_p$ we have $\#E_p \leq 776{,}353$ and we may assume $16 \mid \#E_p$ such that the greatest prime dividing $\#E_p$ is bounded from above by $\tilde{C} := \tilde{v} := 48{,}522$. For the same reason every prime whose square divides $\#E_p$ is less than $\tilde{B} := 220$. We could use ECM with the parameters v, B, C replaced by $\tilde{v}, \tilde{B}, \tilde{C}$ to search for p and thereby checking primality. But the values of v, B, C we choose for factoring do not differ significantly from those given here. In fact, an analysis of the orders of the starting points on the curves of our family modulo all primes $p \leq 774{,}593$ showed that we may use the same ECM parameters for factoring and primality testing. This enables us to factor and test primality at once. Therefore we need no additional circuitry for a primality test.

This primality test fails if we encounter a composite $n = p \cdot q$ where p and q have, in view of ECM and our parameters, the same order characteristics for all 84 curves. That means that for each curve one of the following three cases applies: (a) the orders modulo p and q both divide k (first phase); (b) p and q are both detected by the same (r, s) pair (or the same class of (r, s) pairs since we grouped those into classes by restricting the product tree); (c) p and q are not detected at all. If p and q differ significantly it is unlikely that (a), (b) or (c) applies to them for each curve. The most critical situation arises when $100{,}000 \leq p, q \leq 774{,}593$, as this maximizes the probability of (a) or (b) being applicable. Those p and q lead to 1,377,153,921 squarefree composites $n = p \cdot q$. We analyzed all 84 curves for all those composites and searched for critical n which would fail the primality test. Table 1 shows the number and percentage of those composites n which will not be factored after the given number of curves. If we assume that every cofactor output by the sieving process leads to four composites of the form $p \cdot q$ as discussed above and if we assume that every such composite passes at least 12 curves then Table 1 shows that at most a fraction of $((1 + 7.3 \cdot 10^{-7})^8 - 1) < 6 \cdot 10^{-6}$ of the relations get lost because of semi-smooth cofactors which remain composite after the ECM processing. In reality this number should be even smaller.

Table 1. Average number and percentage of squarefree composites $n = p \cdot q$ missed

# curves	6	12	18	24	30	84
missed	$7.4 \cdot 10^5$	1,009	3.5	0.03	$< 1.7 \cdot 10^{-3}$	0
%	$5.4 \cdot 10^{-2}$	$7.3 \cdot 10^{-5}$	$2.5 \cdot 10^{-7}$	$2.2 \cdot 10^{-9}$	$< 1.2 \cdot 10^{-10}$	0

Performance of ECM. The performance of ECM as a primality test has already been discussed above. Software simulations give evidence that 84 curves suffice to factor most candidates possible. With a set of 84 curves we expect to be able to factor the norms successfully for at least 99.5% of all candidates meeting both the algebraic and the rational semi-smoothness criteria.

4 Hardware Estimates and Implementation Considerations

The objective of our architecture is an AT-efficient design of ECM that for parameters of interest can be implemented with existing technology. In this section we briefly describe the area and time complexity resulting from our choice of algorithms and derive estimates for the overall performance of our design when being applied to the above NFS parameters for RSA-1024.

4.1 Modular Arithmetic

The proposed design requires modular multiplication, squaring, addition, subtraction, and gcd computations. For performing the modular arithmetic operations, we choose Montgomery residues that enable an efficient method for modular multiplication [18]. Montgomery's algorithm replaces divisions by simple shift operations and, thus, is very efficient in hard- and software. The method is based on a representation of the residue class modulo an integer n. The corresponding n-residue of an integer x is defined as $x' = x \cdot r \bmod n$ where $r = 2^m$ with $2^{m-1} < n < 2^m$ such that $\gcd(r, n) = 1$. Since it is costly to switch always between integers and their n-residue and vice versa, we will perform all computations in the residue class.

Modular subtraction and addition can be computed with a single circuit using simple carry ripple adders (CRA). To guarantee a low latency, operations are done word-wise at an appropriate word size and corresponding number of words. An efficient architecture for modular multiplication is described in [24] and seems suitable for our design. Squaring will be realized with the multiplication circuit since a separate squaring circuit would unnecessarily increase the overall area-time (AT) product. A variant with carry ripple adders has been implemented and analyzed for the use with ECM in [20]. The architecture enables a word-wise multiplication and is scalable regarding operand size, word size, and pipelining depth.

For the required gcd computations and modular inversions, we adopt the (extended) binary euclidean algorithm [17] which can easily be implemented

with the presence of a subtraction hardware and some registers. As additional functionality, two registers must perform a simple shift operation (equivalent to a division by two). Since the (extended) gcd operations are always performed after the actual point operations, we can use internal registers for the computation and do not need additional hardware.

4.2 Factorization Unit

A detailed analysis of the modular arithmetic underlying a hardware implementation of ECM can be found in [20] and its modification for this contribution is summarized in Appendix A.

ECM Cluster. Excluding the time for pre- and post-processing, for the 1024-bit parameters discussed above a single candidate requires a total of 14 ms on the rational and 29 ms on the algebraic side if we assume a (realistic) clock frequency of 240 MHz. For the rational and algebraic ECM unit, the overall transistor count amounts approximately to 290,000 and 340,000, respectively. Assuming standard 0.13 μm CMOS technology, the silicon area required for an ECM cluster, i. e., 6 ECM units together with the reduction unit, is no more than 4.92 mm^2 (rational) and 5.76 mm^2 (algebraic). For details on the complexity estimate of the required area and time, we refer to Appendix A. Figure 1 shows the basic layout of a factorization unit consisting of the norm evaluation, the division pipeline, a central control unit with memory, and the ECM clusters.

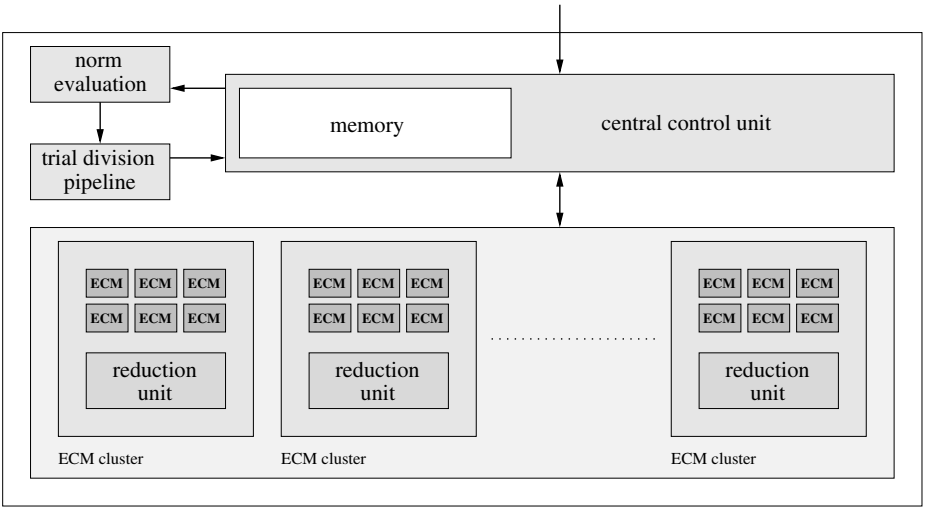


Fig. 1. Basic layout of a factorization unit

Division Pipeline. For the pre-processing of presumably semi-smooth numbers, we perform a trial division by 9592 primes and 108 prime powers up to 100,000 which is realized by a pipelined structure. Once the pipeline is filled it can handle all divisors at a rate of approximately 330 (rational) and 210 (algebraic) numbers per second at a clock rate of 240 MHz. This is sufficient since trial division is done on each chip. The estimated area consumption of the architecture is approx. 60,000 and 75,000 equivalent logic transistors (0.17 mm^2 and 0.21 mm^2) on the rational and algebraic side, respectively.

Central Control Unit. Per chip, we assume a central control unit taking care of all incoming pairs (a, b), the respective factors found during trial division, the computation of the curve parameters, and the corresponding results from the ECM stages. The central control unit can be realized with a standard CPU core attached to some memory for keeping track of the numbers coming from the trial division pipeline and their factorization. We estimate such a unit to consume no more than 1 mm^2 of silicon area.

For a moderate chip size of 147 mm^2, the size of a Pentium 4 processor, we can group 174 rational or 150 algebraic ECM units (29 rational or 25 algebraic clusters) on a single chip.

4.3 Application to TWIRL

For the discussed NFS parameters, when combining our design with TWIRL we estimate that it suffices to handle about 1000 sieve locations per second (cf. Section 2.1). Basing on software simulations, we assume that on the rational side on average 61 curves are used for the factorization of one norm. This requires a computing time of 0.8 seconds per norm. On the algebraic side on average 70 curves are used, this results in a factorization time of 2.0 seconds per algebraic norm. With 5 chips of the size of 147 mm^2 (each including 29 ECM clusters) up to 1080 rational norms can be factored per second. With 6 chips of the same size (each including 25 ECM clusters) up to 440 resulting candidates can be checked on the algebraic side. Thus, for the discussed parameters, about 11 chips of standard size should suffice to substitute the diary logic from TWIRL and to compute almost all of the occurring cofactor factorizations.

5 Conclusion

The above discussion shows that the integration of ECM with a diary-free TWIRL results in a sieving design with a significantly simpler layout, but basically the same performance as the original TWIRL. For parameters as currently expected for an NFS-based factorization of RSA-1024, the additional circuitry can be expected to fit on about 11 chips of standard size and moderate complexity. Due to its moderate technological requirements, our design might also be of interest for being used in connection with "classical" sieving implementations.

Acknowledgments

We thank Adi Shamir and Eran Tromer for valuable discussions on TWIRL and Thorsten Kleinjung for helpful discussions on the NFS.

References

1. A. Oliver L. Atkin and François Morain. Finding suitable curves for the elliptic curve method of factorization. *Mathematics of Computation*, 60(201):399–405, 1993.
2. Daniel J. Bernstein. Circuits for Integer Factorization: a Proposal. At the time of writing available electronically at `http://cr.yp.to/papers/nfscircuit.pdf`, 2001.
3. Richard P. Brent. Factorization of the tenth and eleventh Fermat Numbers. *Computer Science Laboratory, Australian National Univ., Canberra*, Report TR-CS-96-02:1–25, 1996.
4. Jens Franke, Thorsten Kleinjung, Christof Paar, Jan Pelzl, Christine Priplata, and Colin Stahlke. SHARK: A Realizable Special Hardware Sieving Device for Factoring 1024-Bit Integers. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems; CHES 2005 Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 119–130. Springer, 2005.
5. Willi Geiselmann, Hubert Köpfer, Rainer Steinwandt, and Eran Tromer. Improved Routing-Based Linear Algebra for the Number Field Sieve. In *Proceedings of ITCC '05 – Track on Embedded Cryptographic Systems*. IEEE Computer Society, 2005.
6. Willi Geiselmann, Adi Shamir, Rainer Steinwandt, and Eran Tromer. Scalable Hardware for Sparse Systems of Linear Equations, with Applications to Integer Factorization. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems; CHES 2005 Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 131–146. Springer, 2005.
7. Willi Geiselmann and Rainer Steinwandt. A Dedicated Sieving Hardware. In Yvo G. Desmedt, editor, *Public Key Cryptography — PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 254–266. Springer, 2003.
8. Willi Geiselmann and Rainer Steinwandt. Hardware for Solving Sparse Systems of Linear Equations over GF(2). In Colin D. Walter, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems; CHES 2003 Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 51–61. Springer, 2003.
9. Willi Geiselmann and Rainer Steinwandt. Yet Another Sieving Device. In Tatsuaki Okamoto, editor, *Topics in Cryptology — CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 2004.
10. Tetsuya Izu, Noboru Kunihiro, Kazuo Ohta, and Takeshi Shimoyama. Analysis on the Clockwise Transposition Routing for Dedicated Factoring Devices. In Jooseok Song, Taekyoung Kwon, and Moti Yung, editors, *Information Security Applications: 6th International Workshop, WISA 2005*, volume 3786 of *Lecture Notes in Computer Science*, pages 232–242. Springer, 2006.
11. Fabian Januszewski. Ein dedizierter Faktorisierungsalgorithmus auf Basis elliptischer Kurven. Diplomarbeit, Universität Karlsruhe (Germany), Fakultät für Informatik, Institut für Algorithmen und Kognitive Systeme, 2005.
12. RSA Laboratories. The RSA Challenge Numbers. http://www.rsasecurity.com/rsalabs/node.asp?id=2093.

13. Arjen K. Lenstra and Jr. Hendrik W. Lenstra, editors. *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer, 1993.
14. Arjen K. Lenstra, Adi Shamir, Jim Tomlinson, and Eran Tromer. Analysis of Bernstein's Factorization Circuit. In Yuliang Zheng, editor, *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2002.
15. Arjen K. Lenstra, Eran Tromer, Adi Shamir, Wil Kortsmit, Bruce Dodson, James Hughes, and Paul C. Leyland. Factoring Estimates for a 1024-Bit RSA Modulus. In Chi-Sung Laih, editor, *Advances in Cryptology — ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 55–74. Springer, 2003.
16. Hendrik W. Lenstra. Factoring Integers with Elliptic Curves. *Annals of Mathematics*, 126(2):649–673, 1987.
17. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, Florida, USA, 1997.
18. Peter L. Montgomery. Modular Multiplication without Trial Division. *Mathematics of Computation*, 44(170):519–521, April 1985.
19. Peter L. Montgomery. Speeding up the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48:243–264, 1987.
20. Jan Pelzl, Martin Šimka, Thorsten Kleinjung, Jens Franke, Christine Priplata, Colin Stahlke, Miloš Drutarovský, Viktor Fischer, and Christof Paar. Area-Time Efficient Hardware Architecture for Factoring Integers with the Elliptic Curve Method. *IEE Proceedings Information Security*, 152(1):67–78, October 2005.
21. John M. Pollard. A Monte Carlo Method for Factorization. *Nordisk Tidskrift for Informationsbehandlung (BIT)*, 15:331–334, 1975.
22. Carl Pomerance. A Tale of Two Sieves. *Notices of the ACM*, pages 1473–1485, December 1996.
23. Adi Shamir and Eran Tromer. Factoring Large Numbers with the TWIRL Device. In Dan Boneh, editor, *Advances in Cryptology — CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2003.
24. Alexandre F. Tenca and Çetin K. Koç. A Scalable Architecture for Modular Multiplication Based on Montgomery's Algorithm. *IEEE Trans. Comput.*, 52(9):1215–1221, 2003.

A Detailed Cost Estimates

To derive detailed cost estimates for our design, if possible, we provide the exact area and time complexity for the chosen algorithms. However, some algorithms have a non-deterministic runtime and upper bounds on the time complexity are assumed. Hence, the overall estimate should be seen as an upper bound on the time and area complexity. In the sequel, let $m = \lceil \log_2 n \rceil$ be the bitlength of the modulus and let e be the number of words of size w, required to represent a number of size of m bit.

A.1 Complexity of the Basic Building Blocks

For each operation, the area and time complexity is given with respect to the parameters m, e, and p. We will include $T_{init} = 2$ cycles for initialization of the ALU at the beginning of the actual computation.

Memory and Registers: We will take the quite realistic area estimates for an 0.13 μm CMOS process from [23]: For logic, the area per transistor is assumed to be 2.8 μm^2. Due to the highly regular structure, DRAM requires approximately 0.2 μm^2 per bit, equivalent to 0.07 of the area per transistor for logic[3]. For our implementation, we also require SRAM cells which can be clocked with higher frequencies than DRAM and do not need any refresh cycles. SRAM requires less clock cycles for reading and writing but has the disadvantage of an increased area demand for which we will assume 1.5 gate equivalent (NAND) or 6 transistors.

We assume an Arithmetic Logic Unit (ALU) with a certain amount of internal registers which are used frequently as input and output to the arithmetic functions. For our estimate, we pick SRAM at the cost of 6 transistors per bit for the implementation of such. For tables and larger memory blocks for which we can tolerate a higher latency we choose simple DRAM. For the relatively large memory blocks, we take care of the additional area for address, row, and column decoding by introducing some overhead.

Modular Addition and Subtraction: With the algorithms and architectures specified in [20], subtraction requires in the worst case $2 \cdot (e + 1)$ clock cycles, addition $3 \cdot (e + 1)$ clock cycles, where $e = \lceil \frac{m}{w} \rceil$ is the number of words. Hence, the maximum time for an addition or subtraction is bounded by

$$t_{add/sub} = 3 \cdot (e + 1) + T_{init}$$

clock cycles. A single full adder (FA) can be built of 3 NAND and 2 XOR gates, summing up to 24 transistors in standard CMOS technology. With w being the the required number of FAs, we can construct the CRA for approximately

$$A_{add/sub} = 24 \cdot w + 500$$

transistors with an assumed overhead of 500 transistors for the internal control logic.

Modular Multiplication: The number of clock cycles per multiplication is given by

$$t_{mul/squ} = \left\lceil \frac{m}{p} \right\rceil \cdot (e + 1) + 2p + T_{init},$$

where p is the pipelining depth of the design [24]. For standard CMOS technology, the area consumption of the multiplier is

$$A_{mul/squ} = 4 \cdot (84wp + 25p - 22w)$$

transistors. Note that not all values for p are reasonable (see [24] for possible kernel configurations). A word-width of $w = 64$ bit and a pipelining-depth of $p = 4$ processing elements seems to be a good trade-off for speed and area consumption and is chosen in our context.

[3] This corresponds to the average area usage of a transistor of a Pentium 4 processor with $55 \cdot 10^6$ transistors on a silicon area of 147 mm^2.

(Extended) GCD Computation: The binary gcd architecture for bitlength m requires in the worst case $2m$ subtractions and some cycles for multiplexing (shifting can be hard-wired). For the sake of simplicity, we leave out a detailed analysis of the average runtime of the binary gcd and assume as upper bound

$$t_{gcd} = 2m \cdot ((e+1) + T_{init})$$

clock cycles to finish. Prior to each ECM iteration, the precomputation of the required curve parameters needs two modular inversions, which can be computed with the binary extended gcd. The runtime for the algorithm in hardware is at most

$$t_{inv} = (4m+4) \cdot ((e+1) + T_{init})$$

clock cycles [17]. We require no additional register since at the time of the extended gcd computation, all registers for ECM phase 2 are available.

Since we use the ALU for performing the gcd, no additional subtraction circuit or additional registers are required. We assume 2000 transistors for the additional control logic.

Trial Division Pipeline: The trial division by 9592 primes and 108 prime powers up to 100,000 can be realized by a pipelined structure of 10 division circuits. For each division circuit we need two m-bit registers (for the input and the result), two 17-bit registers (for the intermediate result and for the operand), and a 17-bit adder. For the actual division, a simple shift and subtract method is applied, requiring at most m subtractions of 17-bit precision. Since subtractions are performed by additions of the two's complement (plus 1), we can directly use additions by storing the two's complements of the prime powers. The division of a candidate of m bit by a prime power of 17 bit requires no more than $t_{division} = 3 \cdot m + 50$ clock cycles, where the 50 clock cycles are an upper bound on administrative cycles required to initialize the circuit prior computation and to send the result to the central control unit.

Since the division circuit is "too fast" for our purpose, we suggest to use a single circuit for 1000 primes and prime powers by adding required control logic and additional memory to the circuit. Hence, a pipeline of 10 units can handle all divisors at a rate of approximately 330 (rational) and 210 (algebraic) numbers per second at a clock rate of 240 MHz.

The estimated area consumption of the pipeline, including internal control logic, registers, and memory amounts to approx. 60,000 (rational) and 75,000 (algebraic) equivalent logic transistors. Note that the absolute latency of up to 29 ms of the division pipeline is not relevant once the pipeline is filled and a constant throughput is reached.

Reducing the Curve Parameters: The reduction of precomputed parameters for the initialization of ECM requires additional circuitry. We propose to group the ECM units in clusters of six, with a single reduction circuit per cluster. The reduction circuit prepares the next parameters for ECM and operates concurrently to the ECM units. It stores the six (fixed) input values, and two

intermediate values of size of 17 kbit in DRAM. With a pipelined DRAM to SRAM circuit for prefetching memory blocks and with a simple shift and add circuit based on word-wise addition (word size w), we assume the total area of such a reduction unit to be no more than $A_{reduction} = 16{,}100$ transistors, including memory, required logic for the computation and for the distribution of the results. The required time per reduction is bounded by $t_{reduction} = 400,000$ cycles for the largest input. The six inputs are of different size, and we certainly can reduce six parameters during the computational phase of ECM.

A.2 Area and Time Complexity of a Single ECM Unit

The precomputation of the curve parameters $\tilde{x}, \tilde{z}$ and $(t+2)/4$ can be done with the functionality discussed above, together with some of the registers available from phase 2. Prioritizing the use of additions and, conservatively, estimating the time for a multiplication by 2 or 4 (shifting) as $t_{add/sub}$, the computation of $\tilde{x}, \tilde{z}$ and $(t+2)/4$ amounts to

$$t_{precomp} = 6 \cdot t_{mul/squ} + 14 \cdot t_{add/sub} + 2 \cdot t_{inv}$$

clock cycles.

Following [20], phase 1 requires a total of 12 registers. With $t = 30$, the precomputation of the table in phase 2 requires a total of $32 = 2 \cdot 16$ registers of length of m bit for $2r \cdot Q$ with $r \in \{1, 4, 6, 7, 9, 10, 12, 15, 16, 19, 21, 22, 24, 25, 27, 30\}$. We need additional 15 registers for building the tree according to Section 3.2, amounting to a total of 59 registers of SRAM, i. e.,

$$A_{SRAM} = 59 \cdot 6 \cdot m$$

equivalent logic transistors. The precomputation time for the table amounts to 8 point duplications and 10 point additions:

$$t_{table} = 8 \cdot t_{point\_dup} + 10 \cdot t_{point\_add} = 100 \cdot (t_{mul/squ} + t_{add/sub})$$

We can determine the runtime and area consumption of both phases on basis of the underlying ring arithmetic and the corresponding upper bound of the runtimes. A setting with $m_r = 216, m_a = 350, w = 64, p = 4, e_r = 4$, and $e_a = 6$ yields $t_{add,r} = 3(e+1) + T_{init} = 17$ $(e = 4)$, $t_{add,a} = 3(e+1) + T_{init} = 23$ $(e = 6)$ and $t_{sub,r} = 12$, and $t_{sub,a} = 16$ clock cycles. For this configuration, a modular multiplication of full length takes $t_{mul,r} = 280$ and $t_{mul,a} = 626$ clock cycles for the rational bit length $m_r = 216$ and for the algebraic bit length $m_a = 350$, respectively.

For a single gcd we require at most $t_{gcd,r} = 2160, t_{gcd,a} = 4900$ cycles. Hence, the total cycle count for both phases and for a single curve is no more than

$$t_{ECM} = 6500 \cdot t_{add/sub} + 10700 \cdot t_{mul/squ} + 5 \cdot t_{gcd} + t_{precomp}$$

clock cycles including the precomputation time for the table (cf. [11]). Excluding the time for pre- and post-processing, a single candidate on the rational and

algebraic side requires a total of $t_{ECM,r} = 3.2 \cdot 10^6$ and $t_{ECM,a} = 6.9 \cdot 10^6$ clock cycles, respectively. If we assume a frequency of 240 MHz, checking a candidate with one curve requires approximately 14 ms on the rational side and 29 ms on the algebraic side. For reading values from the DRAM in phase 2, we assume an efficient SRAM-based prefetch circuit.

For implementing a single ECM unit capable of testing a single curve at a time, we need

$$A_{ECM} = A_{mul/squ} + A_{add/sub} + A_{gcd} + A_{SRAM} + A_{DRAM} + A_k + A_{control}$$

equivalent logic transistors, where $A_{control}$ is the additional logic required to control both phases of ECM, the internal gcd, and fast DRAM to SRAM logic. We assume the control logic to consume no more than $A_{control} = 100{,}000$ equivalent logic transistors which is a fairly conservative estimate[4]. The ECM unit includes a barrel shifter composed of D-flip-flops (48 transistors per bit) for the 589-bit scalar k required for the point multiplication $k \cdot P$ in phase 1. Furthermore, a DRAM memory block stores the (s', r') pairs for all primes between the last prime of the scalar k and $C = 9680$ for phase 2. The pairs (r', s') are sequentially read from memory and are used to control the second phase of ECM. Note that we can compress the information of (r, s) to smaller values (r', s'): The actual value of s is stored in a small 7-bit counter and is simply increased when the 1-bit value s' is equal to '1', leading to a point addition of $2st \cdot Q$ in phase 2. Hence, only a single bit of memory is required for s'. The value of r' points to the corresponding table entry for $2r \cdot Q$. With $t = 30$, we require only 4 bit for r'. A total of 5120 bit DRAM for a total of 1024 pairs (resulting from 1116 primes) is required. For the rational and algebraic ECM unit, the overall transistor count amounts approximately to 290,000 and 340,000, respectively. Assuming a standard 0.13 μm CMOS process, a single ECM processor requires no more than 0.81 mm^2 (rational) and 0.95 mm^2 (algebraic) area of silicon.

[4] For comparison: A simple microcontroller such as an 8051 derivate can be realized with 40,000 transistors.

Janus: A Two-Sided Analytical Model for Multi-Stage Coordinated Attacks

Zonghua Zhang[1], Pin-Han Ho[1], Xiaodong Lin[1], and Hong Shen[2]

[1] Department of Electrical and Computer Engineering
University of Waterloo, Ontario, Canada
[2] Department of Computer and Mathematics,
Manchester Metropolitan University
All Saints, Manchester, England, M15 6BH
z14zhang@uwaterloo.ca, {pinhan, xdlin}@bbcr.uwaterloo.ca,
H.Shen@mmu.ac.uk

Abstract. The multi-stage coordinated attack (MSCA) bring many challenges to the security analysts due to their special temporal an spacial characteristics. This paper presents a two-sided model, Janus, to characterize and analyze the the behavior of attacker and defender in MSCA. Their behavior is firstly formulated as Multi-agent Partially Observable Markov Decision Process (MPO-MDP), an ANTS algorithm is then developed from the perspective of attacker to approximately search attack schemes with the minimum cost, and another backward searching algorithm APD-BS is designed from the defender's standpoint to seek the pivots of attack schemes in order to effectively countermine them by removing those key observations associated with the system state estimates. Two case studies are conducted to show the application of our models and algorithms to practical scenarios, some preliminary analysis are also given to validate their performance and advantages.

1 Introduction

Among the diverse attack variants, the most destructive and difficult one to detect are those that occur in stages over time and cooperated by a group of attackers/attack parties, which is called multi-stage coordinated attacks (MSCA). To accomplish such a compromise, attacker needs to undergo a process of reconnaissance, penetration, attack, and exploit. Meanwhile, a group of attackers need to plan and cooperate each other for their common goals by resource/tools sharing, task allocation, information communication and synchronization. Due to the special characteristic of MSCA, the simple combination and correlation of some individual countermeasures can hardly provide effective safeguard for the computer networks with distributed potential vulnerabilities. An ideal approach not only considers the system state transitions (for multi-stage) but also handle interactions between attacker's joint actions (for coordinated).

Rather than focused on the development of specific countermeasures, this paper sheds light on the modeling and analysis of MCSA from a high-level

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 136–154, 2006.
© Springer-Verlag Berlin Heidelberg 2006

viewpoint. In general, we envision a framework in which the security-related information (key observations) of the dominant hosts in the computer networks, can be utilized to characterize their trust relationships and causal vulnerabilities. We also envision that attacker capabilities and attack scenarios can be constructed and represented in terms of particular observed preconditions, so that collusive behaviors of multiple attackers can be isolated and multi-stage attacks can be mitigated in case of further compromise. The hope is that those two concerns, just like two sides of one coin, from the standpoints of defender and attacker respectively, can be combined together to achieve a complementary perspective for the derivation of more generic models of attacker and defender, as well as the development of effective methods and techniques for the detection MSCA. We propose an analytical model named Janus[1](Joint ANlytical model with two Unified Shields) in this paper, which essentially formulates the behavior of both attacker and defender as Partially Observable Markov Decision Process (POMDP) by taking into account their specific concerns. Furthermore, two algorithms with apparently opposite standpoints are derived from the analytical models, which aim to seek the minimum cost of attack for attackers and discover the key observations of attacks for defenders respectively. To the best of our knowledge, this is the first work giving a formal formulation and characterization on MSCA, although some specific attack variants have been modelled and analyzed.

The remainder of this paper is organized as follows. Section 2 gives a general attack analysis. In Section 3, a structural framework is formulated to represent and characterize MSCA's behavior, some basic properties are also derived. Section 4 conducts a specific two-sided analysis based on the model. To show the application of our proposed model, in Section 5, we illustrate two practical scenarios as case studies. Section 6 concludes the paper and points out the future work.

2 MSCA Analysis

Unlike those traditional attacks, which are usually launched by a single attacker to a single victim in a short period, MSCA are always conducted by a group of organized attackers sharing the same objective and attacking tools, and usually accomplish the intended goal in a long period via a number of collusive operations. Two key properties, namely, *multi-stage* and *coordinated*, cause MSCA different much from those traditional attacks and therefore hard to be detected by the typical models. Specifically, those two properties can be further depicted as follows,

Property 1. [**Multi-Stage**] or stealthy, which means that each attack is consisted of a sequence of distinct steps or subgoals, while each step represents an

[1] The god of gates and doorways in Roman Mythology, depicted with two faces looking in opposite directions.

atomic activity conducted by attackers or an achievement of a particular subgoal, no matter malicious or apparently legitimate

A typical example with this property is *remote Buffer Overflow* attack, which firstly involves a surveillance step to search exploitable host in the network, followed by an intrusion step using a known vulnerability, then followed by a privilege escalation step to improve the status of attacker for accessing the target, and some malicious goals, such as data theft or denial of some system service, are finally achieved. Another specific example of multi-stage attack is *vulnerability-finding-worms*, whose life-cycle generally includes four steps [9,20]: propagation, activation, infection and replication. Such attacks are sequential chains composed by several atomic operations, while any failure of any step would cause the attack fail to succeed. The sequential and probabilistic nature of such attack allow them to be formulated as a Hidden Markov Model (HMM), in which the current system state always depends on the previous one while cannot be observed by the operators, and what the operators are able to observe is those observations (user privilege, file status, etc.) emitted by the underlying system state. However, in practice, a challenging issue is that the transition between system states occasionally is separated by a significant temporal interval, and although a successful multiple phase attack requires the consequential operations in sequence, the atomic operations within a particular phase might be interchangeable.

Property 2. [**Coordinated**] or collusive, a group of attackers (or a collection attack parties employed by a single attacker[2]) cooperate each other and simultaneously compromise a target host or network by joint rather than individual actions.

A very simple form of coordinated attack is distributed attacks like DDoS [12] in which perpetrators remotely control a multitude of compromised systems (or zombie computers) to attack a single target and direct the further attack, thereby causing denial of service for users of the whole network. The key characteristic of such attacks is that the compromising point is multiple, the correlated operations and aggregated effects rather than the individual compromise result in the successful attack. The attacker involved in the attack does not necessarily mean human, it also might be artificial agents/tools, malicious scripts/codes acting on behalf of human, even activated by a single attacker. More, although there is no compelling need to discriminate the MSCA and coordinated attacks here, it is worth nothing that the coordinated attacks usually takes several stages, but the correlation among the independent attackers sometimes are not so closely, and the sequential property of the actions are not very obvious and even some of stages may not appear or discernable, while our concern in this paper are those coordinated attacks with observable stages.

So far, there are many effective measures have been developed to countermine multi-stage attacks (stealthy) and coordinated attacks respectively, however, the integration of those two attack forms bring the defender more challenges: (a)

[2] We use attacker and attack party interchangeablely in the rest of this paper.

the coordinated attack is able to escape from intrusion detection (no matter signature-based or anomaly-based) by breaking the predefined attack patterns into many apparently innocent pieces (stealthy); (b) some attacker may distract the IDS by intentionally triggering its alerts and consuming its resources, in this case, although some of the minor or decoy attacks are detected, the major or true attack goal can still succeed; (c) attackers can take various forms to achieve the same goal, for instance, at a particular stage, once one atomic attack has been detected, it can still take another action to bypass the detection, or launch some similar attacks simultaneously with the main attack.

More formally, we assume a network consisting a set of potential victims $V = \{v_1, v_2..., v_m\}$, where v_i denotes a host in the network with some particular vulnerabilities $\{v_i^1...v_i^{k_i}\}$ that can be exploited by attacker, and a group of attackers $E = \{e_1, e_2, ...e_n\}$ attempt to crack V during multiple stages T, where $T = \{1, 2, ...l\}$ is a discrete variable. We also suppose that the target system undergoes a sequence of states $S = \{s_1, s_2, ...s_n\}$ under attack during some particular stages, while each attacker e_i has his own action space u^i. Thus, the multi-stage coordinated attack scenario can be generalized with following properties: (1)$Pr\{\hat{u}_i|s_i, v_i\}$ is assumed as the probability of a group of attackers E taking joint actions $\hat{u}_i = u^1 \times u^2 \times \cdots \times u^n$ at stage t, with the knowledge of system state s_i and victim v_i. The elements has following specific meaning; (2) it is the joint action $\hat{u}_i$ of E rather than the individual action u^i of attacker e_i moves the system state from s_i to s_j; (3) the joint action $\hat{u}_i$ depends on the preconditions of V in state s_i, while the concurrent action list changes with the specific sate; (4) from the viewpoint of attackers, they always intend to take the minimum set of actions (n of $\hat{u}$ is as small as possible) to achieve their goal (or subgoals) for the sake of saving cost (Section 4.1) and decreasing the probability of being detected.

3 A Formal Structural Framework

Based on the understanding of MSCA, this section aims to further characterize the attacker's behavior by formulating it as a multi-agent partially observable Markov decision process (MPO-MDP), and depict the general behavior of MSCA via some quantitative analysis, several specific properties are also derived from the model.

3.1 Model Formulation

Although a MSCA case involves a group of heterogenous attackers sharing the common goal and assistant tools, before achieving the final goal, it is still reasonable to assume that each attacker is independently acting in its own environment with uncertain perceptions and actions according to the particular observations. From the perspective of rational attackers, suppose his/her goal can be ultimately achieved as the system state evolves over a long-run under attacks, an optimal attack strategy with the minimum cost thereby is the mostly desirable.

However, due to the inherent uncertainty of the system with respect to the unknown vulnerabilities and the effects of defense mechanism, neither the evolution of the system state nor the generation of observations is a well-posed problem to be optimized directly, a probabilistic rather than a deterministic model therefore is more suitable to characterize attacker behavior. It is also worth noting here that the main concern of the model is to characterize the MSCA as a whole rather than to specify the behavior of individual attacker explicitly. The individual attacker takes action according to the estimated true system state, which is indirectly observed in its own environment, therefore, the general attack is partially observable for each individual attacker. More, since the next system state is dependent only upon the current state and the previous overall action of E, the decision of taking action for MSCA is a Markov process. So POMDP is formulated here.

Formally, a POMDP model is structurally characterized by four key elements [1]: a finite state space S, an action space U, an observation space Z, and a (possibly stochastic) reward $r(i) \in \mathbb{R}$ for each state $s_i \in S$, or in another sense, cost $c_{i,j}(u)$ for state transition from s_i to s_j with a particular action u. As a POMDP model, the interaction between an individual attack party e_x and its operating environment includes a sequence of decision stages as follows: (1) at state i, the system is in a particular state $s_i \in S$, and the underlying state emits an observation $z_i \in Z$ according to a probability distribution $\nu(s_i)$ over observation vectors; (2) the attacker takes action $u_i^x \in U$ in accordance with a randomized policy based on a probability distribution $\mu(z_i)$ over actions, with known z_i; (3) u_i^x determines a stochastic matrix $Pr(u_i^x) = [p_{ij}(u_i^x)]$, where $p_{ij}(u_i^x)$ is the probability of making a transition from state s_i to state s_j under action u_i^x; (4) in each attacking stage, e_x receives a reward signal r_i, while the aim of the attacker is to choose a policy so as to maximize the long-term average of reward, or to minimize the aggregate costs $c_{i,j}(\hat{u})$ of transferring system states. So the Markov chains for state transitions s_i and s_j are generated as a Markov chain: $s_i \in S[\nu(s_i)] \rightarrow z_i \in Z[\mu(z_i)] \rightarrow u_i \in U[p_{ij}] \rightarrow s_j \in S$.

As the behavior of each attacker can be formulated as a POMDP, the entire attack scheme naturally can be modelled as a multi-agent POMDP, or MPOMDP. Formally, the meta-action of the attacker U contains the cross product of all the actions available to each attacker, that is, $U = \{\hat{u}_i | \hat{u}_i = u_i^1 \times u_i^2 \times \cdots \times u_i^m\}$. At each stage, each individual attacker chooses his/her action according to the observation vector, and the atomic actions are then combined to form the meta-action. For stochastic policies, the overall action distribution is the joint distribution of actions for each attacker, $\mu(u_1, u_2, ...u_n | z_1, z_2, ...z_n)$. Furthermore, some practical constraints have to be considered due to MSCA's specific characteristic. Since the system states cannot be observed directly, they can only be represented as a set (or conjunction) of those observations z_i that are true in the state. For example, an attacker e_1 successfully cracked v_1 as a super-user and another attacker e_3 accessed the log files in v_2, the observation z_i thus can be represented as $e_1(v_1) \wedge e_3(v_2)$.

Based on the formulation and the specific concerns, the above parameters involved in the attack scheme can be organized into a family of action-dependent matrices: $m{\cdot}n{\times}n$ state transition probability $Pr\{s_j|s_i,\hat{u}_i\}$ of matrices F, $m{\cdot}n{\times}q$ observation probability $Pr\{z_i|s_i,\hat{u}_{i-1}\}$ (or $\nu(s_i)$) of matrices H, $m\cdot n\times n$ transition reward matrices Y, and $q\cdot m\times m$ action probability $Pr\{\hat{u}_i|z_i,\hat{u}_{i-1}\}$ of matrices Q (or $\mu(z_i)$, which essentially equals to $Pr\{\hat{u}_i|s_i,z_i\}$). All the matrices can be populated empirically by attacker with the prior knowledge, and the more sophisticated the attacker the more accuracy of the value, and from the attacker's viewpoint, one critical point is the derivation of reward signal, which should be maximized provided subgoals are achieved at each stage, i.e., $max\{\mathbb{E}[\frac{1}{M}\sum_{t=1}^{M} r_t]\}$(where M is the total number of stages after the final goal being achieved), and r_t can be specified to quantify the attacker's intents to the achievement of the goal. A more practical evaluation metric is the attack cost $c_{i,j}(\hat{u}_i)$, which is always expected to be minimized when enabling the transition of state s_i to s_j, and the objective function is $min\{\mathbb{E}[\frac{1}{|N_A|}\sum_{i=0}^{|N_A|} c_{i,j}(\hat{u}_i)]\}$ where $|N_A|$ is the total number of system states under MSCA, and $c_{i,j}(\hat{u}_i)$ can be defined with the significance of the subgoals, as well as the anticipated system state transitions.

3.2 Basic Properties

More specifically, a particular MSCA scenario can be specified as several concerns: *which* attacker e_i is performing the action u^i, *what* are the preconditions $\hat{v}_i$ of this action, *whether* other attackers are involved at the current stage, if so, *who*, and *what* concurrent actions have to be taken, *what* is the system state (estimated by observations) after the action.

Property 3 [Preconditions]. A successful MSCA depends on the status of victim V that can be taken advantage by attackers E. At the initial state, V mainly denotes the set of exploitable nodes in the system, while at other states, it also includes the conditions/effects that have been generated by previous actions.

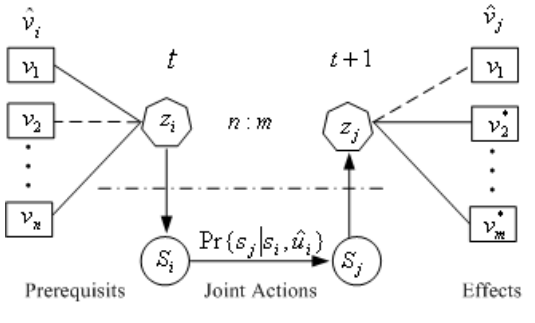


Fig. 1. MSCA State Transitions

The property shows that vulnerability checking and analysis is always the initial and essential step of MSCA. So far many techniques and tools have been developed to analyze vulnerability by checking system logs and monitoring specific monitoring performance metrics, most of which are based on the modeling of network specifications, such as fault tress, graph models, and performance models. Attack trees [16] and graph-based network vulnerability analysis [14,18] are two typical methodologies. For example, attacker trees are usually constructed in a given specific environment, and then quantify vulnerability by mapping known attack scenarios into trees; while graph-based approach analyzes risks to specific network assets and examines the possible consequences of a successful attack. The analysis system usually requires a database of common attacks (which might be broken into atomic steps), the information related with network configuration and topology, and even attacker's profile as the preconditions, the nodes of the graph represents an attack stage, and the graph thus enables us to identify the attack paths with the highest probability of success attack.

Property 4 [Concurrent Actions]. A group of attackers' joint action $\hat{u}_i$ keeps logically consistent in each system state s_i, which means that action u_i^j of attacker e_j does not necessarily contradicts with action u_i^k of attacker e_k, and the consistency of independent actions should be dependent on that of preconditions.

Naturally, the concurrent actions of a group of attackers is closely related to the attack efficiency, provided the malicious goal will ultimately be achieved. For instance, for a group of sophisticated attackers, they may cooperate well and no member acts any negative effect on the evolved system state, the joint action at each stage thereby successfully achieves the subgoals (at least generate the necessary preconditions for latter stages). While for a group of amateur attackers, the same goal might cost more even the same preconditions and tools are available.

Property 5 [Action Effects]. The effect of the joint action $\hat{u}_i$ is a list of observations, which are jointly consistent, without any conflicts, in the system state s_i. All or part of the concurrent observations can be taken as the preconditions by join action $\hat{u}_i$ for achieving the next system state s_j.

Figure 1 illustrates the relationships between the those introduced elements. Note that the dash line connecting vulnerable node v_2 and observation set z_i means v_2 does exist, but it is not taken as z_i; the dash line connecting v_1 and observation set z_j means v_1 is an existing vulnerability, but it is not generated by the join action $\hat{u}_i$; The dash circle of s_i and s_j means the system states that cannot be observed directly. The effect of join action $\hat{u}_i$ (only those v generated by the previous join action) and the existing v combine as the preconditions $\hat{v}_j$ for the next action $\hat{u}_j$. Those three properties depict a complete MSCA scenario to capture those key elements of attacker behavior. Although the major step of a MSCA is the concurrent action, the goal can hardly be achieved without insightful understanding of its preconditions and action effect, especially when the goal is expected to be obtained with a desirable cost.

4 Janus: A Two-Sided Analysis

Although the model is formulated mainly from the perspective of attacker, it also takes into account those concerns of security analyst, which thereby can serve both as a semantic model for the characterization of normal behavior and defender. Two heuristic algorithms are then developed to infer the model.

4.1 Attacker-Centric Analysis

In the model, the system state S contains both normal ones S_N and abnormal ones S_A under MSCA, namely $S = S_N \cup S_A$, which are not accessible for the observers while can be distinguishable via particular observations Z by their knowledge and skills. However, for a group of attackers, there is no such a compelling need to differentiate all the possible state transitions, rather, they only need to discern those attack-relevant states by some distinctive features of the states. Therefore, the states in the attacker's analytical model derived from the general one only means those attack-relevant states, i.e., the system states under attacks $S_A = \{s_0, s_1, \cdots s_a\}$, i.e., $S_A \subseteq S$.

As the model shows, reward signal or attacking cost can be used to evaluate attacks' efficiency. Assume a group of attackers E successfully attack a target by T stages, and the final state is s_a, the most desirable reward signal should be $max\{\mathbb{E}[\frac{1}{T}\sum_{i=1}^{T} r_i]\}$. Suppose the initial system state is s_0, and the system states are transited in a sequential order, i.e., $s_0, s_1, \cdots, s_{a-1}, s_a$, the cost of transiting system states can be computed as,

$$C_{0,a}(E) = c_{0,1}(\hat{u}_0) + c_{1,2}(\hat{u}_1) + \cdots + c_{a-1,a}(\hat{u}_{a-1}) = \sum_{i=0}^{a-1} c_{i,i+1}(\hat{u}_i)$$

More generally, for $s_i, s_j \in S, i \neq j \in [0, a]$, $C_{0,a}(E) = \sum_{i,j} c_{i,j}(\hat{u}_i)$. Obviously, for coordinated attackers E, the smaller $C_{0,a}(E)$ the better. In most cases, a smaller $C_{0,a}(E)$ means a smaller set of concurrent actions $|U|$, which are closely related to the efficiency of MSCA based on the following observations: (a) a smaller $|U|$ might require a smaller $|V|$, which means that a smaller concurrent action sets usually needs less prerequisites for attack and, (b) a smaller $|U|$ usually requires a smaller $|E|$, which means that a smaller number of attackers are involved in the attack procedure and, (c) small $|U|$, $|V|$ and $|E|$ reduce the complexity of attacks, and may have less probability of being detected. Here the definition of $C_{o,a}(E)$ needs more concern, which essentially guides the evolution of the model. Generally, attack cost can be measured by time, computational costs&resources (hardware, software, internet connectivity, user privilege), and some assistant tools, etc., and it is also relevant with the attacker's intents, risks, etc. A desirable $C_{0,a}(E)$ should be a function correlating all those relevant factors. P. Liu et al. [10] gave a concrete discussion in their proposed AIOS model composing attacker intent, objectives, and strategies, although the model is different form ours, the definition of cost can be used as well.

Suppose the goal will be achieved ultimately, none of attackers, excluding those brute-force ones, would prefer those cost-consuming schemes if they have

other options of cheaper ones, in another word, they would not use a large set of concurrent actions if a smaller subset can achieve the desired effect. If so, all our analysis are meaningless. The observation can be briefly described as follows,

Observation 1. For a particular target which can be finally cracked, provided prerequisites V, there must exist an optimal concurrent action set U by which a group of attackers E can achieve the goal with the minimum attack cost $C_{0,a}^{min}(E)$.

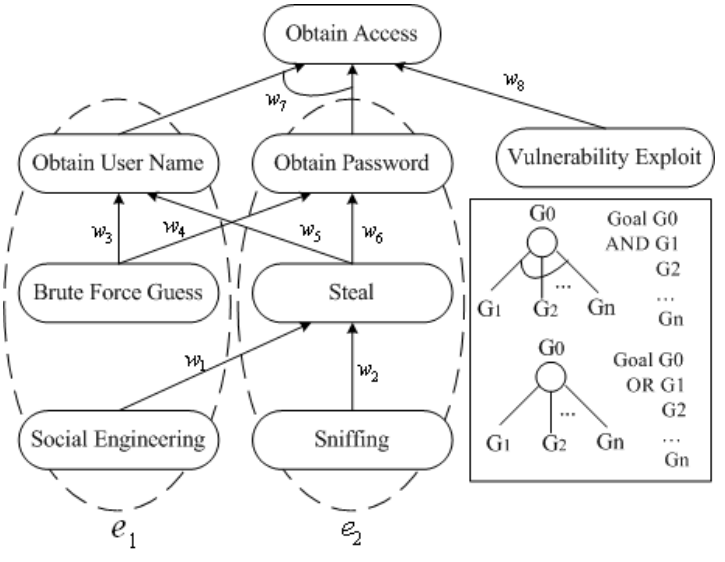


Fig. 2. Attack Scenario

More formally, we model the action-dependance state transition as a directed *state contact graph* $G =< S_A, W >$. The edges of the graph $W = V \times U$, where V is the set of preconditions for a particular joint action, and U is the collection of joint actions. Each directed edge represents a transition between two system states $s_i, s_j \in S_A$ at a certain stage. We represent each edge by a tuple $w =< s_i, s_j, \hat{u}_i, z_i >$ ($z_i \in \hat{v}_i$ is the exploitable preconditions) where s_i is the previous system state and s_j is the target system state with $\hat{u}_i \times z_i$. Edge w is thus $s_i[\hat{u}_i \times z_i] \rightarrow s_j$. Based on the model and properties, the observation can be generalized as a corollary,

Corollary 1. In the directed system state contact graph G, attackers E always intend to achieve the least-weight-path w_{min} from source node s_0 to the destination node s_a.

Figure 2 shows a simple attack scenario: attacker e_1 and e_2 cooperate to compromise a server, where attacker e_1 responsible for the stealing of *user name* and e_2 tries to get *password* by *steal* or *guess*; different means have different cost

represented by w_i; for the final state, attackers are able to access the server using a legitimate user name and password. From the attack graph we may find there are several ways to achieve the final goal, among which, one scheme with the least cost is that e_2 steals *user name* (w_5), and e_1 cracking the password by *brute Force Guess* (w_4). We cannot balance w_6 and w_4 here, since the risk of stealing a password might be very high. However, checking every possible subset of attack graph is exponential in the number of attacks, and finding the least-weight-path actually is *NP*-complete [17]. Also, it is worth noting here that the formulation of G is based on the assumption that attackers' actions have no loops, or excluding the wrong operations, recoveries, etc. Moreover, insights into the individual attacker's behavior might facilitate us to have a better understanding on a group of attacker's joint actions,

Corollary 2. For a particular attacker e_i, its action-dependence edges can only be regarded as a collection of discontinuous lines connecting two different system states.

For a particular attack scheme, e_i's operating traces can also be viewed as a sequence of state-related actions, namely, $\tau(e_i) = (u_0^i(s_0), u_1^i(s_1), \cdots, u_k^i(s_k))$, where $u_j^i(s_j)$ represents the action u_j^i is taken by attacker e_i in system state s_j, and it might be null if attacker e_i is not involved in a particular attacking stage. This is based on the assumption that with the knowledge of G, attacker e_i always knows which action to be executed in each stage. The general attacking scheme is thus the combination of all the attackers' action traces, i.e, $\tau(E) = (\hat{u}_0(s_0), \hat{u}_1(s_1), \cdots, \hat{u}_a(s_a))$. More generally, a group coordinated attackers' capacity can be measured by the tuple (U, N), where U is all the available actions (more specifically, the knowledge, skills, and assistant tools, etc.) available to the attackers, N is the total number of participators. Although the individual attackers in practice might have different capabilities and experiences, considering the common target and the internal sharing of the knowledge/skills/tools during the coordinated attacks, it is reasonable assume a group of attackers to be *homogeneous*. Based on the formulated model and the derived assumptions, also motivated by the attacker's incentive, we attempt to develop an algorithm inspired by those ideas from ant colony optimization (ACO) algorithms family [5, 7], called *Attackers Nondeterministic Trail Search (ANTS)* (formulated and designed in the Appendix), to search for such an attacking scheme as described in corollary 1. In the algorithm, each attacker is viewed as a context-awareness ant hunting foods (subgoals), while the observations z_i at each stage construct covert particular channels for ants' action-dependence context. The criteria for the termination of the algorithm can be set as required, it might be a goal that has been achieved or a minimum attack cost exceeding a certain threshold, while the subgoals can be viewed as some specific preconditions for the next target. Considering the specific correlation among attackers (by constraints Ω), the searching process can be accelerated by their inter-communication (i.e., the procedure *update_internal_state()* and *take_next_action($\hat{U}_\Omega$)*, which also avoids the algorithm getting into local optima.

4.2 Defender-Centric Analysis

Taking the same analytical model as the basis, defender can take advantage the inferred information for the prevention of MSCA's exploitation: (1) the model facilitate us to develop specific techniques to defend against or mitigate the attacker's action in both temporal and spacial spans; (2) if the attacker's graph G is well modelled by the prior vulnerability correlation and analysis, the *pivot* of the attacking scheme thus can be figured out and removed; (3) corresponding cost-saving countermeasures can be taken based on the estimates of system state transition with particular defense-objectives.

Vulnerability analysis has been taken as an effective methodology to examine security-related properties to enhance the dependability and security of computer systems, and many analytical models and tools [3,4,15] have been developed so far. Here, we do not focus our attention on the development of specific analytical tools, we would rather to examine one method which is capable of figuring out the key stages of an MSCA by assuming relative vulnerabilities have been recognized and attacking graphs have been constructed [17,18]. For a particular attacking graph G, if a set of edges connecting essential exploits, namely, *pivots* of the attack graph are cut off, the attackers would never achieved their goal successfully.

Observation 2. An exploit might depend on a set of preconditions v and undergo multiple elementary actions u in order to take advantage of a single vulnerability (or subgoal), the vulnerabilities and relative examiners therefore allow us to derive a cause-consequence relationship for each basic actions.

Same to the construction of attacker graph G, the correlation among vulnerabilities can also be extracted as a directed *state contact graph* $G' =< S_N, W' >$, where S_N is a set of underlying system states representing the state transitions under known vulnerabilities, while W' is the collection of abstracted edges connecting $s \in S_N$. Note here, defenders not only concern those state transitions resulting in attacks, but also seemingly normal transitions, therefore, $S_A \subseteq S_N$ and $|S| \leq |S_A \cap S_N|$. Also, $W' = V \times U$, where V is the preconditions and, U is vulnerable operations, $w \in W'$ thus denotes the multiple vulnerable operations u on several objects v that are involved in exploiting a vulnerability, in another word, changing system states. A corollary characterizing such property can be generalized as follows,

Corollary 3. In the directed graph G', there exist at least one path, usually a collection of edges w', from the source node s_0 to the destination node s_n, without which, graph G' would turn to a disconnected graph being cut into several parts.

Obviously, if such edges w' do exist, and can be found out approximately by heuristic algorithms in non-deterministic manners, defenders can easily understand adversary's intents by figuring out the key observations. Let us revisit figure 2, if the *password* can be saved safely causing w_6 infinite (or remove the password), attackers would never gain the access to server, actually this is a

simple case of key management and password-based access control. Since our basic assumption is that G' has been constructed by vulnerability analytical tools such as model-checking, a backward searching algorithm might be feasible and efficient to explore the objective with those desired properties. We intend to develop such an algorithm called *Attacker's Pivots Discovery by Backward Searching*, or APD-BS (shown in appendix), which can also derived from the ACO algorithm family. The main idea is generalized as follows: (1) select the most significant vulnerabilities resulting in the system compromised state s_n, i.e., for those observations with a higher probability $Pr\{z_i|s_n, \hat{u}\}$; (2) put the ants on the interested node, which is capable of tracing back those *neighbors* meeting constraints in probabilistic manners; (3) rank the edges with the amount of pheromone left by those ants walking from source node to end node, based on which, the most significant *pivots* can be figured out; (4) the above three steps are carried out iteratively until the termination criterion is met.

The main element of this metaheuristic algorithm is *ants*, which construct candidate traces (a complete trace is a solution) for the problem by individually and iteratively computation. A complete trace contains a collection of correlated observations being emitted during the state transitions between s_0 and s_n, while intermediate trace only contains parts of them. At each step, every ant k computes a set of feasible expansions to its current trace and moves to one of these probabilistically according to a probability distribution p_{ab}^k (ant k from observation a to the next one b) by combining and specifying following two values, (A) the attractiveness ε_{ab} of the move, as computed by some *a priori* desirability of that move; (B) the trail level τ_{ab} of the move, indicating how proficient is has been in the past to make that particular move, which is essentially *a posteriori* indication of the desirability of that move.

Taking into account our specific concern, and based on the pre-knowledge that $H(a) = Pr\{a|s_i, u_{i-1}\}$ and $H(b) = Pr\{b|s_i, u_{i-1}\}$, we define the attractiveness of the ants' move as the correlation coefficient of those two probabilities,

$$\varepsilon_{ab} = \frac{Cov(H(a), H(b))}{\sqrt{D(H(a))}\sqrt{D(H(b))}} \tag{1}$$

where $Cov(H(a), H(b))$ is the covariance of $H(a), H(b)$ and $\sqrt{D(H(a))}$ is the variance of $H(a)$, and the ant's pheromone trail update rule can be defined as,

$$\tau_{ab} = \rho\tau_{ab} + \tau_0(1 - \frac{c_i}{\overline{c}}) \tag{2}$$

where $\rho \in (0, 1]$ is a coefficient such that $1 - \rho$ represents the decrease of trail between two generations of complete traces, and τ_0 is the initial trail level which is usually fixed to be an arbitrary small positive value. $\overline{c}$ is a moving average on the cost of the last l traces, and c_i is the cost of the ith ant's trace. We can see that the cost for system state transition which generates a essentially is taken as an hidden guidance for ant's movement, this is reasonable if we consider the fact that, as we have discussed previously, attackers always manage to seek the

cheaper actions for system state transition. Henceforth, the probability for ant k to make the move is given by

$$p_{a,b}^{k} = \begin{cases} \dfrac{\tau_{ab} \cdot \xi + \varepsilon_{ab} \cdot (1-\xi)}{\sum\limits_{x \in tabu_k} \tau_{ax} \cdot \xi + \varepsilon_{ax} \cdot (1-\xi)}, & \\ & \text{if } b \notin tabu_k \\ 0, \text{otherwise} & \end{cases} \tag{3}$$

where the sum is over all the feasible moves and $\xi \in (0, 1]$ is a control parameter balancing the relative importance of the trail τ_{ab} and the attractiveness ε_{ab}, $tabu_k$ is an observation dependent set for kth ant's feasible moves. In this sense, p_{ab}^{k} actually is a tradeoff between the desirability of the current move and the past traces. Note that since the development of our algorithm is based on the assumption that observatons/vulnerabilities have already been characterized by some particular assistant tools, three matrices F, H, Q in the analytical mode can be populated with the prior knowledge in advance and thus serve the algorithm.

5 Case Studies

This section aims to conduct two case studies to show the application of our proposed models and algorithms to real attack scenarios.

5.1 Intrusion-Aware Automatic and Adaptive Control

The general MPO-MDP model formulated in the section 3 presents us a formal way to utilize the probabilistic state estimate as a sufficient statistic and optimal design defense strategies. The basic idea proposed in [8], in which a host-based autonomic defense system was developed for the provision of the system survivability. Based on the functional decomposition, defender can select the proper countermeasures according to the feedback from system state estimator. The decomposition of the model contains two key parameters: (a) $\mathcal{I}_t$, all the information received by the defender prior to selecting the tth action u_t, including previous observation z_{t-1}, action u_{t-1}, and priori information $\mathcal{I}_{t-1}$; (b) $\mathcal{B}_t$, the system state estimate in stage t, which is a n-length column vector, and the ith element $b_t(i) = Pr(s_i|\mathcal{I}_t)$ representing the relative confidence that state s_i is indeed the true system state in stage t. Here $\mathcal{B}_t$ can be taken by the defender as the basis to estimate current system state via recursive derivation $\mathcal{B}_t = \varphi(z_t, u_{t-1}, \mathcal{B}_{t-1})$, where φ is an estimation policy. Based on the probabilistic state estimate, the defender thus can select the candidate countermeasures according to a response policy μ, namely, $u_t = \mu(\mathcal{B}_t)$. The defense policy subject to the key element $c_{i,j}(u)$, the cost for state transition, which can be defined specifically to achieve the tradeoffs among such costs relevant with system failure, system maintenance, and responses, and the objective of defender is just to save the total cost during operation stages.

We also have developed an automatic detection coordinator [22] to construct a multi-layered boundary defense system by being formulated as a MPO-MDP. In

the model, the general response policy is several parametric anomaly detectors' coordinated action, i.e., at each detection stage, the basic anomaly detectors take the concurrent actions associated with their respective observation vectors and system state estimates. The general reward signal allows those basic ADs to adjust their behavior periodically to adapt to the changing system states. However, it it worth noting that the determination of the optimal countermeasures and the minimization of the given objective function is typically not possible, and thus approximation methods and heuristics must be applied to find near-optimal policies, while our ANTS and APD-BS algorithm may serve as useful tools to solve this problem.

5.2 Worm Enclave Identification

As mentioned, epidemic-style worms can be viewed as a special case of MSCA. We propose *worm enclaves* here to represent the host responsible for launching a propagating worm attack and the links among those infected hosts which construct the initial stages of the attack tree. Formally, assume the network communication between end nodes is a directed contact graph $G =< V \times E >$ (notations are self-contained in this section), where vertices $V = H \times T$ representing the hosts in a certain stage t, E is the set of edges representing network flows between hosts, the worm enclaves is thus a graph G_t at a particular stage t. A similar problem has been formulated in [21], and they proposed a solution using random moonwalks. Regardless of the scanning strategies and the specific exploitable vulnerabilities, their method aims to identify the worm origin without any *a priori* knowledge about the attack. The key observation support the algorithm is that the communication flows among the attackers and the associated set of compromised hosts constructs a causal tree with the root node of initial infected host (or worm origin). Obviously, it is hard to detect out the attacks by observing the communication flows individually from the single host, since the behavior only can be figured out collectively from a set of hosts and flows. With the similar motivation, our algorithm APD-BS aims to find out the worm enclaves in terms of tree-structured subgraphs as defined previously, moreover, the algorithm can not work in general computer networks, but also in P2P networks having special link mode.

Firstly, we assume a network contains N hosts and E edges; for a particular host i, the number of its incoming communication flows at time t is $I(t)$ and the number of outgoing links is $O(t)$. The system state the host undergoes is assumed as a set $s_t = \{Nrm, Spt, Inf\}$, where Nrm means that system is under normal operation; Spt means the system is probably under attacks (it may be further classified according to the lethality); and Inf means the system has been infected. For the ant locating in the host, it may take action $u_i = \{W, B, G\}$ (W is white for normal, B is black for infected, G is gray for suspected) according to the estimate of s_t. The cost of action thus can be simply defined as follows,

$$c_{ij}(u_i) = \begin{cases} \gamma_1, \text{if } s_j = Nrm, u_i = B \text{ (false positive)} \\ \gamma_2, \text{if } s_j = Inf, u_i = W \text{ (false negative)} \\ \gamma_3, \text{if } s_j = Spt, u_i = W \\ 0, \text{otherwise} \end{cases} \tag{4}$$

Secondly, we need to specify the key parameters and main steps of the algorithm: (a) $\lambda_t = I(t)/O(t)$ is the ratio of the number of incoming flows and outgoing flows, $p_0 = \lambda_0$ thus can be initialized using historic traffic records logged by the networks, and usually $\lambda_t < 1$ for an infected host since it generally originates more flows than it receives; (2) for a particular ant a locating in the host i, its observation vector at time t is z_t, which is essentially a row vector composed by $I(t)$ and $O(t)$ and indexed by the nodes the communication flows respectively, i.e.,

$$z_t = \begin{pmatrix} O_{h_1}(t), O_{h_t}(t), ..., O_{h_m}(t) \\ I_{h_1}(t), I_{h_t}(t), ..., I_{h_m}(t) \end{pmatrix} \tag{5}$$

where $m \geq n$ in order to contains those flows with new or invalid host address.

Thirdly, N initialized ants are put on N hosts individually and then run the algorithm. For a particular host a, it has to calculate the similarities with its neighbors, i.e., ε_{ab}, and thus updates its pheromone with equation (2) and (3). With a predefined threshold, the algorithm eventually outputs several $Pivots$, which constructs the worm enclaves with corresponding flows.

One advantage of the algorithm is that it can be implemented both offline and online. With the prior knowledge of the network bing infected by worms, the algorithm is able to find out those origin hosts launching the attacks by static backward searching and thus further investigation and action can be taken. With the prior knowledge of the traffic records logged by the network (training data), the probabilities in the analytical model F, H, Q can be populated and thus enables the algorithm to work online by dynamic backward searching and take corresponding actions in time. This actually can prevent the worms from further propagation in realtime. Also, some computational cost and time complexity need to be considered for the evaluation of the algorithm. Suppose the algorithm takes total T steps (time units) for one-pass searching, the time complexity is $O(T)$, but one-pass usually does not guarantee its convergence, K times of running results in the complexity $K \cdot O(T)$; while for the space complexity, at each step, ants have to maintain their own observation vector z_t. For each ant, the size of observation vector is $2N$, so the overall space complexity is $O(N^2)$ (where N is the number of hosts). For the computational cost, one interesting case is that ants will not bring much trouble to the normal host, since they will move away from it in several steps and never return. While for those infected hosts which have already suffered from worms, the ants contribute a little to the overall cost compared with those of worms.

6 Concluding Remarks and Future Work

To cope with MSCA, in this paper, we presented a two-sided model, for its representation and analysis. The model was formulated as a MPO-MDP to

characterize both attacker and defender's behavior, and the basic properties of MSCA were also analyzed. Based on the characterization, an ANTS algorithm was developed to search for attack schemes with minimum action cost; and another searching algorithm APD-BS was developed to capture the most significant observations of a successful attack. Two well-posed problems have been analyzed as case studies, together the preliminary analysis showing their promising properties.

The future work is mainly focused on the specific implementation and evaluation of our two algorithms, and meanwhile developing more efficient local search (LS) algorithms to accelerate the convergence speed of those algorithms. Although we have given some preliminary theoretical analysis on the algorithms in specific application scenarios, both of them are also expected to be evaluated in simulation-based testing environments with real trace data.

References

1. D. Aberdeen, "A Survey of Approximate Methods for Solving Partially Observable Markov Decision Processes," *National ICT Australia Report*, Canberra, Australia, Dec. 8, 2003.
2. S. Braynov and M. Jadiwala, "Representation and Analaysis of Coordinated Attacks", *Proceedings of the 2003 ACM workshop on Formal methods in security engineering*, pp.43-51, 2003.
3. H. K. Browne, W. A. Arbaugh, J. McHugh, W. L.Fithen, "A Trend Analysis of Exploitations," *Proceedings of 2001 IEEE Symposium on Security and Privacy (S&P2001)*, pp.214-229, May 14-16, 2001, Oakland, California, USA.
4. S. Chen, Z. Kalbarczyk, J. Xu, R. K.Iyer, "A Data-Driven Finite State Machine Model for Analyzing Security Vulnerabilities," *2003 International Conference on Dependable Systems and Networks (DSN'03)*, pp.605-614, San Francisco, 2003.
5. O. Cordon, F. Herrera, T. Stutzle "A Review on the Ant Colony Optimization Metaheuristic: Basis, Models and New Trends", *Mathware and Soft Computing* 9(2-3), pp.141–175, 2002.
6. K. Daley, R. Larson, J. Dawkins, "A Structural Framework for Modeling Multi-Stage Network Attacks", *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'02)*, pp.5-10, Vancouver, Canada, 2002.
7. M. Dorigo, V.Maniezzo, and A. Colorni, "The Ant System: Optimization by a colony of cooperating agents," *IEEE Trans. Syst. Man, Cyber. Part B,* vol.26, pp.29-41, 1996.
8. O. Patrick Kreidl, T. M. Frazier, "Feedback Control Applied to Survivability: A Host-Based Autonomic Defense System," *IEEE Trans. on Reliability,* Vol.53, No.1, pp.148-166, March 2004.
9. E. Levy, "Worm Propagation and Genetric Attacks", *IEEE Security and Privacy*, Vol. 3, No. 2 pp. 63-65, March/April 2005.
10. P. Liu, w. Zang, M. Yu, "Incentive-Based Modeling and Inference of Attacker Intent, Objectives, and strategies," *ACM Transactions on Information and System Security,* Vol. 8, No. 1, Feb. 2005, Pages 78-118.
11. S. Mathew, C. Shah, S. Upadhyaya, "An alert Fusion Framework for Situation Awareness of Coordinated Multistage Attacks", *Proceedings of the Third IEEE International Workshop on Information Assurance (IWIA'05)*, pp. 95-104, College Park, MD, USA, 2005.

12. J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher, *Internet Denial of Service: Attack and Defense Mechanisms,* Prentice Hall, ISBN: 0-13-147573-8, 2005.
13. P. Ning, Y. Cui, D. S.Reeves, D. Xu, "Techniques and Tools for Analyzing Intrusion Alerts", *ACM Trans. on Information and System Security*, Vol. 7, No.2, May 2004, Pages 274-318.
14. C. Phillips, L. Swiler, "A graph-based system for network-vulnerability analysis," *Proceedings of the 1998 Worshp on New Security Paradigms,* pp.71-79, 1998.
15. R. W. Ritchey, P. Ammann, "Using Model Checking to Analyze Network Vulnerabilities," *2000 IEEE Symposium on Security and Privacy (S&P'00)*, pp.156-165, May 14-17, 2000, Oakland, California, USA.
16. B. Schneier, "Attack Tress," *Dr. Dobb's J.*, vol. 12, Dec. 1999; www.ddj.com/articles/1999/9912
17. O. Sheyner, J. Haines, S. Jha, etc., "Automated Generation and Analysis of Attack Graphs," *Proceedings of the 2002 IEEE Symposium on Security and Privacy (S&P'02)*, pp. 273-284, May 12-15, 2002, Oakland, California, USA.
18. L. Swiler, C. Phillips, D. Ellis, and S. Chakerian, "Computer-attack graph generation tool," *DARPA Information Survivability Conference and Exposition,* pp.146-161, Anaheim, California, 2001.
19. F. Valeur, G. Vigna, C. Kruegel, et al., "A Comprehensive Approach to Intrusion Detection Alert Correlation", *IEEE Trans. on Dependable and Secure Computing*, pp.146-169, Vol.1, No.3, July-September 2004.
20. N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "A taxonomy of computer worms", *Proceedings of the 2003 ACM CCS workshop on Rapid Malcode (WORM'03)*, pp.11-18, Washington. DC, USA 2003.
21. Y. Xie, V. Sekar, D. A. Maltz, M. K. Reiter, H. Zhang "Worm Origin Identification Using Random Moonwalks," *Proceedings of 2005 IEEE Symposium on Security and Privacy, (S&P 2005)* pp.242-256, Oakland, CA, May 2005.
22. Z. Zhang, H. Shen, "Constructing Multi-Layered Boundary to Defend Against Intrusive Anomalies: An Autonomic Detection Coordinator", *Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)*, pp.118-127, Yokohama, Japan, June 2005.

Appendix: Two Heuristic Algorithms

ANTS: Attackers Nondeterministic Trail Search

The formulated problem essentially belongs to a group of (constrained) shortest path problems which is *NP-complete* hard and can only be solved by approximate algorithms. Therefore, to solve this problem, the ANTS algorithm can be specialized as follows: (a) there is a set of constraints Ω for the concurrent actions of attackers; (b) the available actions of attackers is a finite set $U = \{u_1, u_2, \cdots, u_l\}$; (c) for every system state, a set of actions can be taken over U as $\delta =< u_r, u_s, \cdots, u_w, \cdots >$. Assume $\hat{U}$ is the set of all possible coordinated actions, we denote by $\hat{U}_\Omega$ the set of feasible sets with respect to the constraints Ω. The elements in $\hat{U}_\Omega$ define the feasible actions. $|\delta|$ is the size of a set δ, i.e., the number of actions in the set; (d) there is a neighborhood structure among concurrent action sets defined as follows: δ_2 is an adjacent action set of δ_1 if (1) $\delta_1 \in U$ and $\delta_2 \in U$, (2) δ_2 can be reached from δ_1 by an additional logical action, i.e., $\delta_2 =< \delta_1, u_r >$($u_r \notin \delta_1$, while $u_r \in \delta_2$). The feasible adjunct action

of δ_1 is the set containing all action sets $\delta_2 \in \hat{U}_\Omega$; if $\delta_2 \notin \hat{U}_\Omega$, δ_2 is viewed as the infeasible adjunct action set of δ_1; (e) an attacking scheme AS is an element of $\hat{U}_\Omega$ verifying all the requirements; (f) there is an attack-specific cost $Cost$ to evaluate every AS.

```
void ANTS(U, Num)
  Initialize (U, Num, minimum_cost, z0);
  //U is action set, Num is the number of attacker members,
  // z0 is preconditions in the initial state
  while (termination_criteria_not_met)
    repeat in parallel for k = 1 to Num
      initialize_ant(k);
      L = update_ant_memory(); //understanding of the current concurrent action
      while (current_system_state ≠ target_system_state)
        compute_transition_probability (F, H);
        take_next_action(Û_Ω);
        L = update_internal_state();
      end while
      if (state_transited)
        compute_reward_signal(r_t);
        cost = compute_transition_cost();
        deposit_pheromone_update(r_t);
      end if
      minimum_cost = update_minimum_cost(cost);
      get_ant_trail(k);
    end repeat in parallel
    get_concurrent_action_list();
    if (attacking_goal_not_achieved)
      return("Attack Failed!");
    end if
  end while
```

APD-BS: Attacker's Pivots Discovery via Backward Searching

```
void APD-BS()
Initialize (p_o, current_observation, minimum_cost);
//current_observation is a set of observations that z0 currently available
//p_o is apriori probability of observation selection, minimum_cost = ∞
Put ants on those N selected nodes with Pr{z0|s_n, û} > p_o;
while (termination_criteria_not_met)
  repeat in parallel for k = 1 to N
    initialize_ant(k);
    L = update_ant_memory(); //get current context info.
    while (current_observation ≠ ∅)
      for(j = 1 to number_of_current_observation)
        a = get_current_observation[j];
        b = neighbor(a);
        //neighbor is defined as those observations resulting in the same system state
        ε_ab = attractiveness_compute(a, b);
```

```
      end for
      move_backward(); //make the move according to equation (3)
      compute_move_cost(); //update the k-th ant's trace cost
      append_new_observation(tabu_k); //update the k-th ant's new feasible move
    end while
    cost = get_current_trace_cost();
    if(cost < minimum_cost)
      minimum_cost = cost; //update the k-th ant's moving costs
      update_trace();//update the k-th ant's trace
    end if
  end repeat in parallel
  for (each_move_of_ants())
    Update_trail_level();//update N ants' pheromone trail level by equation (2)
  end for
end while
Pivots_selection(); //select those observations connecting
                    //by those edges with the most amount of pheromone
```

A Time-Frame Based Trust Model for P2P Systems*

Junsheng Chang, Huaimin Wang, and Gang Yin

School of Computer, National University of Defense Technology,
HuNan Changsha 410073, China
{cjs7908, whm_w, jack_nudt}@163.com

Abstract. Two major challenges regarding peer's trust valuation in P2P systems are how to cope with strategically altering behaviors and dishonest feedbacks of malicious peers efficiently. However, the trust models employed by the existing systems do not provide adequate support to coping with quick changes in peers' behavior and aggregating feedback information, then we present a time-frame based trust model. We incorporate time dimension using time-frame, which captures direct experiences and recommendations' time-sensitivity, we also introduce four trust parameters in computing trustworthiness of peers, namely, trust construction factor, trust destruction factor, supervision period factor and feedback credibility. Together, these parameters are adjusted in time using feedback control mechanism, thus, trust valuation can reflect the dynamics of the trust environment. Theoretical analysis and simulation show that, our trust model has advantages in modeling dynamic trust relationship and aggregating feedback information over the existing trust metrics. It is highly effective in countering malicious peers regarding strategic altering behavior and dishonest feedbacks of malicious peers.

Keywords: peer to peer, trust model.

1 Introduction

P2P (Peer-to-Peer) technology has been widely used in file-sharing applications, distributed computing, e-market and information management [1]. The open and dynamic nature of the peer-to-peer networks is both beneficial and harmful to the working of the system. Problems such as free-riders and malicious users could lead to serious problems in the correct and useful functioning of the system. As shown by existing work, such as [2, 3, 4, 5, 10, 11, 12], reputation-based trust management systems can successfully minimize the potential damages to a system by computing the trustworthiness of a certain peer from that peer's behavior history. However, there are some vulnerabilities of a reputation-based trust model. One of the detrimental vulnerabilities is that a malicious node may strategically alter its behavior in a way that benefits itself such as starting to behave maliciously after it attains a high reputation. Another widely recognized vulnerability is the shilling attack [6] where

* Supported by National Basic Research Program of china under Grand (No.2005CB321800), National Natural Science Foundation of China under Grant (No.90412011) and National High-Tech Research and Development Plan of China under Grant (No. 2003AA115210, 2003AA115410, 2005AA112030).

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 155 – 165, 2006.
© Springer-Verlag Berlin Heidelberg 2006

malicious nodes submit dishonest feedback and collude with each other to boost their own ratings or bad-mouth non-malicious nodes [7].

With these issues in mind, we present a dynamic P2P trust model based on time-frame for quantifying and assessing the trustworthiness of peers in P2P system. We incorporate time dimension using time-frame, which captures experience and recommendation's time-sensitivity, we also introduce four trust parameters in computing trustworthiness of peers, namely, trust construction factor, trust destruction factor, supervision period factor and feedback credibility. Together, these parameters are adjusted in time using feedback control mechanism, thus, trust valuation can reflect the dynamics of the trust environment. Theoretical analysis and simulation show that, our proposed trust model has advantages in modeling dynamic trust relationship and aggregating feedback information over the existing trust metrics. It is capable to detect and penalize the suddenly misbehaving peers, as well as those that exhibit oscillatory malicious behavior. Moreover, the trust model can filter out dishonest feedbacks effectively.

The reminder of this paper is structured as follows. In the second section, the related works are introduced; the dynamic P2P trust model based on time-frame will be illuminated in the third section; and in the fourth section, a simulation about this model is laid out. Conclusions and future works are in the end.

2 Related Work

Marsh [8] is among the first to present a formal trust model, incorporating properties of trust from psychology and sociology. It is well-founded yet complex model; it does not model reputation in the trust model. This model has been further extended by Abdul-Rahman and Hailes to address reputation-based trust in virtual communities [9]. A number of reputation mechanisms for P2P systems [2, 3, 4, 5, 10, 11, 12] have been proposed following the trust models from [8] and [9].

To cope with strategically altering behaviors of malicious peers efficiently, Xiong and Liu [5] show that the average metric does not cope with “dynamic personality behavior. They remedy the problem in their adaptive metric which narrows the window of experiences taken into consideration by the average. This indeed increases the sensitivity of the metric in respect to changes of behavior. However, their metric is unable to detect and penalize oscillatory malicious behavior. If the oscillatory behavior is not detected, a malicious peer could always find the rules for trust increasing and decreasing and accordingly adjust her cheating behavior to get the maximum payoff from the system while also keeping a reasonable reputation [13]. Claudiu-Duma investigates the requirements on the dynamics of trust in P2P systems and proposes a versatile trust metric which satisfies these requirements [14]. In particular, the proposed metric is capable to detect and penalize both the sudden changes in peers’ behavior and their potential oscillatory malicious behavior. But, it does not differentiate the direct experiences and recommendations, so, one peer can not avoid to a certain extent collaborating with the peer that has given it bad performances, even if it gives the rest of the network good ones. Huang’s work [15] shows that subjective logic can not deal with trust dynamics efficiently. Cvrˇcek [16] also shows that Dempster-Shafer’s belief theory as well as the average trust metric,

both extensively used for modeling trust, do not have the properties required for dealing with trust dynamics. Instead, Cvrˇcek arrives at the conclusion that simple formulas might work. Thus, our model uses simple formulae and works well.

To effectively aggregate feedback information, PeerTrust [5] proposes to use a personalized similarity measure (PSM for short) to rate the feedback credibility of another node x through node n's personalized experience, the evaluation of recommendation credibility is depending on the common set of peers that have interacted with requestor and the recommendatory peers. As the increase of peers' quantity, the common set is always very small [17]. This evaluation algorithm of feedback credibility is not reliable. Research [11] proposes an algorithm based on the maximum likelihood estimation to enhance the veracity of the recommendation credibility in the situation of small common set of peers. Eigentrust [3] considers the recommendation trust as being equal to the service trust. This metric is not suitable in circumstances where a peer may maintain a good reputation by providing high quality services but send malicious feedbacks to its competitors. Research [10] proposes the weighted majority algorithm (WMA), the main idea is to assign and tune the weights so that the relative weight assigned to the successful advisors is increased and the relative weight assigned to the unsuccessful advisors is decreased. But, the approaches mentioned above don't consider more complex malicious strategies, for example, peers could try to gain trust from others by telling the truth over a sustained period of time and only then start lying, colluding peers could inflate reputation using unfair ratings flooding.

3 Time-Frame Based Trust Model

The main issues characterizing the reputation systems for P2P are the trust metric (how to model and compute the trust) and the management of reputation data (how to securely and efficiently retrieve the data required by the trust computation) [2]. Efficient and secure feedback information management in P2P systems has been addressed in [2] and [18]. These systems allow efficient retrieval of all the experiences necessary for trust computation, and we will assume the existence of such a system underlying our trust model. Thus, the focus of this paper is on trust metrics, which, as motivated in the introduction, are capable to cope with strategic altering behavior and dishonest feedbacks of malicious peers.

3.1 Trust Evaluation Algorithm

To get more accurate trust value of the peer reflecting the 'fresh' trust status, time property should be considered. Suppose peer i is the end-peer collecting the feedbacks about the trust status of peer j during period $[t_{start}, t_{end}]$. If $t_{end} - t_{start}$ is divided into different sub-periods $\{t_1, t_2, \ldots, t_n\}$ ($t_1 < t_2$, $t_2 - t_1 = \Delta t$), we call the sub-period time-frame. Let successive time-frame be numbered with consecutive integers starting from zero. R_{ij}^{n} denotes the raw trust value of peer j computed as an aggregation of the feedbacks collected by node i in time-frame n.

There are two kinds of trust relationships among peers, namely, direct trust and indirect trust [19]. The direct trust of peer i to peer j can be evaluated from the direct

transaction feedback information between i and j. The indirect trust of i to j can be computed according to the transaction feedback information of peers who have interacted with j.

In time-frame n, the trust value of peer i to peer j's service is denoted by , which is defined in formula (1).

$$R_{ij}^{n} = \lambda * D_{ij}^{n} + (1-\lambda) * \sum_{r \in I(j)} \frac{Cr_{ir} * D_{rj}^{n}}{\sum_{r \in I(j)} Cr_{ir}} \tag{1}$$

Where D_{ij}^{n} denotes the direct trust value of i to j, the "self-confidence factor" is denoted by λ, which means that how a peer is confident to its evaluation of direct trust value. $\lambda = h / H$, h is the number of the direct interactions considered, and H is the maximum number to be considered for a peer, and the upper limit for λ is 1. $I(j)$ denotes the set of recommendation peers, namely, the peers interacted with j directly except peer i. $\forall r \in I(j)$ is a recommendation peer, its recommendation credibility is denoted by Cr_{ir}.

Let e_{ij} denote the feedback evaluation from i to j in time-frame n, where real number $e_{ij} \in [0,1]$. The direct trust value from i to j is defined in formula (2).

$$D_{ij}^{n} = \frac{1}{m} * \sum e_{ij} \tag{2}$$

Where m is the number of transactions between peer i and peer j in time-frame n.

The dependable trust value T_{ij} is evaluated based on the raw trust value sequence $< R_{ij}^{1}, R_{ij}^{2}, \ldots, R_{ij}^{n} >$, R_{ij}^{n} is the most recent one. For simplicity, we assume that the trust values of peers are updated periodically within each time-frame. For peer j, peer i maintains a pair of factors (i.e. current trust construction factor α and current trust destruction factor β), enhancing the dependability of T_{ij} with respect to sudden behavioral changes of node j. α and β satisfy the constraint $\alpha < \beta$, which implies that more efforts are needed to gain the same amount of trust than to loose it [5]. α and β are modified when a trust degradation event is triggered by the fact that the coming raw trust value is lower than the trust invested. Upon a trust degradation event, the target peer j is put under supervision. His α is decreased and β is increased. If the peer j does not conduct any trust degradation event during the supervision period, α and β are reset to the initial values. Otherwise, they are further decreased and increased respectively, so it can detect and penalize the suddenly misbehaving peers. Current supervision period of an peer increases each time when he conduct a trust degradation event, so that he will be punished longer next time, which means an entity with worse history is treated harsher, thus it can also

detect and penalize also to detect and penalize the possibly long term oscillatory behavior.

For each peer j, peer i maintains a trust vector comprising five fields: current dependable trust value T_{ij}, current trust construction factor α, current trust destruction factor β, current supervision period $cPeriod$, rest of supervision period $sRest$. Once peer i has calculated the raw trust value (R_{ij}^n) for the peer j in time-frame n, peer i can update the trust vector for node j in time-frame n using TrustUpdate algorithm. TrustUpdate algorithm includes the following parameters: initial trust construction factor α_{ini} and trust destruction factor β_{ini}, initial penalty ratios for trust construction factor, trust destruction factor and supervision r1, r2 and r3 such that r1, r2 ∈ (0, 1) and r3 > 1, trust degradation event threshold ε. We define T_{ij} as a reinforcement learning update rule for non-stationary problems [20]. If integrating R_{ij}^n will increase trust value, the learning rate $\gamma = \alpha$, otherwise, $\gamma = \beta$.

```
Algorithm TrustUpdate( R_ij^n , T_ij^n )
  Input: R_ij^n    Output: T_ij^n
  if R_ij^n − T_ij^(n−1) > ε          //put under supervision
       β = β + r1 * (1 − β)
       α = r2 * α
       sRest = sRest + cPeriod
       cPeriod = r3 * cPeriod
       γ = β                  // the learning rate
  else
       γ = α
       if sRest > 0
           sRest = sRest - 1
           if sRest = 0    //restore α and β
                α = α_ini
                β = β_ini
  T_ij^n = (1 − γ) * T_ij^(n−1) + γ * R_ij^n
  return T_ij^n
```

3.2 Feedback Reliability

An important challenge in building a reputation system is to make it robust to misleading or unfair feedback. A malicious peer may submit dishonest feedbacks in order to boost the ratings of other malicious peers or bad-mouth non-malicious peers. The situation is made much worse when a group of malicious peers make collusive

attempts to manipulate the ratings. To solve the above problems, we introduce feedback reliability and update it after a time-frame.

Malicious peers can consider more complex malicious strategies, for example, colluding peers could inflate reputation using unfair ratings flooding, peers could try to gain trust from others by telling the truth over a sustained period of time and only then start lying. The effect is mitigated by two properties of our approach. First, trust valuation considers second-hand feedback information by in the limited frequency. Second, feedback credibility is updated after a time-frame then it can detect dishonest feedback quickly.

In time-frame n, let $CSet(i,r)$ denote the common set of peers that have interacted with both peer i and peer r, denotes the feedback difference between peer i and peer r is denoted by $diff_{ir}^{n}$. $diff_{ir}^{n}$ can be computed in equation 3.

$$diff_{ir}^{n} = \frac{\sum_{j \in CSet(i,r)} \left| D_{ij}^{n} - D_{rj}^{n} \right|}{\left| CSet(i,r) \right|} \tag{3}$$

Where D_{ij}^{n} and D_{rj}^{n} are the direct trust values from peer i and peer r to common interaction peer j respectively.

Based on the feedback difference $diff_{ir}^{n}$ between peer i and peer r, peer i can update the feedback credibility of peer r in equation 4.

$$Cr_{ir} = \begin{cases} Cr_{ir} + \frac{(1 - Cr_{ir})}{2} * \left(1 - \frac{diff_{ir}^{n}}{\theta} \right) & diff_{ir}^{n} < \theta \\ Cr_{ir} - \frac{Cr_{ir}}{2} * \left(1 - \frac{\theta}{diff_{ir}^{n}} \right) & otherwise \end{cases} \tag{4}$$

Where θ is the maximal tolerance of peer i to peer r for feedback bias. In this way, if a reporting peer r is malicious, its feedback credibility is gradually reduced when its opinion does not match that of peer i. And peer r with a lower credibility value therefore contributes less to the aggregated reputation at peer i.

4 Evaluation and Comparison

We performed initial experiments to evaluate our proposed trust model and show its effectiveness in countering malicious peers regarding strategic altering behavior and dishonest feedbacks of malicious peers. In our trust model, the quality for a peer to be a SP (service provider) is independent of the quality for a peer to be a rater which submits feedback after an interaction. We first define the types of qualities of both SPs and raters used in our evaluation. Three types of behavior patterns of SPs are studied: good peers, fixed malicious peers and dynamic malicious peers. Good peers and fixed malicious peers provide good services and bad services without changing their qualities once the simulation starts respectively. Dynamic malicious peers alter their behavior strategically. The behaviors of peers as raters can be one of the three

types: honest peers, fixed dishonest peers and dynamic dishonest peers. Honest and fixed dishonest peers provide correct and incorrect feedback without changing their patterns respectively. Dynamic dishonest peers provide correct feedback strategically, we only consider the dynamic dishonest peers which tell the truth over a sustained period of time and only then start lying. Our initial simulated community consists of N peers, N is set to be 128. The percentage of the bad SPs is denoted by pb, the percentage of the bad raters is denoted by pf. Table 1 summarizes the main parameters related to the community setting and trust computation. The default values for most experiments are listed. The experiments are programmed in JAVA.

Table 1. Simulation Parameters

	Parameter	Description	Default
Community Setting	N	number of peers in the community	128
	pb	percentage of malicious peers in the community	25%
	pf	percentage of dishonest raters in the community	80%
Trust Computation	λ	self-confidence factor	dynamic
	α_{ini}	initial trust construction factor	0.05
	β_{ini}	initial trust destruction factor	0.1
	ε	trust degradation event threshold	0.05
	$r1$	penalty ratio for trust destruction factor	0.1
	$r2$	penalty ratio for trust construction factor	0.9
	$r3$	penalty ratio for supervision period	2
	θ	the maximal tolerance for feedback bias	0.1

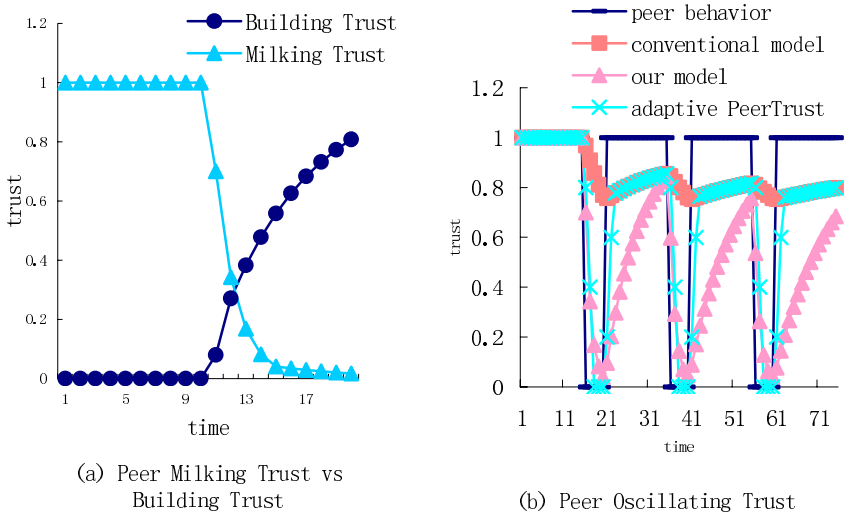
4.1 Effectiveness Against Strategic Altering Behavior of Peers

The goal of this experiment is to show how trust model we proposed works against strategic dynamic personality of peers. We use the same approach in PeerTrust, we focus on the changing behaviors of peers without simulating dishonest feedback in this experiment. We simulated a community with all good peers but a dynamic malicious peer with dynamic personality. We simulated three changing patterns. First, the peer builds trust and then starts milking it. Second, the peer is trying to improve its trust. Third, the peer oscillates between building and milking reputation, this peer is configured to regularly cheat, with a rate of 25%. That is, after each 30 good interactions the malicious peer behaves badly for 10 interactions. The experiment proceeds as peers randomly perform transactions with each other and a good peer is selected to compute the trust value of the malicious peer periodically. In a time-frame, each peer performs 10 interactions with other peers.

Figure 1(a) shows the computed trust value of both the peer who is milking its reputation and the peer who is building its reputation. The trust model we proposed can rapidly react to changes in peer behavior, moreover, it is consistent with the principle of quick drop and lent raise of trust. A peer is quickly detected for its bad

behavior but it cannot simply increase its trust value quickly by acting well for a short period so the cost of rebuilding reputation is actually higher than the gain of milking it.

Figure 1(b) show the computed trust values of the peer who is oscillating between building and milking its trust by our trust metric, the conventional average metric and the adaptive PeerTrust. The average metric is less sensitive to changes in peer behavior. Comparatively, both adaptive PeerTrust and our trust metric can rapidly react to changes in peer behavior, and their sensitivities do not depend on the number of accumulated experiences. However, the sensitivity of adaptive PeerTrust is the same for positive and negative changes, contracting the principle of quick drop and lent raise of trust. Moreover, adaptive PeerTrust can not penalize the oscillatory behavior of a malicious peer. On the contrary, by adjusting the trust construction factor, the trust destruction factor and supervision period factor our dynamic metric is capable of dealing with oscillating peers.



(a) Peer Milking Trust vs Building Trust

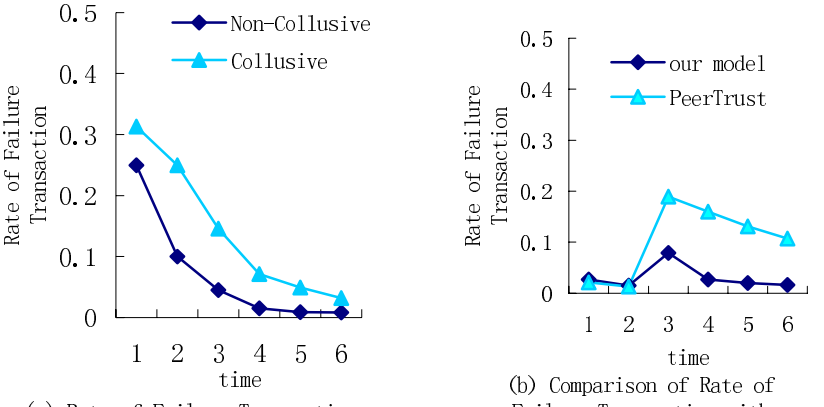
(b) Peer Oscillating Trust

Fig. 1. Effectiveness against strategic altering behavior of peers

4.2 Effectiveness in Aggregating Feedback Information

This simulation evaluates the immunity of the trust model to the collusion and badmouthing attacks. This set of experiments demonstrates the benefit of trust model we proposed, peers compare the trustworthiness of peers and choose the peer with the highest trust value to interact with. A transaction is considered successful if both of the participating peers are good peers, otherwise is a failure transaction. We define the rate of failure transactions as the ratio of the number of failure transactions over the total number of transactions in the community up to a certain time. A community with a lower transactions failure rate has a higher productivity and a stronger level of security. The experiment proceeds by repeatedly having randomly selected good peers

initiating transactions. For each request, a certain percentage of peers respond. The response percentage is set to 5% in the experiments. The initiating peer selects the peer with highest trust value to perform the transaction. To measure the effectiveness of aggregating feedback information, we simulate an unhealthy environment and set the percentage of the bad raters 80% and the percentage of the bad SP 25% which are all fixed bad SPs. In the first experiment, bad raters are fixed dishonest peers, and in the second experiment bad raters are dynamic dishonest peers, which tell the truth over a sustained period of time and only then start lying.



(a) Rate of Failure Transaction in Non-collusive and Collusive Setting

(b) Comparison of Rate of Failure Transaction with PeerTrust in Collusive Setting

Fig. 2. Effectiveness in Aggregating Feedback Information

Figure 2(a) shows the rate of failure transactions with respect to the number of time-frame in collusive and non-collusive setting. In the non-collusive setting, every peer provides transaction feedback information independently, so the rate of failure transactions of good peers is gradually reduced. While in the collusive setting, dishonest peers' collusive behaviors hardly disturb honest peers' judgment. It needs more interactions to differentiate good peers from bad peers. However, the system still benefits from our approach of aggregating feedback information significantly and shows robustness against the collusion.

Figure 2(b) shows the comparison of the immunity to the collusive behaviors between PeerTrust PSM approach and our approach in the situation where bad raters are dynamic dishonest peers, which tell the truth over the first two time-frames and only then start lying. From the figure, we conclude that our approach enhanced the veracity of the recommendation credibility and reduced the rate of fake services sooner. Because feedback credibility is updated in an up-to-date manner, trust model we proposed aggregates feedback information efficiently and works better than PeerTrust PSM in this situation.

5 Conclusions and Future Work

We present a time-frame based dynamic P2P trust model. We incorporate time dimension using time-frame, which captures direct experiences and recommendations' time-sensitivity, we also introduce four trust parameters in computing trustworthiness of peers, namely, trust construction factor, trust destruction factor, supervision period factor and feedback credibility. Together, these parameters are adjusted in time to reflect the dynamics of the trust environment using feedback control mechanism, thus, the trust evaluation has better adaptability to the dynamics of trust. Theoretical analysis and simulation show that, the trust model we proposed has advantages in modeling dynamic trust relationship and aggregating feedback information over the existing trust metrics. It is highly effective in countering malicious peers regarding strategic altering behavior and dishonest feedbacks of malicious peers.

As a next step, we will enhance this trust model with the mechanism for providing the incentives for reporting truthful feedback and incorporating context dimension. We will also be evaluating our trust model as applied to a peer-to-peer network.

References

1. Andy Oram: Peer to Peer: Harnessing the power of disruptive technologies. ISBN 0-596-00110-X, 2001
2. Karl Aberer, Zoran Despotovic. Managing Trust in a Peer-2-Peer Information System. In the Proceedings of Intl. Conf. on Information and Knowledge Management, 2001
3. Sepandar D. Kamwar, Mario T. Schlosser, Hector Garcia-Molina. The Eigentrust Algorithm for Reputation Management in P2P Networks. In the Proceedings of the twelfth international conference on World Wide Web, Budapest, Hungary, 2003
4. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Managing and sharing servents' reputations in p2p systems. IEEE Transactions on Data and Knowledge Engineering, 15(4):840-854, Jul./Aug. 2003
5. L. Xiong and L. Liu. PeerTrust: Supporting reputation-based trust in peer-to-peer communities. IEEE Transactions on Data and Knowledge Engineering, Special Issue on Peer-to-Peer Based Data Management, 16(7):843-857, July 2004
6. S. K. Lam and J. Riedl: Shilling recommender systems for fun and profit. In Proceedings of the 13th World Wide Web Conference, 2004
7. Mudhakar Srivatsa, Li Xiong, Ling Liu: TrustGuard: countering vulnerabilities in reputation management for decentralized overlay networks. WWW 2005: 422-431
8. Marsh Stephen: Formalising trust as a computational concept. PhD Thesis. Scotland, University of Stirling, 1994
9. A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In 33rd Hawaii International Conference on System Sciences, 2000
10. B. Yu, M. P. Singh, and K. Sycara. Developing trust in large-scale peer-to-peer systems. Proceedings of First IEEE Symposium on Multi-Agent Security and Survivability, 2004
11. Leitao Guo, Shoubao Yang, Jing Wang, and Jinyang Zhou: Trust Model Based on Similarity Measure of Vectors in P2P Networks. GCC 2005, LNCS 3795, pp. 836 – 847, 2005
12. DOU Wen, WANG Huai-Min, JIA Yan, ZOU Peng: A Recommendation-Based Peer-to-Peer Trust Model. 2004 Vol.15 No.04 Journal of Software

13. D. W. Manchala. E-commerce trust metrics and models. Internet Computing, 4(2):36–44, 2000
14. Claudiu Duma, Nahid Shahmehri, Germano Caronni: Dynamic Trust Metrics for Peer-to-Peer Systems. Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA'05)
15. Huang Chenlin: The Study of Dynamic Trust Relationship Modeling and Managing. PhD Thesis, PhD Thesis. China, University of Nudt, 2005
16. D. Cvrˇcek. Dynamics of reputation. In 9th Nordic Workshop on Secure IT-systems (Nordsec'04), pages 1–14, November 2004
17. Zhou Jun Feng, Tang Xian, Guo Jing Feng: An Optimized Collaborative Filtering Recommendation Algorithm. 2004, Vol.41, No.10, Journal of Computer Research and Development (in Chinese)
18. A. Singh and L. Liu. TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems. In IEEE International Conference on Peer-to-Peer Computing, pages 142–149, 2003
19. Thomas Beth, Malte Borcherding, Birgit Klein: Valuation of Trust in Open Networks, Proc. 3rd European Symposium on Research in Computer Security -- ESORICS '94
20. R. S. Sutton and A. G. Barto. Reinforcement Learning: An Introduction. MIT Press, Cambridge, Massachusetts, 1998

Spatial Context in Role-Based Access Control

Hong Zhang[1,2], Yeping He[1], and Zhiguo Shi[1,2]

[1] Institute of Software, Chinese Academy of Sciences, Beijing 100080, PRC
[2] Graduate School of the Chinese Academy of Sciences, Beijing 100049, PRC
{hong, yphe, szg}@ercist.iscas.ac.cn

Abstract. Controlling access to resources in location-based services and mobile applications require the definition of spatially aware access control systems. However, traditional RBAC model does not specify these requirements. In this paper, we present an extension of the RBAC model to deal with spatial and location-based information, which called LRBAC. In LRBAC, the final permission set of a user depends on the physical location in which a user is situated. The ability to specify the spatial boundary of the role allows LRBAC to be flexible and express a variety of access policies that can provide tight and just-in-time role activation. Besides a real position obtained from a specific mobile terminal, users are also assigned a logical location domain that is application dependent. Then, we extend LRBAC to deal with hierarchies and present how complex spatial role hierarchies in the location-dependent case can be generated by applying Cartesian products as an arithmetic operation over role hierarchies and logical location domain hierarchies.

1 Introduction

The rapid development in the field of wireless and mobile networking fostered a new generation of devices suitable for being used as sensors by location technologies able to compute the relative position and movement of users in their working environment. The widespread deployment of location-based services and mobile applications has resulted in a strong demand for location aware access control systems. Models with location-based access control are becoming increasingly necessary in these applications, and are adequately receiving considerable popularity in the research and industrial community alike.

A user carrying a mobile terminal is on the move (constantly changing position) and may request access to certain resources from several sites. These sites may have different level of trustworthiness or they may be different administrative domains such that the resources available to a user should differ from one site to another. Thus, in organizations where access to sensitive resources are limited to a specific location, wireless location based services require means for obtaining the position of the requesting user in order to mediate the authorization request. As another example, consider a large scale company. Suppose that the set of users includes different categories of employees, such as staff, payroll manager and sales manager. Each category of employees needs to access different information resources. For example staff should be allowed to consult public

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 166–178, 2006.
© Springer-Verlag Berlin Heidelberg 2006

books or telephone directory inside the company buildings. Payroll manager may be granted the right to read all staff's payrolls. But he must read those payrolls in his own office, and not anywhere else in the company. It thus reduces the possibility of curious employees being able to read their colleague's payrolls over the payroll manager's shoulder. Sales managers are allowed to consult certain business contracts only in a security area.

To deal with the requirements listed above, an access control model with location information is needed. Roles in these policies are no longer static but dynamic, depending on the physical location at the time of request. Traditional RBAC [1][12] offers an elegant solution to the problem of managing complex access control rule sets. Although RBAC is very useful for modeling access control in a variety of applications, its role are inherently subject centric [1][2]. It cannot be used to capture security-relevant context from the environment, which could have an impact on access decisions. We extend the RBAC model to allow spatial entities to model objects, user locations, and geographically bounded roles. The roles are automatically (de)activated by the position of the user. Besides a physical position, obtained from a given mobile terminal, users are also assigned a logical location domain which is integrated into the spatial role hierarchy.The evaluation of policies takes into account both role of a requester and the location domain in which a user is situated. The permissions assigned to users depend on their positions; objects to which permissions must be granted are located in that space. In conclusion, the main contributions of the paper are threefold: the proposal of a formal model for dealing with spatial context of an access control system; the introduction of the concept of spatial role and effective role; the complex spatial role hierarchies in the location-dependent case can be generated by applying cross product over role hierarchies and logical location domain hierarchies.

The remainder of this paper is organized as follows. In section 2, we briefly introduce a variety of technologies for location detecting and investigate how to describe a location. The LRBAC model is presented formally in section 3. In section 4, we discuss hierarchies in LRBAC. We discuss related work in section 5. Section 6 concludes the paper and outlines future work.

2 Preliminaries

In order to make RBAC spatially aware, we need to first investigate how to get physical locations and how to describe a location. In this section we categorize location detection approaches and location description approaches, then introduce some basic definitions, such as logical location domain and location mapping function.

2.1 Location Detection

For the system to be able to make authorization decision based on the spatial dimension in which the user is situated, the mediator must be able to obtain the location of the mobile terminal where the access request was made from.

There exists several location detecting technologies with various granularity for both indoor and outdoor position estimation of mobile terminals. Geospatial coordinates are usually acquired by using GPS receivers. GPS works properly only for outdoor determination of the position of a mobile terminal. For indoor location tracking of mobile terminals one may store civil address information or location attributes of a room in a Bluetooth device or palmtops with Wi-Fi cards. When a user hold a device with Bluetooth support enters the room, it can get the location information of the room through Bluetooth beacons. The type of location estimation technique used depends on the requirement of accuracy to the mobile terminal's position, which is required by the system in the authorization process. For example, a user requesting access to a secure service limited to a specific room in a building may require fine granularity in order to ensure that the user does not try to access the service from the room next door.

Location is a sensitive piece of information;releasing it to random entities might pose security and privacy risks. To address these issues, we refer to [18][19].

2.2 Location Description

Location in the context-awareness application needs the modeling of physical environment and representation of locations. Numerous location models have been proposed in different domains, and can be categorized into two classes [17]:

- Hierarchical (topological, descriptive or symbolic, such as room).
- Cartesian (coordinate, metric or geometric, such as GPS).

Since the hierarchical location model is good for representation of spatial relationships and it has the virtue of human readability and self-descriptive, we adopt it for representing locations. It decomposes the physical environment to different levels of spaces. For example, the *CompanyA* is decomposed into several sub-spaces: *Building No.1*, *Building No.2*, *Building No.3*, *Building No.4*, etc. Each of these buildings is divided into smaller composing sub-spaces, until we reach enough precision. Such a hierarchy is called a *space tree*, in which each node corresponds to an actual space in the physical environment. The parent-child link in the tree implies super/sub space relationships between two spaces. Figure 1 illustrates part of the space tree for *CompanyA*. It is up to the location service designer to decide how to decompose the physical environment.

In LRBAC, we use *RLOC* to represent the set of real locations. We assume that areas defined in *RLOC* cover the whole responsibility domain of LRBAC. The *RLOC* can divided into sub domains, called *physical location domains* denoted as $\rho_i, i = 1, \cdots, k$, which reflect the ability of the underlying architecture to uniquely map user location into specific rooms, e.g., room 1025. We assume that underlying infrastructure is unable to distinguish different locations inside ρ_i for any $i = 1, \cdots, k$. However, using physical location domains in LRBAC can be unpractical because physical location domains represent infrastructure of location detection system but we need the structure location domains reflecting organizational infrastructure. Therefore, we introduce logical location domains

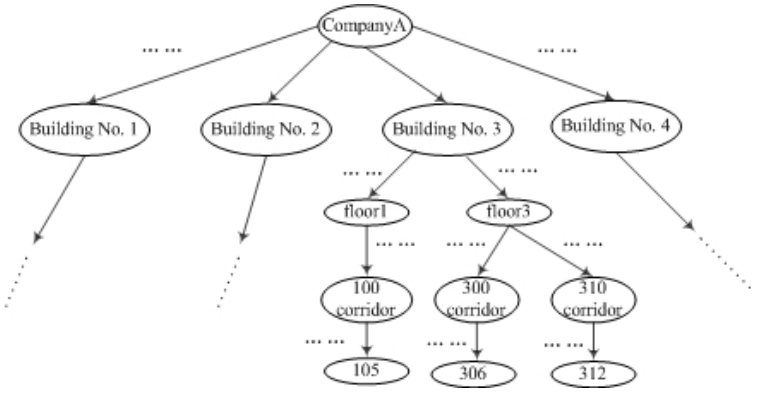


Fig. 1. Hierarchical Space Tree

that express organizational location infrastructure and organizational security policy. For example, within a company we can define logical location domains representing locations such as departments, public area and even individual offices.

Definition 1. *(Logical Location Domain) The logical location domain defines the boundaries of the logical space in which the role can be assumed by the user. We denote with LLOC as the set of all logical location domains.*

In general, physical and logical location domains are highly correlated. Logical location domains can be defined as composition of physical domains. For example, allocation of a department in a company can be described by location expression $DepDom = [\rho_1, \rho_2]$ as area covered by physical location domains ρ_1 and ρ_2. Logical location domains can be computed for any real locations by using specific location mapping function. And the mapping function is application-dependent.

Definition 2. *(Location Mapping Function) Let RLOC be set of real locations, LDOM be set of logical location domains. The location mapping function MapLoc: RLOC → LDOM, given a real location rp, returns a logical location domain.*

Consider the scenario previously discussed in Section 1, we may partition logical location domain into *SADom*, *SMDom*, *PMDom*, *OtherDom*, and they represent security areas in which business contracts are placed, sales managers' offices, payroll managers' offices and the other areas in the company, respectively. Thus, we can define domain for a company $CompDom = SADom + SMDom + PMDom + OtherDom$. The example demonstrates the idea of using location expressions to define new domains. Generally, the same position can belong to different logical location domains. Since logical location domains can be seen as sets, we define new location domains by using domain operations that are similar to operations used in set theory, i.e., union, intersection, difference and complementation, etc.

3 The LRBAC Model

The LRBAC model only uses logical location domain as a context parameter and logical location domain has semantic and geometric meaning. To specify the spatial boundary of the role, we introduce the concept of spatial role. In order to determine the final permissions set of a user in sessions based on user positions, the concept of effective roles has also been introduced. Using these two central concepts, we have proposed the formal model for dealing with spatial context of an access control system. In this section we will present the formal definition of the LRBAC model. After that we discuss the concept of effective role.

3.1 Formal Description of LRBAC

Based on the formalization of the RBAC model in [1][12], we present a precise description of an access control model that includes spatial roles. *Spatial role* combines the roles with logical location domain and it indicates the spatially bounded role.

Definition 3. *(Spatial Role) A spatial role is a pair (r, ldom), where r is the role name and ldom the logical location domain of the role. We denote with SR the set of spatial roles.*

The logical location domain defines the boundaries of the space in which the role can be assumed by the user. Users are assigned spatial roles. Notice that the same role can be associated with different location domains.

An example of spatial role is the pair (*Payroll Manager*, *PMDom*) in which: Sales Manager is the name of the role; *PMDom* the role logical location domain. Through using location mapping function it follows that it is always possible to determine whether the current logical location domain of a user is contained in a specified location domain and thus which roles in the session are effective. Next definition provides the formal semantic of the LRBAC.

Definition 4. *LRBAC has the following components:*

- *U, SR, OP, O, S, RLOC, LDOM stand for users, spatial roles, operations, objects, sessions, real locations and logical location domains, respectively.*
- $PRMS = 2^{OP \times O}$*, is the set of permissions.*
- PA*: $PRMS \times SR$, is a many-to-many mapping permission to spatial role assignment relation.*
- *AssignedPrms:* $SR \rightarrow 2^{PRMS}$*, the mapping of spatial roles onto sets of permissions. Given a spatial role sr,* $AssignedPrms(sr) = \{p \in PRMS | (p, sr) \in PA\}$.
- $UA \subseteq U \times SR$*, a many-to-many user to spatial role assignment relation.*
- *AssignedSession:* $U \rightarrow 2^S$*, assigns a user onto a set of sessions.*
- *AssignedUser:* $SR \rightarrow 2^U$*, the mapping of spatial role onto sets of users. Given a spatial role* $(r, ldom) \in SR$*,* $AssignedUser((r, ldom)) = \{u \in U | (u, (r, ldom)) \in UA\}$.

- *SessionUser:* $S \rightarrow U$*, is a function mapping each session* s *to the single user* $SessionUser(s)$ *that is constant for the session's lifetime.*
- *SessionRoles:* $S \rightarrow 2^{SR}$*, is a function mapping each session* s *to a set of spatial roles* $SessionRoles(s) \subseteq \{(r, ldom) \in SR | (SessionUser(s), (r, ldom)) \in UA\}$.

3.2 Sessions and Roles

$SessionRoles(s)$ corresponds to the roles that can be potentially activated in session s. In general, if a user is assigned to several roles, it is up to him to decide which $SessionRoles(s)$ he is authorized to activate. In our work, the roles which integrated into spatial context information are dynamic in nature and the user does not select the role to be activated directly. However, depending on the location in which a user is situated during the session, only a subset of such roles is effective and permission granted. The roles are automatically (de)activated by the environment. To determine effective session roles, system has to assess the containment between current logical location domain and a specified logical location domain. Therefore, we introduce the concept of effective role to determine the final permissions set of a user.

Definition 5. *(Effective Roles) Effective session roles are defined as the function ESR:* $S \times RLOC \rightarrow 2^{SR}$ *such that* $ESR(s, rloc) = \{(r, lext) \in SR | (r, lext) \in SessionRoles(s) \wedge ldom = MapLoc(rloc) \wedge Contains(lext, ldom) = True\}$*, where Contains(lext, ldom) relationship between lext and ldom holds when ldom is contained in lext.*

Follows from Definition 5, we get the following theorem and the proof of the theorem is trivial.

Theorem 1. $\forall s \in S, \forall rloc \in RLOC. ESR(s, rloc) \subseteq SessionRoles(s)$

Roles to be take effect are selected automatically according to the real location of the user. Effective roles are the basis for determining whether to grant or reject an access request. An access request is a tuple (s, $rloc$, op, o) stating that the user of session s in real location $rloc$ attempts to perform operation op on object o, hence (s, $rloc$, op, o) $\in S \times RLOC \times OP \times O$. The authorization mechanism intercepts this attempt and only allows it to proceed provided that permission (op, o) belongs to the set of permissions assigned to the roles that are effective in s when the session user is in real location $rloc$.

Definition 6. *(Authorization Function) Given an access request* $ar = (s, rloc, op, o) \in S \times RLOC \times OP \times O$*, ar can be allowed at real location rloc if* $(op, o) \in \bigcup_{sr \in ESR(s, rloc)} AssignedPrms(sr)$.

4 Hierarchies in LRBAC

The concept of hierarchy is important in RBAC1 [1] and it reflects the structure of the organization and the respective responsibilities of the roles. Hierarchies

of user roles will be used for permission inheritance. Hierarchical LRBAC, i.e. HLRBAC, adds to LRBAC the support to model hierarchies.The HLRBAC integrates the spatial context parameter into the role hierarchy and does not store them in a separate structure which provides a unified framework and is convenient to implement in real applications.

Before modeling hierarchical LRBAC, we introduce a basic role hierarchy for a location-aware application which is introduced in Section 1. For this, Section 4.1 introduces a sample role hierarchy for this large scale company. After that, we give a second hierarchy, i.e. logical location domain hierarchy. To show how complex role hierarchies in the location-based application can be generated from basic role hierarchies, we will outline the properties of partially ordered set required for our purposes in Section 4.3. For location-dependent access control, we will create a spatial role hierarchy by utilizing the Cartesian product of partial orders in Section 4.4. Section 4.5 gives the formal definition of hierarchical LRBAC.

4.1 Basic Role Hierarchy

Here, we will provide a basic role hierarchy containing all the roles of our sample scenario. Furthermore, we will outline sample permissions of these roles.

Our scenario concerning about a large scale company. There are three roles in the company: Staff, Sales Manager, Payroll Manager.

- Role *Staff*: This role stands for staff members. The users assigned to this role may be allowed to access resources that reserved for public use in the company, such as consult electronic books and telephone directory.
- Role *SM* (Sales Manager): This role is assigned to the users that are responsible for sales tasks within the company. The sales manager needs to read the product prices and business contracts so that he may manage contract negotiations with client executive manager according to product prices.
- Role *PM* (Payroll Manager): This role is assigned to payroll manager of the company. The payroll manager is allowed to read all staff's payrolls. He is also determines how much each staff's salary should be.

Figure 2(a) depicts the partial order of the roles introduced above. Role *Staff* is the most junior role. The *SM* and *PM* are two most senior roles and their responsibilities are different.

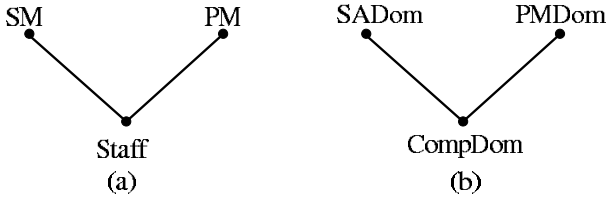


Fig. 2. Hierarchies for (a)basic roles RH in a company and (b)sample logical location domain CH

4.2 Logical Location Domain Hierarchy

In the Section 1, we have mentioned that access control system use spatial context for their access control decisions. Consequently, in location-dependent access control systems, where there are various logical location domains, there should be different levels of permissions. If a logical location domain of a user corresponds to a specific physical location implies different permissions, then these logical location domains should influence the composition of the corresponding roles in role engineering.

We illustrate this idea by means of a sample hierarchy for protection levels, i.e., the hierarchy of logical location domain. Considering the semantics of several logical location domains which introduced in Section 2.2, we depict the hierarchy in Figure 2(b). These logical location domains reflect organizational security policy for different users. If a user A is authorized to access resources by restricting he is situated in the public area, then user A will be assigned to a logical location domain $CompDom \in CH$. The logical location domain *CompDom* is authorized for a specific permission set $P_{CompDom} \in PRMS$. A user B which decided to specify the boundaries of the space is within a security area will be assigned to a logical location domain $SADom \in CH$. The logical location domain *SADom* is authorized for a specific permission set $P_{SADom} \in PRMS$. Analogously, if there is a user C who willing to define the extent of the space is in the payroll manager's office, then user C will be assigned to a logical location domain $PMDom \in CH$. The permission set P_{PMDom} is determined by the same way.

Change logical location domain like *CompDom*, *SADom* and *PMDom* can be done easily in the access control policy. In practice, the basic role hierarchy and logical location domain hierarchy may be more complex than shown in this example. There are more role and spatial constraints necessary in order to express fine-granularity access control policy in real-world organizations. We just use this example to get the idea on how to deal with the problem of different security levels of location domain in access control and how to construct spatial role hierarchies.

4.3 Arithmetic over Partially Ordered Sets

We will outline the properties of partially ordered sets required for our purposes. Since the theory of partially ordered sets is described in detail in the literature, we only describe the aspects that are required to model spatially context-dependent role hierarchy. For more information, we refer to [9][10].

Definition 7. *(Partially Ordered Set) A partially ordered set is a structure* $P =< P, \leq >$ *such that* P *is a set and* $\leq$ *is a binary relation on* P *that is (i) reflexive:* $x \leq x$*, (ii) transitive:* $x \leq y, y \leq z \Longrightarrow x \leq z$*, (iii) antisymmetric:* $x \leq y, y \leq x \Longrightarrow x = y$*.*

Without introducing spatial context information, we can specify role hierarchies for RBAC with the definition above. We suggest now to model the logical location domain itself as a partially ordered set and then apply a specific arithmetic

over partially ordered sets to compose a new hierarchy, namely, the spatial role hierarchy. With the help of Cartesian product, we create the spatial role hierarchy from the basic role hierarchy and the logical location domain hierarchy as specified in Section 4.2.

Definition 8. *(Cartesian Product of Partially Ordered Sets) Let* $P_1, \cdots, P_n$ *be ordered sets. The Cartisian product* $P_1 \times \cdots \times P_n$ *can be made into an ordered set by imposing the coordinate-wise order defined by* $(x_1, \cdots, x_n) \leq (y_1, \cdots, y_n) \leftrightarrow (\forall i)\ x_i \leq y_i$ *in* P_i

4.4 Spatial Role Hierarchy

In Section 4.2, we provide two sample hierarchies for roles and logical location domains in a company. In this section, we will present how complex spatial role hierarchies in the spatial context-dependent case can be generated. For that, we assume the two hierarchies to be independent. Then, we can create a new role hierarchy by utilizing the Cartesian product of two partial orders. The resulting role hierarchy integrates the spatial context parameter into the role hierarchy for the scenario introduced in Section 1. Thus, it provides location-aware access control for the company, where logical location domain is considered a spatial context for the access control decision.

We present this new spatial role hierarchy in Figure 3. It includes all introduced roles, which are the result of the Cartesian product of *RH* and *CH*, and depicts the inheritance relations among these roles which will be described in detail in Section 4.5. Spatial role (*Staff*, *CompDom*) is for all employees of the company whose logical location domain is the company buildings, i.e., *CompDom*. A user who assume role (*Staff*, *CompDom*) is permitted to consult public books and telephone directory. Spatial role (*SM*, *SADom*) is for sales managers of the company whose logical location domain is a security area where business contracts are placed and (*SM*, *PMDom*) is for one sales manager whose current position is in payroll manager's office. Obviously, spatial role (*SM*, *SADom*) has the permission to read product price and relevant business contracts, but spatial role (*SM*, *PMDom*) has neither the permission to get contracts information nor

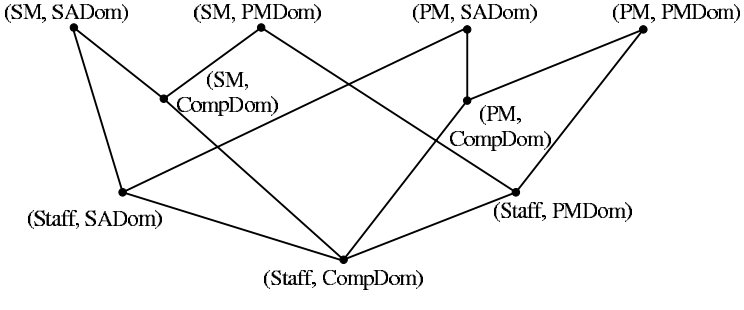


Fig. 3. Spatial role hierarchy $SRH = RH \times CH$

permission to read payroll. Similarly, spatial role (*PM*, *PMDom*) has the permission to read all staff's payrolls and he also has the permission to determine staff's final salary according to some regulations of the company, whereas role (*PM*, *SADom*) has not the same permission because its logical location domain is *SADom*. Furthermore, spatial role (*PM*, *SADom*) hasn't the permission to read contracts because it isn't consistent with security policy of the company, i.e., this operation of read contracts can only be carried out by sales managers. Sales manager's permission is effective when his current location is in *SADom*, whereas payroll manager's permission is effective when his current location domain is in *PMDom*.

4.5 Hierarchical LRBAC

In this section we give the formal definition of hierarchical LRBAC (HLRBAC). According to [1][12], the hierarchical level can be defined by introducing a partial order between roles such that $r_i \leq r_j$ means that: (i) r_j inherits all permissions assigned to r_i; (ii) users who have been assigned r_j have also assigned r_i.

Moreover, since in LRBAC we introduce the concept of effective role, we also assume that (iii) if r_j is effective, and thus the user can play that role in a session s, also r_i results to be effective in s.

There are two cases we should consider when discuss inheritance relations among spatial roles. First, basic roles are different and logical location domains are the same, such as (*Staff*, *CompDom*) and (*PM*, *CompDom*). A payroll manager with logical location domain *CompDom* inherits all the permissions of the junior role Staff with the same logical location domain.

Another case to be considered is when the basic roles are the same while role logical location domains are different. Consider two spatial roles (*PM*, *CompDom*) and (*PM*, *SADom*) with *SADom* is contained within *CompDom*. A payroll manager with logical location domain *SADom* necessarily inherits all the permissions of the payroll manager with larger location domain.

To summarize, HLRBAC can be formally defined as follows.

Definition 9. *(Hierarchical LRBAC) HLRBAC is defined from LRBAC by introducing a partial order between spatial roles.*

- $SRH \subseteq SR \times SR$, *a parital order over SR,* $(r_1, ldom_1) \leq (r_2, ldom_2)$ *holds if* $r_1 \leq r_2$ *and* $ldom_1 \supseteq ldom_2$.
- *AuthorizedPrms:* $SR \rightarrow 2^{PRMS}$ *such that, given a spatial role sr, AuthorizedPrms(sr) returns all permissions assigned to sr and to all its ancestors, i.e.,* $AuthorizedPrms(sr) = \{p \in PRMS | sr' \leq sr \wedge p \in AssignedPrms(sr')\}$.
- *AuthorizedUser:* $SR \rightarrow 2^{U}$ *such that, given a spatial role sr, AuthorizedUser(sr) returns all users assigned to sr and to all its descendants, i.e.,* $AuthorizedUser(sr) = \{u \in U | sr \leq sr' \wedge (u, sr') \in UA\}$.

From the previous definition it follows that the ordering between spatial roles corresponds to the ordering of position granularities: as the role becomes more specific while the logical location domain gets smaller. That is, the more senior roles are those operating on smaller regions.

Since we adopt the concept of effective role, there is a property about effective roles. Assume that a spatial role $sr_2 = (r, ldom_2)$ is effective in a session s and a real physical location $rloc$ and that $sr_1 = (r, ldom_1)$ and $sr_1 \leq sr_2$. From the definition of partial order of spatial role, we infer a fact that the logical location domain of sr_2 is contained in the logical location domain of sr_1. This means that if sr_2 is effective then sr_1 is also effective. In our example, this means that (*PM*, *CompDom*) is effective when (*PM*, *PMDom*) is effective in a certain location.

To summarize, the properties of spatial role hierarchies are given as follows.

Theorem 2. *Let* $s \in S, rloc \in RLOC, sr_1 \in SR, sr_2 \in SR$. *Suppose that* $sr_1 \leq sr_2$. *The following properties hold:*

- $AuthorizedPerms(sr_1) \subseteq AuthorizedPerms(sr_2)$;
- $AuthorizedUser(sr_2) \subseteq AuthorizedUser(sr_1)$;
- $sr_2 \in ESR(s, rloc) \longrightarrow sr_1 \in ESR(s, rloc)$.

5 Related Work

A number of recent research contributions deals with different approaches to consider context information in access control decisions [2][5][6][7][8][11][13][14]. These contributions range from abstract high-level models to case studies and concrete (implemented) software systems. The type of context information that is used for access control depends on the situation, e.g., for workflow and collaboration tasks in [7][11], where context refers to the state of the workflow or task. In this section, we only consider related work that aims to consider location context for systems.

Currently, several access control systems with spatial and nonspatial context-aware extensions have recently been proposed [2][3][6][14]. Even though some preliminary proposals have been reported adding contextual information, such as spatial and temporal information to access control mechanisms, such approaches are simplistic and do not formalize the key components of the systems to give a sharp and precise definition. Most such models do not provide any means for expressing hierarchies for permission inheritance. In these works, they store spatial context in a separate structure and the contexts are considered as an additional step in each access control decision. In [15], an extension of RBAC is proposed, the role is automatically activated when the user is in a given position, the position itself does not have any semantic meaning but simply a geometric value and the model is targeted to wireless network applications and does not introduce the concept of spatial role to deal with geographically bounded roles. Moreover, spatial contexts are implemented as condition functions and spatial contexts are not integrated into the role hierarchy. Bertino et al. [4] has recently been proposed a spatially-aware RBAC model called GEO-RBAC. The GEO-RBAC a common spatial data model is adopted in order to provide a uniform and standard based representation of locational aspects. GEO-RBAC relies on the OGC [16] spatial model to model spatial objects, user positions, and roles, namely, locations are defined in geometric ways, whereas this spatial model hides hierarchical relationships and it needs considerable extra specification to enable deduction of

spatial relationships. Instead, we describes logical location domain according to the organizational security policy, not fully in geometric ways. In GEO-RBAC, there are two different hierarchies are defined which will result in considerable expense to compute inherited permission. And GEO-RBAC is to secure applications in the commercial spatial DBMS and GIS (Geographical Information Systems). If adopt GEO-RBAC to deal with location-aware applications in the large scale company which introduced in Section 1, the implementation will be unnecessarily complex and using GPS based equipment is expensive.

6 Conclusion and Future Work

In this paper we have presented LRBAC model, an extended RBAC model which supporting location-aware applications. LRBAC relies on the hierarchical location model to model spatial objects, making the approach is good for representation of spatial relationships and human readability. The main innovative feature of the model is spatial role hierarchies that are generated from basic role hierarchy and logical location domain hierarchy by applying Cartesian product as a special arithmetic operation over them. Location domains are integrated into the role hierarchy. By introducing the concept of spatial role, the spatial boundary of the role can be customized and introducing the concept of effective role to determine the roles are effective in sessions based on user current position. Moreover, the logical location domain has certain semantic meaning.

In future work, we plan to extend the LRBAC to support constraints, including spatial separation of duty constraints which is able to deal with conflicting logical location domains, location-based cardinality constraints and location-based temporal constraints that can address time-dependency of spatial information. We also plan to consider more spatial relationships that may exist between the spatial elements in space, such as *Touch*, *In*, *Cross*, *Overlap*, *Disjoint* etc in [20]. In this paper we have assumed that the location service always returns location information. We plan to extend our model to the consideration of policies constraining to whom and how location information is to be provided.

References

1. R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role base access control models. In IEEE Computer, 29(2): 38-47, 1996
2. M. Covington, W. Long, S. Srinivasan, A.K. Dev, M. Ahamad, and G.D. Abowd. Securing Context-aware Applications Using Environment Roles. In Proc. of the 6th ACM Symposium on Access Control Models and Technologies, Chantilly, Virginia, USA, 2001, pages 10-20.
3. F. Cuppens and A. Mige. Modelling Contexts in the Or-BAC Model. In Proc. of 19th Annual Computer Security Applications Conference. IEEE Computer Society, 2003, pages 416-427.
4. E. Bertino, B. Catania, M.L. Damiani, and P. Perlasca. GEO-RBAC: A Spatially Aware RBAC. In Proc. of the 10th ACM Symposium on Access Control Models and Technologies, 2005, pages 29-37.

5. M. Wilikens, S. Feriti, A. Sanna and M. Masera. A context-related authorization and access control method based on RBAC: A case study from the health care domain. In Proc. of the 7th ACM Symposium on Access Control Models and Technologies, 2002, pages 117-124.
6. R.J. Hulsebosch, A.H.Salden, M.S. Bargh, P.W.G. Ebben and J. Reitsma. Context-Sensitive Access Control. In Proc. of 10th ACM symposium on Access Control Models and Technologies, Sweden, 2005, pages 111-119.
7. R. Thomas. Team-based access control (TMAC): A primitive for applying role-based access controls in collaborative environments. In Proc. of 2nd ACM workshop on Role-based access control,1997, pages 13-19.
8. R. Wolf and M. Context-Dependent Access Control for Web-Based Collaboration Environments with Role-Based Approach. In Proc. of 2nd International Workshop on Mathematical Methods,Models,and Architectures for Computer Network Security, 2003, pages 267-278.
9. B. Davey, H. Priestley. Introduction to Lattices and Order. Cambridge University Press, 2002.
10. B. Jonsson. Arithmetic of ordered sets. In Proc. of the NATO Advanced Study Institute, 1981.
11. D. G. Cholewka, R. H. Botha, and J. H. P. Eloff. A Context Sensitive Access Control Model and Prototype Implementation. In Proc. of the IFIP TC11 15th International Conference on Information Security, Beijing, China (2000),2000, pages 341-350.
12. D. Ferraolo, R. Sandhu, S. Gavrila, R. Kuhn and R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. In Proc. of ACM Transactions on Information and System Security, Vol. 4, 2001, pages 224-274.
13. A. Kumar, N. Karnik, and G. Chafle. Context Sensitivity in Role-based Access Control. In. ACM SIGPOS Operating Systems Review, v. 36 n.3, 2002, pages 53-66.
14. M. Covington, M. Moyer, and M. Ahamad. Generalized Role-based Access Control for Securing Future Applications. In Proc. of 23rd National Information Systems Security Conference. IEEE Computer Society, 2003, pages 416-427.
15. F. Hansen and V. Oleshchuk. Spatial Role-based Access Control Model for Wireless Networks. In Proc. of IEEE Vehicular Technology Confer-ence (VTC), Orlando, USA, 2003.
16. OpenGIS Consortium. OpenGIS Simple Features Specification for SQL. Technical Report OGC 99-049, 1999.
17. C. Jiang and P. Steenkiste. A Hybrid Location Model with a Computable Location Identifier for Ubiquitous Computing. In Proc. of the Fourth International Conference on Ubiquitous Computing (UBICOMP 2002), Goteborg, Sweden, 2002, pages 246-263.
18. U. Hengartner and P. Steenkiste. Implementing Access Control to People Location Information. In Proc. of the 9th ACM Symposium on Access Control Models and Technologies, 2004, pages 11-20.
19. U. Hengartner and P. Steekiste. Protecting Access to People Location Information. In Proc. of the First International Conference on Security in Pervasive Computing, 2003, pages 25-38.
20. E. Clementini, P. di Felice, and P. Van Oosterom. A Small set of Formal Topological Relationships Suitable for End-User Interaction. In Proc. of the 3th International Symposium on Advances in Spatial Databases, 1003, pages 277-295.

An Efficient Scheme for Detecting Malicious Nodes in Mobile Ad Hoc Networks*

Jongoh Choi[1], Si-Ho Cha[2,**], and JooSeok Song[1]

[1] Department of Computer Science, Yonsei University, Seoul, Korea
{jochoi, jssong}@emerald.yonsei.ac.kr
[2] Dept. of Information and Communication Engineering, Sejong University, Seoul, Korea
sihoc@sejong.ac.kr

Abstract. This paper proposes a scheme capable of effectively detecting a malicious node that normally operates during determination of a route over a mobile ad-hoc network (MANET) but modifies or drops data during data transmission or reports wrong information regarding a normal node, using a report message and a report table that list reporter nodes and suspect nodes. In the existing schemes, a malicious node that provides wrong information can be easily identified but cannot be removed from a network. To solve this problem, the proposed scheme determines a suspect node as a malicious node when more than k lists of reporter nodes and suspect nodes are recorded in the report table in case where k malicious nodes are over the network. The proposed scheme is applicable to both DSR and AODV routing.

1 Introduction

Most of researches on the MANETs have focused on a wireless channel access or multi-hop routing based on an assumption that network elements operate in a friendly and cooperative environment. However, since malicious nodes and uncooperative situations may occur in an actual network environment, there is a growing need for a security scheme that guarantees secure communications between mobile nodes. In fact, researches on the security scheme have been conducted. Although there are researches on the security of a wire network, the following matters must be considered in designing a security scheme for the MANET. First, the MANET has an open peer-to-peer construction. Thus, it is possible to devise security for a router that separates the inside and outside of a wire network, but it is difficult to provide security for the MANET since it has no particular router. Second, a wireless channel is shared by a plurality of nodes. Since both an authorized node and a malicious node can access the wireless channel, the network can be easily attacked. Third, network resources over the

* This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment).
** Corresponding author.

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 179–193, 2006.
© Springer-Verlag Berlin Heidelberg 2006

MANET are very limited than over a wire network. In general, a low power mobile node that uses battery power does not have resources and performance enough to perform an encrypting process that requires large processing overhead. Therefore, the low power node is vulnerable to external attacks. Fourth, a change in the mobility of mobile node or the state of a wireless channel changes network topology dynamically and remarkably. Nonetheless, mobile nodes always desire to receive secure communications services from a network regardless of where they are [1].

The MANET may be used as a military communications network in a war field, an urgent rescue communication network when a disaster occurs, or a communications network in a temporary meeting. Accordingly, effects on the network or a society caused when a malicious attacker arbitrarily modifies or damages data transmitted in the network in such an urgent case, is far more serious than them in a wire network. Further, the MANET is a temporary communications network that can be used for a short time or must be used for a predetermined time and installed again. Accordingly, it is required to immediately detect and take measures for an attack against the MANET.

There are basically two approaches to protect the MANET: proactive and reactive. In the proactive approach, a malicious node is detected and excluded from the network so as to determine a routing route with only friendly and cooperative nodes. In the reactive approach, when an attacker compromises the MANET although the route is determined as described above, a malicious node is detected and excluded from the network [1]. This paper will study the reactive approach. The conventional studies of the MANET have been focused on detection of a node that maliciously drops or modifies data. That is, they do not provide a method of identifying a malicious node that makes a false report of a normal node. In contrast, this paper proposes an scheme that not only identifies a malicious node, which drops or modifies packets, using a report table storing previous report lists, but also detect a malicious node that makes a false report of a normal node, thus degrading the performance of a network.

The construction of this paper will briefly be described. Section 2 describes related works regarding identification of a malicious node. Section 3 proposes a method of detecting a malicious node that reports wrong information. Section 4 proves the good performance of the introduced scheme through a simulation using the NS-2 simulator. Section 5 provides the conclusion of this paper.

2 Related Work

2.1 Attacks Against Routing

Attacks against routing include all actions that prevent routing information from being transmitted according to a routing scheme for the MANET. To protect the routing scheme from attacks, the Ariadne for DSR scheme [2] suggests authentication of a routing message exchanged between nodes using a one-way HMAC key chain, thereby preventing a change in a source route during DSR. The SAODV scheme [4] suggests a routing message be divided into two parts:

a part to be changed and the other part to be unchanged during transmission of the routing message. Then, the hop count of the changed part is processed using a hash chain and the hop count of the other part is processed using digital signature, thereby protecting the routing information. The SEAD scheme [5] proposes hop count and a sequence number be made using a hash chain. In addition, much research efforts have been conducted to protect the routing scheme from attacks, but this paper will discuss packet forwarding attacks and security therefor, not routing attacks.

2.2 Attacks Against Packet Forwarding

Attacks against packet forwarding indicate actions of a node that normally operates during determination of route but does not transmit data packets to a next node. The node may be a selfish node that does not transmit data to a next node and transmits only its data to save its resources, or a malicious node that has an intention to degrade the performance of network. In any case, both the selfish and malicious nodes must be detected and controlled to guarantee the performance of network. Throughout this paper, both the selfish and malicious nodes will be referred to as a malicious node. Attacks against packet forwarding include dropping or arbitrarily modifying a data packet to be transmitted, repeatedly forwarding an already transmitted data packet, and transmitting a large amount of insignificant packets within a network to consume network resources, thereby increasing contention for or congestion in a wireless channel. There are various approaches to prevent the above attacks, e.g., the Watchdog and Pathrater scheme [3], Byzantine Fault Detection [6], and management of a selfish node [7]. A detailed description thereof will now be described. The Watchdog and Pathrater scheme is advantageous in that it is possible to detect a malicious node in the route at a packet transfer level, not a link level. In the Byzantine Fault Detection, when the destination node receives data, it transmits an ACK to the source node. If the source node does not receive the ACK within a predetermined length of time, it determines that data loss occurs in the route. In the management of a selfish node scheme, each node transmits data to a next node and stores a copy of the data in its buffer. Upon receipt of a certificate from the next node, the node confirms that the data was properly transmitted to the next node.

2.3 Problems

However, the above approaches have a defect since it is impossible to determine whether a malicious node in the route makes a false report about a normal node to the source node S.

In the Watchdog and Pathrater scheme [3], the node C normally receives a data packet from the malicious node B and transmits it to the destination node D, and the malicious node B overhears the transmission of the data packet. However, the malicious node B reports to source node S that the node C does not transmit the data packet to the destination node D. If the destination node D sends an ACK to the source node S, the source node S will realize that the

node B makes a false report. However, when the malicious node B drops the ACK, the source node S will wrongly determine that the node C is a malicious node, based on the report of the node B.

In the Byzantine Fault Detection [6], the malicious node 3 drops the received data, normally processes a probe message received from the source node S, sends an ACK to the source node S. In this case, the source node S receives the ACK from the node 3 and do not receive an ACK from the node 4. Therefore, the source node S will wrongly determine that the node 4 is a malicious node.

When detecting a malicious node using a certificate, the malicious node B transmits data to the node C, receives a certificate from the node C, and makes a false report to the source node S together with a certificate issued by the node C although the node C transmits the data to the destination node D. In addition, the node B drops an ACK from the destination node D via the node C.

Accordingly, the source node S wrongly determines that the node C is a malicious node without suspecting the node B. That is, the above approaches cannot provide a solution to an attack of falsely reporting a normal node.

3 Proposed Scheme

3.1 Assumptions

It is assumed that nodes are connected via a bi-directional link, and each node operates in a promiscuous mode and thus can overhear transmission of data of a neighbor node. Also, it is assumed that asymmetric encrypting is used to prevent a node's false report, a private key K_i^- of a node i over a network is not known to any node except the node i, and a public key K_i^+ of the node i is known to all nodes over the network. Thus, in order to reports a node k as a malicious node, the node i must encrypt a report message using the private key K_i^- and broadcast the report message in the network (reporter: node i, and suspect: node k). All the nodes receiving the report message can read the report message by decoding it using the public key K_i^+. Since the private key K_i^- is not known, the other node cannot make a false report message.

3.2 Proposed Scheme

The proposed scheme is fairly similar to the Watchdog and Pathrater scheme. Specifically, each node over a network transmits data to a next node, stores a copy of the data in its buffer, and overhears transmission of the data of the next node to confirm whether the next node transmits the data to its neighbor node.

Referring to Fig. 1, a node B transmits data to a node C, stores a copy of the data in its buffer, and overhears transmission of the data of the node C to determine whether the node C transmits the data to a destination node D. If the node B does not overhear the transmission of data of the node C within a predetermined length of time, the node B increases a failure tally for the node C. When the tally is greater than a threshold, the node B determines that the node C made a misbehavior. However, a malicious node's misbehavior is reported to

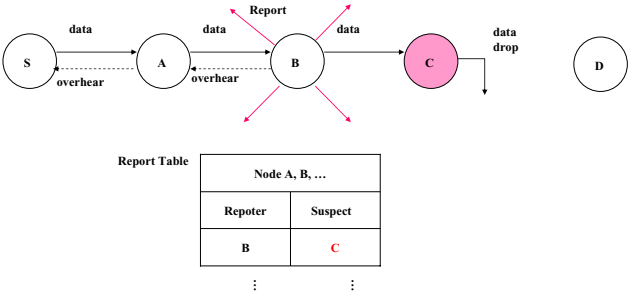


Fig. 1. Proposed Algorithm

a source node S via unicast in the Watchdog and Pathrater scheme, but the misbehavior is reported to all nodes over the network in the proposed scheme, thereby immediately detecting and removing a malicious node. Each of the nodes receiving the report determines whether a reporter and a suspect node listed in the report are recorded in its report table. If the reporter and the suspect are listed in the report table, the node disregards and drops the report. If not so, the node enters the reporter and the suspect node in the report table. In response to the report, the source node S sets up a new route while excluding a route in which the reporter and the suspect node are included as neighbor nodes. When the number of times that a node reports to the source node S is greater than k equivalent to the number of malicious nodes over the network, the node is determined as a malicious node and excluded from the network.

Fig. 2 is a flowchart of the proposed scheme that identifies a malicious node, reports the malicious node to a source node using a report message, and manages a report table based on the report message. We represent several scenarios to prove the good performance of the proposed scheme in a network and explain a method of detecting and excluding a malicious node. The scenarios include cases where a malicious node drops or modifies data, a malicious node submits a false report regarding a normal node, and a certain node submits a false report regarding a normal node.

Case 1: A malicious node drops data.
Referring to Fig. 3, when a malicious node C does not transmit data to a destination node D and drops the data, a preceding node B cannot overhear transmission of data of the node C within a predetermined length of time and thus determines that the node C does not transmit data and drops it. Thus, the node B reports the node C as a malicious node.

Case 2: A malicious node modifies data.
Referring to Fig. 4, a malicious node C arbitrarily modifies the content of or a part (or the entire part) of a header of data received from a node B, and transmits the modified data to a node D. Then, the node B overhears the transmission of the data of the node C and compares the transmitted data with a copy of the data stored in a buffer of the node B. When the comparison

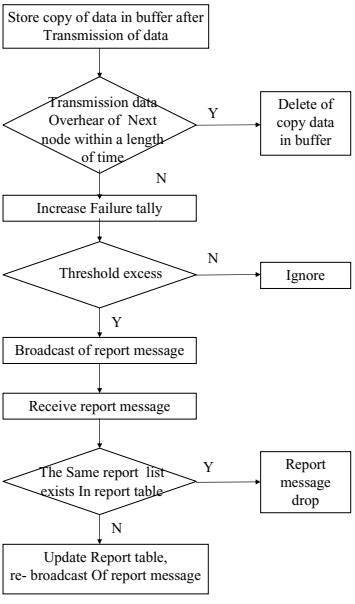


Fig. 2. Flowchart of Proposed Algorithm

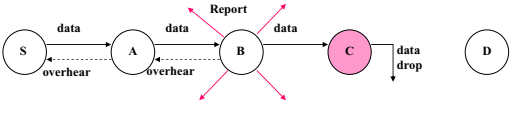


Fig. 3. When Malicious node drops data

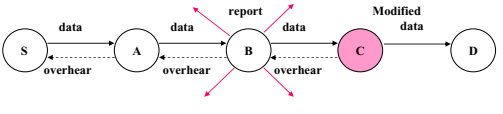


Fig. 4. When Malicious node modifies data

reveals that the data was arbitrarily changed, the node B considers the node C as a malicious node and reports the node C to a source node S. In case 1 or 2, when the node B submits a report regarding the node C, a report list of (a) of Fig 5 is recorded in the report tables of all nodes over the network. When the source node S receives the report and does not receive an ACK from the destination node D, the source node S determines that a malicious node is in the current route and sets up a new route. If the malicious node C is included in the new route or another route from the source node S to the destination node D, the malicious node C will drop or arbitrarily modify data, and thus, other nodes L and K will report the node C as a malicious

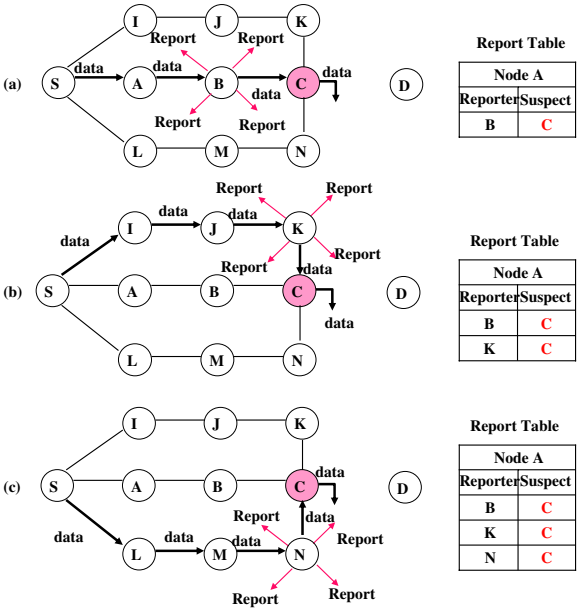


Fig. 5. Measures against case 1, 2

node to the source node S [see (b) and (c) of Fig 5]. In other words, when a malicious node operates normally during determination of a route, it can be included in the determined route. However, when the malicious node does not properly transmit data to a neighbor node, other nodes report the malicious node so that the malicious node is recorded as a suspect node in the report table.

If two malicious nodes are on the network, it is possible to consider the node C as a malicious node and exclude it from further network operations when the node C is recorded as a suspect node at least three times. This is because the number of times that the two malicious nodes can submit a false report regarding a normal node while cooperating with each other is two. Accordingly, when the node C is reported as a malicious node at least three times, this report can be considered as a true report, not a false report.

Case 3: A malicious node disguises itself as another node and submits a false report. Since the proposed scheme uses asymmetric encryption using a private key and a public key, it is possible to prevent a malicious node from disguising itself as a normal node using the identification of the normal node, and submitting a false report. Referring to Fig 6, even if a node B can disguise itself as a node X and submit a false report message R, the node B does not know a private key K_X^- of the node X, and must encrypt the false report message R using its private key K_B^- and broadcast the false report message R.

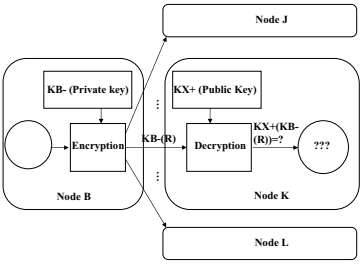


Fig. 6. When Malicious node disguises as another node

Upon receiving the false report message R, each node considers that the node X transmits the false report message R and decodes it using the public key K_X^+ of the node X. However, since the false report message R was not encrypted using the private key K_X, the false report message R cannot be decoded properly, thus causing an error. Therefore, each node can realize that the false report message R is a false report. For this reason, the node B cannot disguise itself as another node and submit a false report.

Case 4: A malicious node submits a false report regarding another node. When a malicious node M submits a false report regarding a node X irrespective of route setup or data transmission, a list of a reporter and a suspect node is entered in a report table of each node, indicated by 1 of Fig 7. If the malicious node M continues submitting a false report at a current or new position, a report list is added to the report table of each node, indicated by 1 and 3 of Fig 7. When the malicious node M is recorded as a reporter in the report table of each node more than k times, the node M is identified as a false reporter, and thus cannot participate in network operations.

Case 5: A malicious node submits a false report regarding a normal node. As shown in Fig 8, when a malicious node B submits a false report regarding a normal node C and drops an ACK transmitted from the node D, a source node S sets up a new route without determining whether the report of the node B is false. Instead, a list of (a) of Fig 8 is added to a report table of each node. When the node B makes a false report at a new or different route

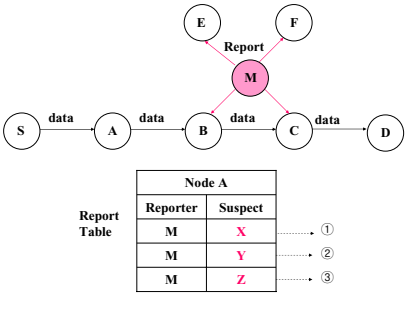


Fig. 7. In case of Temporary Report

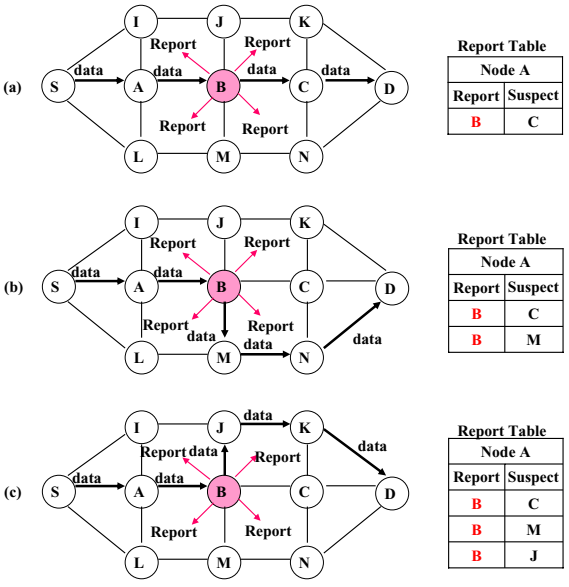


Fig. 8. In case of false report

again, lists of (b) and (c) of Fig 8 are added to the report table of each node. Referring to (a) through (c), only the malicious node B is listed as a reporter but nodes listed as suspect nodes are different from one another, which proves that the node B made false reports. When two malicious nodes are over the network and the node B is recorded as a reporter in the report table three or more times, the node B is considered as a malicious node that submitted false reports. If the node B is a good node and nodes C and H are malicious nodes, the lists (a) and (b) may be created according to true reports, not false reports. Therefore, when the number of reports is larger by at least one than the number of malicious nodes, a suspect node is considered as a malicious node.

3.3 Application of Proposed Scheme to AODV

This chapter will discuss a method of applying the proposed scheme to the DSR protocol and the AODV routing protocol that are two representative on-demand routing protocols for the Ad Hoc Network. It is easy to exclude a node, which is identified as a malicious node, during determination of a route. Referring to (a) of Fig 9, when a node A broadcasts a RREQ message, a malicious node B receives and rebroadcasts the RREQ message. Normal nodes E, C, and F receive the RREQ message from the malicious node B, realize that the node B is malicious node from their report tables, and do not allow transmission of the RREQ message to other nodes in the network, thereby excluding the node B from the route.

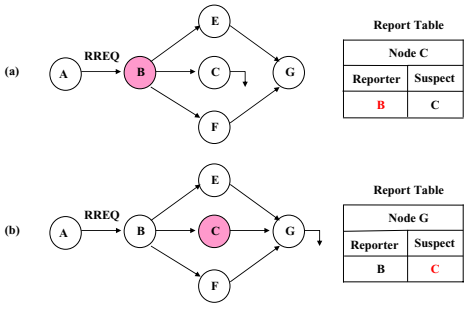


Fig. 9. Application of proposed Algorithm to AODV

There is a case where a pair of a reporter and a suspect is reported once or more in the network but it is difficult to identify a malicious node. Referring to (a) of Fig 9, the nodes B and C are recorded as a reporter and a suspect, respectively, in the report table of each node over the network. If the reporter node B is a malicious node that maliciously submits a false report, the node B broadcasts the RREQ message and the nodes E, C, and D receive the RREQ message. The normal node C drops the RREQ message, since it does not desire to reflect the RREQ message, which describes that the node B is the reporter and the node C is the suspect, in determining a new route. However, the nodes E and F rebroadcast the RREQ message from the node B so that a new route is determined to reflect to determine the route according to the RREQ message. Referring to (b) of Fig 9, since a node C is a malicious node, the node C does not drop the RREQ message that the node B broadcast, and rebroadcasts it so that both the nodes B and C can be included in a new route. This is because the node C predicts even if the malicious node C is included in the new route again and drops data from the node B and the node B reports this fact, the report of the node B would be disregarded in the network. In this case, the AODV routing requires an additional scheme to exclude a malicious node. Specifically, according to the additional scheme, a previous node address field previous_add is added to the RREQ message, thus allowing a node receiving the RREQ message to notice a node transmitting the RREQ message and a previous node preceding the node that transmits the RREQ message. Referring to (b) of Fig 9, the malicious node C receives the RREQ message from the node B and transmits the RREQ message to the node D. Then, the node D notices that the previous node (previous-hop) of the node C is the node B based on the RREQ message and a report table of the node D, and do not transmit the RREQ message to exclude the nodes B and C from the route.

3.4 Application of Proposed Scheme to DSR

The DSR is a type of source routing. In the DSR, each of intermediate nodes in a route can detect all previous nodes of a node that transmits an RREQ message to the intermediate node. If a malicious node or a pair of a reporter and a suspect, which are listed in a report table of the intermediate node, are specified in the

RREQ message, the intermediate node does not transmit the RREQ message to a neighbor node and drops it to exclude the malicious node or the pair of the reporter and the suspect from a new route. In the DSR, a method of excluding a node that is identified as a malicious node from a network is similar to in the AODV routing. Referring to (a) of Fig 9, a malicious node B broadcasts an RREQ message, and nodes C, E, and F receive the RREQ message. However, the nodes C, E, and F notice that the node B is a malicious node from their report tables, and do not transmit the RREQ message to exclude the node B from the network. Next, it is assumed that whether a reporter node or a suspect node is a malicious node has not yet been identified. In this case, similarly in the AODV routing, when a reporter node B is a malicious node, nodes C, E, and F receive a RREQ message from the node B. However, the node C does not broadcast the RREQ message and drop it since it does not desire to reflect the RREQ message that describes the nodes B and C as a reporter and a suspect, respectively, in determining a new route [see (a) of Fig 9]. Instead, the nodes E and F broadcast the RREQ message from the node B so that a new route can be determined based on the RREQ message. If the suspect node C is a malicious node, the AODV routing requires an scheme that adds a previous node address field previous_add into the RREQ message to exclude the node C from a new network. In contrast, the DSR does not require the scheme. This is because the node D receives the RREQ message broadcast again by the node C, reads source routing information from the RREQ message, and does not transmit the RREQ message to prevent the nodes B and C from being included in a new route [see (b) of Fig 9].

4 Analysis and Performance

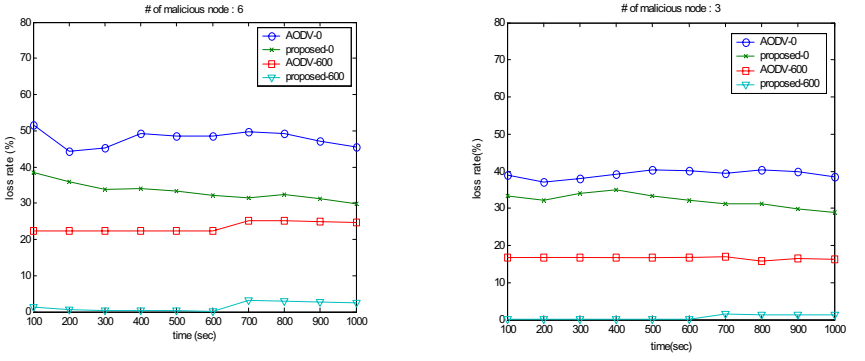
This section proves the good performance of the proposed scheme through a simulation. The simulation was performed using the NS-2 simulator. In the simulation, the proposed scheme was applied to the existing AODV routing protocol since it is easy to apply the proposed scheme to the DSR protocol as described above. Also, in the simulation, the existing AODV routing protocol and the AODV routing protocol that uses the proposed scheme (hereinafter, "the proposed AODV routing) are compared with each other in terms of their average loss rates, average transmission rates, and overheads, on an assumption that a malicious node is over a network. Table 1shows major parameter values used in the simulation.

4.1 Average Loss Rate

(a) of Fig 10 is a graph illustrating average loss rates in the existing AODV routing protocol and the proposed AODV routing protocol when the number of malicious nodes is six and pause times are 0 and 600 sec. The graph reveals that the loss rate in the proposed AODV routing protocol is 10 through 20% less than that in the existing AODV protocol. Also, the longer the length of time of the simulation, the less the loss rate in the proposed AODV routing protocol. That

Table 1. Major parameters

Network size	1000 * 1000 (m)
Number of nodes	60
Number of malicious nodes	3, 6
Simulation time	1000 sec
Pause time	0, 600 sec
Traffic	UDP/CBR



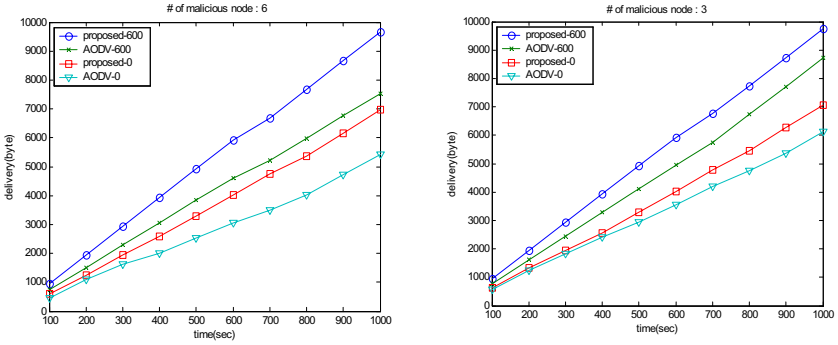
(a) The number of Malicious nodes : 6 (b) The number of malicious nodes : 3

Fig. 10. Average Loss Rates

is, as a predetermined time has passed, the proposed scheme identifies malicious nodes over a network and excludes them from a newly determined route, thereby preventing attacks by the malicious nodes and reducing the loss rate. Further, the more the mobility of malicious node, the higher the loss rate. In other words, it is highly probable that a malicious node would move to be included in a new route. Referring to (b) of Fig 10, the average loss rate when the number of malicious nodes is three, is lower than when the number of malicious nodes is six. That is, the less the number of malicious nodes over the network, the less the probability that the malicious nodes would be included in the route.

4.2 Average Transmission Rate

Fig 11 is a graph illustrating average transmission rates in the existing AODV routing protocol and the proposed AODV routing protocol in the above environment. Referring to Fig 11, since the loss rate in the proposed AODV routing protocol is less than that in the existing AODV routing protocol, the transmission rate in the proposed AODV routing protocol is higher than in the existing AODV routing protocol. Also, a large amount of data can be transmitted when three malicious nodes are over the network, in contrast with when six malicious nodes are over the network. Further, a loss rate is lower and the transmission rate is higher when pause time is 600 sec, i.e., when the mobility of network is small, than when pause time is 0, i.e., when the mobility of network is large.



(a) The number of malicious : 6 (b) The number of malicious : 3

Fig. 11. Average Transmission Rates

That is, when malicious nodes frequently move, they are highly likely to be included in a new route. In this case, the loss rate in the network is increased, thus lowering the transmission rate. The longer the length of time of the simulation, the greater the difference in transmission rates between the existing AODV routing protocol and the proposed AODV routing protocol. This is because malicious nodes over the network are detected and excluded from the network as a predetermined length of time has passed.

4.3 Overhead

Fig 12 is a graph illustrating the transmission gains and overheads in the existing AODV routing protocol and the proposed AODV routing protocol, the transmission gains and overheads being represented in units of bytes. The overhead is a value obtained by subtracting the amount of control packets in the existing AODV routing protocol from that of control packets in the proposed AODV routing protocol. The transmission gain is obtained by subtracting the transmission rate in the existing AODV routing protocol from that in the proposed AODV routing protocol, the result of subtraction being represented in units of bytes. When comparing their overheads in units of packets of a network

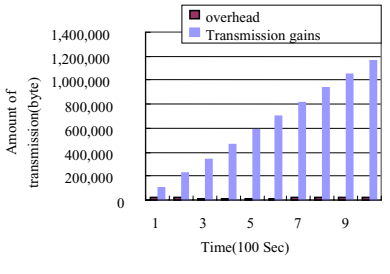


Fig. 12. Transmission gains and overhead

layer, the proposed AODV routing protocol generates more control messages than the existing AODV routing protocol. This is due to broadcasting of a report message in the network when a malicious node is identified in the proposed AODV routing protocol. In general, a control packet is several byte long and a data packet is several hundred byte long. Accordingly, use of the proposed AODV routing protocol obtains more transmission gains with less overhead in considering overall network transmission rates in units of bytes, as illustrated in Fig 12.

5 Conclusions

This paper proposes an scheme that detects and excludes a malicious node that normally operates during determination of a route but abnormally operates during data transmission over the network, using a report message and a report table specifying a pair of a reporter node and a suspect node. In the scheme, a suspect node is determined as a malicious node when k malicious nodes are over the MANET and more than k report lists are recorded in the report table. Accordingly, it is possible to effectively determine whether a node is a malicious node submitting a false report and exclude the node from the network. The proposed scheme is applicable to both the DSR protocol and the AODV routing protocol that are representative on-demand routing protocols for the Ad Hoc Network. Also, this paper proves through a simulation that the AODV routing protocol that uses the proposed scheme is superior to the existing AODV routing protocol in view of their average loss rates and transmission rates. The simulation revealed that the more malicious nodes over the network, the more the mobility of the malicious node, the greater the rate of data loss, and the less the rate of transmission. In particular, as time has passed, the performance of the AODV routing protocol using the proposed scheme becomes still better than the existing AODV routing protocol. This is because the proposed scheme detects malicious nodes over the network and excludes them from the network, thereby reducing the loss of packet, caused by the malicious node. However, the proposed scheme must further be improved to provide more extensive security during determination of a route over the Ad hoc network.

References

1. Hao Yang, Haiyun Luo, Fan Ye, Songwu Lu, and Lixia Zhang: Security in Mobile Ad Hoc Networks: Challenges and Solutions, IEEE Wireless Communications, 2004.
2. Y. Hu, A. Perring, and D. Johnson: Ariadne - A Secure On-Demand Routing Protocol for Ad Hoc Networks, ACM MOBICOM, 2002.
3. S. Marti et al.: Mitigating Routing Misbehavior in Mobile Ad Hoc Networks, ACM MOBICOM, 2000.
4. M. Zapata, and N. Asokan: Securing Ad Hoc Routing Protocols, ACM WiSe, 2002.

5. Y. Hu, D. Johnson, and A. Perring: Sead - Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks", IEEE WMCSA, 2002.
6. B. Awerbuch et al.: An On-Demand Secure Routing Protocol Resilient to Byzantine Failures, ACM WiSe, 2002.
7. Gajin Na et al.: Secure Mechanism to manage selfish nodes in Ad hoc Network, JCCI, 2004.

Mobile RFID Applications and Security Challenges

Divyan M. Konidala and Kwangjo Kim

Information and Communications University (ICU),
International Research Center for Information Security (IRIS)
R504, 103-6, MunjiDong, Daejeon 305732, Republic of Korea
{divyan, kkj}@icu.ac.kr

Abstract. With mobile RFID technology, handheld portable devices like mobile phones and PDAs, also behave as RFID readers and RFID tags. As RFID readers, mobile phones provide an user-friendly approach to quickly and efficiently scan, access and view information about RFID tagged items. As RFID tags, mobile phones can quickly identify themselves in order to communicate with other tagged devices, which provide essential services. At the outset this paper briefly describes Mobile RFID technology and compare it with conventional RFID technology. We pioneer in categorizing Mobile RFID applications into three distinct zones, namely: Location-based Services (LBS) Zone, Enterprise Zone, and Private Zone. We describe application scenarios related to these zones and highlight various security and privacy threats. Finally, we propose a security architecture for LBS zone and describe our future work.

1 Introduction

1.1 RFID Technology

Radio Frequency Identification (RFID) [1] is a means to efficiently and quickly, auto-identify objects, assets, pets, and people. So far, few big companies like Wal-Mart, Proctor & Gamble Co., and Gillette *etc.*, are using RFID technology for real-time tracking of inventory in their supply chain. With the current bar-code technology, each product's bar-code label must be brought before the reader, and labels must be scanned one by one. This leads to laborious, human-error prone, and time consuming inventory check, and also causes customers in a store to wait in long queues at the cashier counter.

Whereas with RFID technology, passive RFID tags are attached to objects/ products and these tags contain tiny, but durable computer chips with very small antennas. Passive tags are powered-up from the interrogation Radio-Frequency (RF) signal of a reader. The tiny computer chips contain an Electronic Product Code (EPC) that uniquely identifies the object to which it is attached to, and the antennas automatically transmit this EPC number without requiring line-of-sight scanning, to RFID readers within a certain RF range. The unique EPC number is like a pointer directing the RFID reader to the right Information

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 194–205, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Server on the EPC Network from where the reader can download additional related data about the product it scanned. Therefore RIFD technology allows quick scanning of products in large bulks (*e.g.,* a whole pallet at a time) thus speeding up the supply chain management

Other advantages of RFID technology include: RFID tags can stand a harsh environment, long read ranges, portable database, multiple tag read/write, and tracking items in real-time, *etc.* [5] gives a good description about RFID technology for supply chain management. RFID automates supply chain management, enabling enterprises to realize significant savings to the top and bottom line. RFID technology greatly helps enterprises to maintain the accuracy of shipments sent and received by parties throughout distribution. As a result we can keep a check on product theft, product counterfeiting, and it also helps in precise product recall.

1.2 Mobile RFID Technology

Currently RFID tags are still expensive, but very soon it would become economical to tag products at the item level. This will open the door for large-scale use of RFID tags on consumer goods. As a result, in near future we can realize, one of the visions of automatic identification and ubiquitous computing, which is the creation of an "Internet of Objects". In such a highly connected network; devices, objects, items of any kind dispersed through an enterprise or in our society can talk to each other, providing real-time information about the objects, location, contents, destination, and ambient conditions. This communication allows much-sought-after, efficient and easy machine-to-machine identification, communication, and decision-making [1]. Thus RFID technology will have a tremendous impact on our society, once it starts to assist people in their daily lives. A right step in this direction is Mobile RFID technology.

With mobile RFID technology, handheld portable devices like mobile phones and PDAs, apart from having the usual voice/data communicating features, also behave as RFID readers and RFID tags. As a result, Mobile RFID brings the conventional RFID technology closer to common users rather than just constraining it's usage to supply chain management. The following section describes the various applications of Mobile RFID technology.

1.3 Applications of Mobile RFID Technology

With Mobile RFID technology users can efficiently perform two major tasks, namely: download and view information represented by RFID tags, and machine-to-machine identification and communication.

Download & View Information represented by RFID tags: Just by bringing a Mobile RFID enabled portable device near to a RFID tagged object, we can quickly and easily download information represented by that RFID tag and view that information via mobile device's display screen. For example:

- We can download information about a particular location by scanning RFID tagged sign posts, and landmarks.
- We can download bus routes by scanning RFID tagged Buses.
- We can download prices of RFID tagged merchandise sold at stores, and published in catalogs for Smart Shopping.
- We can download movies, music, trailers, show timings, and theater locations by scanning RFID tagged movie posters, music CDs, *etc.*

Machine-to-Machine identification and communication: When Mobile RFID enabled portable device behaves as a RFID tag we can consider the following applications:

- We can authenticate ourselves to a RFID reader in order to access a particular facility (building, home, *etc.*) or services.
- We can carryout micro payments at subway stations, bus, newspaper stands, and gas stations by bringing our mobile device near to a RFID reader.
- We can give out information about our mobile device's model no. and size of it's display screen, in-order to download and view suitable multimedia content from a multimedia kiosk.
- We can make a quick call or send an instant message by scanning RFID tagged photographs, business cards, address books, *etc.*

We strongly believe that Mobile RFID technology has a great future and it's a very challenging research area. It is poised to be one of the future killer applications and services for mobile communications field.

1.4 Mobile RFID Application Zones

Mobile RFID applications can be broadly categorized into three zones namely: Location-based Services (LBS) Zone, Enterprise Zone, and Private Zone. From now on and Henceforth we consider a "mobile phone" to be our portable device, which has Mobile RFID enabled technology, *i.e.,* this mobile phone is incorporated with both RFID reader and tag functionalities. In the subsequent sections we describe each of these zones and their corresponding security and privacy threats.

2 Location-Based Services (LBS) Zone

In a LBS zone, service providers provide services that are "related to" and "available at" customer's current location. The coverage of this zone is very large, which includes all public places, roads, shopping malls, cinema halls, and food courts, *etc.* Service providers deploy RFID tagged items/devices (*e.g.,* posters, sign boards, maps, shopping catalogs, commodities, digital photo printers, multimedia servers, and RFID readers to receive payments, *etc.*) all around, which will enable us to carry out the above-mentioned two major tasks.

2.1 Security for Mobile RFID at LBS Zone

In this section we describe various security threats related to Mobile RFID at LBS Zone and later propose a security framework. Table 1, summarizes the security assessment of this zone.

In LBS zone, most of the RFID tags respond to every mobile phone, otherwise the main purpose of these tags to provide "location-based instant information" would be defeated. Therefore, we do not consider a tag-reader mutual authentication and strong secure communication between RFID tag and mobile phone. But there is one problem, these publicly available tags can be fake or must have been illegally modified (cloned) and no longer truly represent the information of the item in question. As a result, we at least need a one-way authentication mechanism, which authenticates the RFID tag to the mobile phone. [2] provides description of some of the RFID tag-reader authentication schemes that better serve this purpose. The most popular among them are challenge-response schemes that are based on symmetric key encryptions, hash functions, and hash chains.

Also we assume that for this task in LBS zone, most of the items/products are tagged with low-cost passive RFID tags like EPCglobal Class-1 Generation-2 UHF tags [4]. Generally a user's mobile phone may be used to scan one RFID tag at a time, we assume that the distance between the RFID tag and the mobile phone is too short to consider an active eavesdropping by an adversary. For further security assessment, let us consider the following scenario:

Scenario: *Alice visits a shopping mall. She uses her mobile phone to scan RFID tags attached to various items that are being sold. After scanning a particular RFID tag, the mobile phone is allowed to access shopping mall's "Information Server (IS)", which contains a detailed database about the scanned RFID tag. As a result, the mobile phone can download and store the price, picture, features, and manufacturer details of that item. The mobile phone must not be allowed to download other sensitive details like the number of pieces sold so far, its profit margin, and stock availability,* etc., *in order to prevent corporate espionage, this information is strictly for the shopping mall's inventory checking staff. Alice's mobile phone must also be protected from being directed to, accessing, downloading information, from malicious IS. Malicious IS can either induce virus code into to the mobile phone or extract sensitive data off the mobile phone. Alice must be able to scan tags in the shopping mall anonymously without revealing her true identity and buying habits. The shopping mall must be able to verify Alice's age incase she wants to download details about alcohol, and mature books or multi-media content. On the other hand, Charlie, an adversary, stalks Alice into an elevator. Charlie must be prevented from using his mobile phone to scan and retrieve sensitive information off any RFID tagged item that Alice is carrying in her bag/purse.*

From the above scenario, we identified the following security threats and security requirements:

Secure Job Delegation & Trust Model: There would be many competitive service providers selling location-based services to users. A user's mobile phone may need to communicate with many service provider's Information Server. Mobile phone should identify and authenticate genuine information servers and be able to secure the entire transaction and also protect the owner's privacy. But these tasks could create a huge burden on the low-computing and resource-poor mobile phone and is certainly not user friendly. Therefore it would be lot easier for the mobile phone to securely delegate its work to a trusted high-computing and resource-rich entity, such as a mobile operator. This approach helps in reducing the communication and computational burden on the mobile phone. Establishing an efficient and a convincing trust model is very much required to ensure secure transactions, key distribution, and job delegation. With existence of a trust model, it would be lot easier for the mobile phone to delegate its work to the mobile operator.

Detect Malicious Tag Information Servers: User's mobile phone must be allowed to access and download information from only genuine and authentic tag information servers. Therefore it is essential to authenticate and authorize every information server that the mobile phone is trying to access.

Authorized Tag Information Access: Some of the information represented by RFID tags must be available to only authorized people. But with the onset of Mobile RFID technology, RFID readers (incorporated into mobile phones) will soon become ubiquitous. Therefore it becomes essential for information servers to categorize which user's mobile phone is entitled to download what kind of information. This requires efficient authentication, authorization, and access-control protocol. Information represented by RFID tags must be made available to mobile phones, based on the privileges of the user *e.g.,* customer, staff, juvenile, adult, gold/platinum member of an organization, *etc.*

User Privacy Protection: After scanning a particular RFID tag for information, the identity and location of user must not be revealed to the service provider. This personal information could allow service providers and vendors to generate detailed profiles of the user, his buying interests, and transactions information. Adversaries must not be able to scan RFID tagged items already purchased by users.

Data Integrity & Confidentiality: We require secure Electronic Data Interchange (EDI) between the mobile phone and service provider's Information Servers.

Table 1 gives the summary of security threats and security requirements for this zone. We consider two distinct communication channels between: Mobile Phone & RFID Tag (for scanning a tag), and Mobile Phone & Service Provider's Information Server (for retrieving information represented by the tag).

Table 1. Mobile RFID-LBS Zone Security Assessment

Threat	Security Req.	Tag ↔ MP	MP ↔ SP-IS
User ID Privacy	Pseudonyms	O	O
	Anonymous Credential	O	O
Illegal Info. Access	Authentication	O	O
	Authorization	X	O
	Access Control List	X	O
Eavesdropping	Encryption/Decryption	X	O
	Digital Certificate	X	O
Key/Pwd	Trust Model	X	O
Compromise	Key/Pwd Management	X	O
MP: Mobile Phone SP-IS: Service Provider's IS		X: Not Req.	O: Req.

3 Building Blocks: Mobile RFID - LBS Zone

The building blocks of Mobile RFID infrastructure in LBS zone is similar to EPCglobal's RFID infrastructure. EPCglobal [6] is leading the development of industry-driven standards for the Electronic Product Code (EPC) to support the use of Radio Frequency Identification (RFID) in supply chain management. Due to space constraint we do not explain the EPCglobal RFID System Architecture. But the details we provide can be understood easily. Expect that we introduced mobile operator and eliminated the need of EPC Middleware. Since mobile RFID would mostly scan one tagged item at a time, there is no need for filtering software to make the mobile RFID data clear.

- Mobile RFID (M-RFID): Mobile Phone with both RFID Reader and Tag functionalities, is used to scan tagged items available everywhere.
- RFID Tags: every RFID tag contains its unique EPC number. EPC is a globally unique serial number that identifies an item in the supply chain. EPC data/number contains: EPC Manager number (identifies the company), Object class (similar to a stock-keeping unit, also called product number), Serial number (specific instance of the object class being tagged, objects own unique identifier). EPCglobal allocates manufacturers specific blocks of EPC numbers, and manufacturers then add their own product codes and serial numbers to their assigned manufacturer numbers to create unique identifiers - EPCs.

  Further information about the product is stored on a network of servers and databases called EPC Network. Therefore, unique EPC number acts like a pointer directing the RFID reader to the right entity on the EPC Network from where the reader can download additional related data about the product it scanned.
- Mobile Operator (MO): In the current mobile communications paradigm we have already put in a great deal of trust in MO, as it handles all our

voice and data communications. It maintains a record of each subscriber's call details, contact information, and credit card details, *etc.* It even has the capability to easily determine our current location and tap into our communications. But what protects us from MO turning hostile is that it has to very strictly adhere to and follow legal, security and privacy policies imposed by the law. Our architecture extends this trust in MO to secure and provide privacy protection for Mobile RFID transactions. This approach is very practical and easily deployable, as the current mobile communications infrastructure is widely spread and highly stable. MO takes responsibility on behalf of M-RFID to select, identify, and authenticate genuine ECP-IS. MO behaving like a "Trusted Proxy" processes the request on behalf of the M-RFID, greatly reducing the communication and computational burden on the user's mobile phone and also provides users privacy protection. MO also takes responsibility on behalf of M-RFID to select, identify, and authenticate only the genuine SPs and their information servers.

– EPC Network: Just like the global look-up system such as the Domain Name Service (DNS), VeriSign [5], after obtaining the contract from EPCglobal, has invested heavily in building and marketing an EPC Network specifically to look up EPC data. It becomes very necessary to look up each EPC number on a central data repository like we do with a Web page or other system using DNS. Keeping EPC data as an unique reference or primary ID, further information about the respective product is stored on databases and servers of EPC Network. This network assists local company staff and geographically distributed supply chain partners to easily and efficiently access information on any product they are handling from any location. The EPC Network [5] consists of three main components: Object Naming Service (ONS), the EPC-Information Services (EPC-IS), and the EPC-Discovery Services (EPC-DS).

4 Security Architecture: Mobile RFID - LBS Zone

This section describes our proposed security architecture of the Mobile RFID as depicted in Figure 1.

– Step 1: M-RFID scans a RFID tag
– Step 2: RFID tag responds with EPC number
– Step 3: M-RFID authenticates itself to MO via login ID/pwd and sends the EPC number to MO
– Step 4: MO sends EPC number to the ONS
– Step 5: ONS responds with URL of the EPC-IS related to the EPC number in question
– Step 6: MO fetches the anonymous M-RFID certificate from its database and sends it along with EPC number to the URL of EPC-IS. The certificate does not contain the identity of M-RFID but contains some related information like age, proof of privileged membership, *etc.*
– Step 7: EPC-IS verifies the certificate and checks the access-control list in its database.

– Step 8: Depending on the access rights of that certificate, EPC-IS responds to MO with related data about the EPC number in question.
– Step 9: MO sends the EPC information to the M-RFID. This communications can be encrypted using an established session-key
– Step 10: MO stores details of this transaction in the database of this M-RFID. Later, M-RFID can query some information about the tags it accessed previously on a particular date, time, location (for compare shopping) and also items it purchased.
– Step 11: M-RFID can purchase tagged items. MO can pay the vendor on behalf of M-RFID and later get the money from M-RFID via monthly telephone bills.

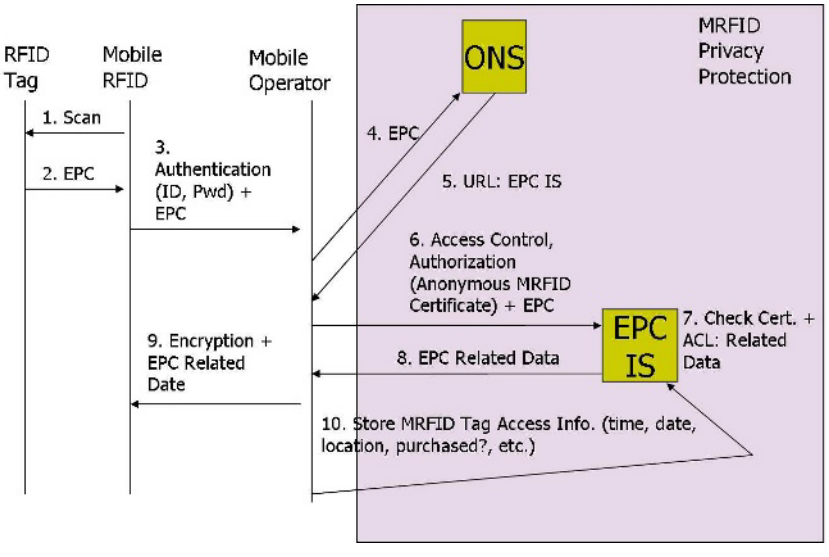


Fig. 1. Mobile RFID - LBS Zone Security Architecture

4.1 Security Solutions

Mutual Authentication mechanism between M-RFID and MO. A simple ID/Password authentication for M-RFID and MO's PKI certificate verification by M-RIFD is necessary for mutual authentication between M-RFID and MO. This provides secure job delegation, trust model, data integrity and confidentiality between M-RFID and MO.

Mutual Authentication mechanism between MO and EPC-IS. MO takes responsibility on behalf of M-RFID to select, identify, and authenticate only the genuine SPs and their information servers. This protects M-RFID from accessing malicious EPC-IS servers. Since MO and EPC-IS are resource rich entities, they both can authenticate each other via PKI-based certificates. Thus providing data integrity and confidentiality.

Anonymous Certificates for Identity management, authentication, and authorization. M-RFID can request anonymous certificate from MO. This certificate does not contain the true identity of M-RFID but contains other details like age, whether the user is a gold card member or not, staff or visitor, *etc.* This protects the privacy of the owner of M-RFID and also assists EPC-IS to provide corresponding information about the EPC number in question.

M-RFID privacy. Our approach protects both location and information privacy of M-RFID. With the use of anonymous certificate the vendor or the service providor of the tagged item can never know the true identity of the M-RFID's owner. To prevent an adversary from scanning the handbag of Alice, and obtain information about the tagged items purchased by her, we suggest the following two approaches:

Kill the Tag: EPCglobal Class 1 Gen 2 UHF Tags [4] can be embedded with Kill Password. Whenever a RFID reader send this Kill Password to the tag, the tag is killed and rendered permanently unusable and unreadable. Therefore, once a tagged item is purchased by M-RFID, the trustable clerk at the point of sale (cash counter) can obtain the tag's kill password from the shopping mall's information server and using this kill password the clerk can kill the tag permanently. But this approach has a drawback if the customer wants to make use of the tag capabilities at his home, *e.g.,* RFID enabled refrigerator or book shelf, *etc.*

Lock the Tag: EPCglobal Class 1 Gen 2 UHF Tags [4] can be embedded with a 32-bit value Access Password, which means that only a reader that already possesses the right access password can perform mandatory commands on the tag, such as Read, Write, and Lock. Therefore tag's access password can be used for "reader to tag" authentication and in the process allows the reader to access the locked memory banks within the tag, permission to change the lock status of the memory banks, and write data into the tag, etc. A tag has four memory banks: Reserved, EPC, TID, and User. Reserved memory bank is used to store the Kill Password and Access Password, EPC memory bank for EPC number, TID memory bank for tag's unique manufacturer identity number, and User memory bank for additional user data. The reserved memory bank of the tag is permanently locked; as a result the access password can neither be read nor modified by any reader.

Therefore once a tagged item is purchased by M-RFID, the trustable clerk at the point-of-sale (cash counter) can obtain the tag's access password from the shopping mall's information server and using this access password the clerk can lock all the memory banks of the tag including the EPC memory bank. The M-RFID can obtain and store the tag's access password at the point-of-sale. Now the customer can use his/her M-RFID to lock and unlock the tag whenever and wherever required. Since an adversary does not know the tag's access password he can no longer track or get any data from the tag as all the memory banks are locked. Through this approach the tag need not be killed permanently.

5 Enterprise Zone

In this zone mobile phone assists company's mobile staff/employees like inventory checkers, field engineers, maintenance and repair staff, and security guards. It helps them in real-time inventory management, work attendance log, instructions on how to operate tagged items, 'identification of' and 'access control to' tagged equipment and secure enclosures, and proof of staff presence at certain locations in a building that needs to be monitored periodically, *etc.*

The security framework for enterprise zone Mobile RFID applications could be proprietary and confined to the boundaries of a particular organization. In such a confined and well-monitored zone it's not very difficult to establish and enforce an efficient security architecture, trust model, and security & privacy policies. With the availability of up-to-date list of registered employees and items/products in a company; designing and implementing key/ password distribution, data integrity & confidentiality, identification, authentication, and access control protocols among staff, RFID readers, RFID tagged items, and EPC Network is moderately easy and mostly risk free when compared to LBS zone.

Since this zone needs precise authentication and security auditing in order to access RFID tagged items, we require tag-reader mutual authentication and also the true identity of the M-RFID must be revealed, therefore user privacy may not be needed. Table 2, summarizes the security assessment of this zone.

Table 2. Mobile RFID-Enterprise Zone Security Assessment

Threat	Security Req.	Tag ↔ MP	MP ↔ E-EPC
User ID Privacy	Pseudonyms	X	X
	Anonymous Credential	X	X
Illegal Info. Access	Authentication	O	O
	Authorization	O	O
	Access Control List	X	O
Eavesdropping	Encryption/Decryption	X	O
	Digital Certificate	X	O
Key/Pwd	Trust Model	X	X
Compromise	Key/Pwd Management	O	O
MP: Mobile Phone	E-EPC: Enterprise's EPC n/w	X: Not Req.	O: Req.

6 Private Zone

In this zone, mobile phone assists users in their private space like home, garden, garage, car, and workshop. It helps them to make an instant call or send an instant message by scanning RFID tagged photographs, business cards, and address books. By scanning RFID tagged household items with a mobile phone,

we can quickly obtain information like; when would the milk stored in the refrigerator expire, details of the books in the bookshelf, when was the last time a RFID tagged plant has been watered, and when to change the engine oil, *etc.*

This zone is small when compared to the other two zones and therefore it requires a simple security model that can be easily deployed and maintained by the user at his home. Users in this zone can buy off-the-shelf Mobile RFID Kits. These kits can contain RFID tags, Mobile RFID, related hardware, and software with user-friendly GUI. The software can assist the users to easily encode EPC numbers of their choice into the RFID tags, create a portable database in their PC with details about the tagged household items, create passwords to access these tags and the database, and finally secure the wireless/WiFi network in the home environment.

Other option could be, the user can obtain storage space (for free or fee) on the EPC Network (EPC-Information Servers) and via a password protected user-friendly website, he can upload his personal EPC numbers and details of the tagged household items. Whenever he scans his private RFID tag in his home, the Mobile RFID contacts his personal page on the EPC-Information Server and downloads the details about the item in question. This approach alleviates user's burden of configuring his own security system. The EPC-Information Server must provide user privacy protection, and secure communication.

We need to protect this zone from malicious RFID readers sitting outside this zone and trying to track the RFID items inside the zone (*e.g.,* all the expensive items inside the home that are worthwhile to steal). To ward off this threat we need reader to tag authentication. The tag must allow only authorized readers from within the home to scan and query it. Other approach is to install equipment outside the home, that would jam any external malicious noise or radio signals from entering inside the home. Sometimes it may be required that the children, guests and visitors to this zone are provided with different access control rights to the tagged devices. Therefore we need user identity and access

Table 3. Mobile RFID-Private Zone Security Assessment

Threat	Security Req.	Tag ↔ MP	MP ↔ U-EPC
User ID Privacy	Pseudonyms	X	X
	Anonymous Credential	X	X
Illegal Info. Access	Authentication	O	O
	Authorization	O	O
	Access Control List	X	O
Eavesdropping	Encryption/Decryption	X	O
	Digital Certificate	X	O
Key/Pwd Compromise	Trust Model	O	O
	Key/Pwd Management	O	O
MP: Mobile Phone U-EPC: User's Private EPC n/w		X: Not Req.	O: Req.

control list, which specifies the rights and capabilities of the users in this zone. Table 3, summarizes the security assessment of this zone.

7 Conclusion

This paper provides future vision and security challenges of Mobile RFID. We mentioned the various security threats and security requirements at different zones of Mobile RFID applications namely LBS, enterprise, and private zones. And proposed a simple security architecture for the LBS zone, that fits the RFID EPC Network. The advantages of this architecture are as follows: simple, involves less user interactions, secure job delegation between Mobile RFID and Mobile Operator. Also the Mobile Operator conceals the identity of users, as a result service providers and vendors of tagged items cannot maintain users detailed profiles and location information, this protects users privacy. It could be a good revenue generator for the mobile operator and service providers through commissions for every transaction. Our approach is practical and easily deployable, as the current mobile communications infrastructure is widely spread and highly stable. And vendors can still use the the popular RFID EPC network. As our future work we would propose more concrete security architectures for the other two zones of Mobile RFID applications and also propose a simple, secure and privacy preserving payment phase for Mobile RFID applications.

Acknowledgement. This work was supported in part by "Development of Sensor tag and Sensor node technology for RFID/USN" project of ETRI through IT Leading R&D Support Programs of MIC, Korea.

References

1. Patrick J. Sweeney II, "RFID for Dummies", Wiley Publishing,Inc., ISBN: 0-7645-7910-X, 2005.
2. Ari Juels, "RFID Security and Privacy: A Research Survey", RSA Laboratories, 2005,
3. EPCglobal Web site, 2005, http://www.EPCglobalinc.org
4. EPCglobal Inc., "Class 1 generation 2 UHF air interface protocol standard version 1.0.9.", Referenced 2005 at http://www.epcglobalinc.org/standards/
5. VeriSign, "The EPCglobal Network: Enhancing the Supply Chain", White Paper 2005, http://www.verisign.com/stellent/groups/public/documents/white_paper/002109.pdf
6. EPCglobal Specification, "EPCglobal Architecture Framework Version 1.0", http://www.epcglobalinc.org/standards/

An Efficient Forensic Evidence Collection Scheme of Host Infringement at the Occurrence Time*

Yoon-Ho Choi[1], Jong-Ho Park[1], Sang-Kon Kim[1], Seung-Woo Seo[1], Yu Kang[2], Jin-Gi Choe[2], Ho-Kun Moon[2], and Myung-Soo Rhee[2]

[1] School of Electrical and Computer Engineering, Seoul National University, Seoul, Korea, 151-744
[2] KT Information Security Center, Seoul, Korea

Abstract. The Computer Forensics is a research area that finds the malicious users by collecting and analyzing the intrusion or infringement evidence of computer crimes such as hacking. Many researches about Computer Forensics have been done so far. But those researches have focused on how to collect the forensic evidence for both analysis and proofs after receiving the intrusion or infringement reports of hosts from computer users or network administrators. In this paper, we describe how to selectively collect the forensic evidence of good quality from observable and protective hosts at the time of infringement occurrence by malicious users. By correlating the event logs of Intrusion Detection Systems(IDSes) and hosts with the configuration information of hosts periodically, we calculate the value of infringement severity that implies the real infringement possibility of the hosts. Based on this severity value, we selectively collect the evidence for proofs at the time of infringement occurrence. As a result, we show that we can minimize the information damage of the evidence for both analysis and proofs, and reduce the amount of data which are used to analyze the degree of infringement severity.

1 Introduction

With the advent of transaction using Internet, the infringement of personal information and information leakage with many serious damages has been reported. However, the evidence for both analysis and proofs can be modified by the malicious or naive behavior so that it becomes not easy to investigate these crimes effectively when they happen. Therefore, Computer Forensics, or simply called Forensics, has become an important security area, which considers the collection of the non-damaged forensic evidence, its analysis, the inference of the behavior, and the trace-back of the malicious user.

In the traditional Forensics, the gathering, saving and analysis of the evidence have been done at the time of the infringement occurrence such as system hacking, as shown in Fig.1-(a). That is, based on the time of the infringement report

* This research was supported by the University IT Research Center Project.

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 206–221, 2006.
© Springer-Verlag Berlin Heidelberg 2006

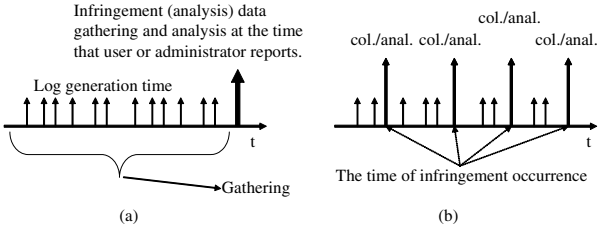


Fig. 1. Comparison of infringement report time: (a)previous(gathering) (b)proposed (collection) where, col. means 'collection' and anal. means 'analysis'

from users or administrators, all the evidence for analysis and proofs from the damaged system has been gathered and analyzed entirely. But, it may occur that the amount of data for analysis of infringement becomes too much because every set of evidence data that has been gathered has to be investigated on. Also, the evidence for analysis and proofs may have been damaged by an attacker or changed by the activity of a normal user. Especially, when the intentional infringement against the system happens, the damage of the evidence for analysis and proofs becomes more serious.

1.1 Previous Approaches for Correlation Analysis

The previous approaches based on correlation analysis aim at low False Positive(FP) and low False Negative(FN) rates, and log reduction by combining a refining algorithm with a fast search algorithm. The correlation approaches are classified into ones that do not require a specific knowledge and ones that rely on a certain knowledge. The proposed approach is categorized into the approach that relies on a certain knowledge.

As the approaches that use a specific knowledge, there have been made several proposals: the Advanced Security Audit-trail Analysis on UniX(ASAX)[AS1] [AS2] based on the rules that are formed as (prerequisite, actions), JIGSAW[SK1] based on attack scenario, Chronicles[BH1] based on time reasoning, and others such as EMERALD Mission-Impact Intrusion Report Correlation System(M-Correlator)[Em1] and M2D2[BM1]. ASAX aims at supporting intelligent analysis of audit trails. ASAX uses a general rule-based language to carry out its analysis, named RUSSEL(Rule-baSed Sequence Evaluation Language) which aims at recognizing particular patterns in files and triggering appropriate actions. The audit trail is analyzed sequentially, record by record, by means of a collection of rules. Active rules encapsulate all the relevant knowledge about the past of analysis and it is applied to the current record by executing the rules for that record. And then, it generates new rules and the process is initiated by a set of rules activated for the first record. ASAX has some limitations in that there is no real rules database and data types are limited within RUSSEL.

JIGSAW is based on the preconditions and consequences of individual attacks. It correlates alerts if the preconditions of some later alerts are satisfied by

the consequences of some earlier alerts. However, it does not correlate an alert if it does not prepare for other alerts. For example, if an attacker tries several variations of the same attack in a short period of time, JIGSAW will treat them separately, and only correlate those that prepare for other alerts. Chronicles aims at providing an interpretation of the system evolution given dated events. It is a time series reasoning system that relies on the reified temporal logic formalism. It predicts forthcoming events relevant to its task; it focuses its attention on them and maintains their temporal windows of relevance. It is efficient in recognizing stereotyped attack scenarios such as the ones launched by automatic intrusion tools. EMERALD M-Correlator was designed to consolidate and rank a stream of alerts relative to the needs of the analyst, given the topology and operational objectives of the protected network. It uses a relevant score produced through a comparison of the alert target's known model against the known vulnerability requirements of the incident type. M2D2 is a formal information model for security information representation and correlation which includes four types of information: information system's characteristics, vulnerabilities, security tools and events/alerts. M2D2 reduces and conceptually interprets multiple alarms, i.e. it models alert aggregation method by utilizing relations between vulnerabilities and topology, between topology and security tools, as well as between security tools and vulnerabilities. However, these approaches are focusing on the efficient detection of intrusion attempt.

1.2 Proposed Approach

Before we describe the proposed approach, we first define the two following terminologies that are used throughout this paper.

- **Gathering** means that we collect the evidence entirely for the investigation of host infringement at the report time from administrators or users;
- **Collection** means that we collect the evidence selectively for the investigation of host infringement at the occurrence time.

As shown in the Fig.1-(b), different from the previous approaches that gather the evidence for analysis and proofs entirely as in the Fig.1-(a), the proposed approach focuses on the detection of real infringement against the observable and protective hosts and focuses on collecting the forensic evidence at the time of occurrence of intentional infringement against them. Noting that when the infringement against the host occurs, it usually takes the form of a multi-step(or stage) attack, we describe how to collect the evidence for both analysis and proofs from the early stage of infringement such as Host Scanning(HS), Port Scanning(PS) and Vulnerability Exploit(VE) to the final stage such as Distributed Denial of Service(DDoS). After we classify the intrusion into the intrusion attempt and infringement that means a real damage at the system, we calculate the value of infringement severity representing the real infringement possibility of the hosts in a multi-step attack. Based on the value of infringement severity of the system for each attack step, we determine the time instant for the forensic

evidence collection. To minimize the possibility of wrong decision on the time of the forensic evidence collection, we correlate the event logs of the Intrusion Detection System(IDS) like the SNORT[Sn1] with the event logs and the security configuration of the host during calculation of the value of infringement severity. We store the evidence for both analysis and proofs at each step of a multi-step attack as a status information for each stage. We summarize the contributions of this paper in Table.1 by comparing with the previous approaches.

This paper is organized as follows. In section II, we describe the proposed approach that calculates the value of infringement severity of the hosts and collects the forensic evidence from the hosts based on the value. We describe the verification result of the proposed approach in section III. In section IV, we summarize the paper.

Table 1. Comparison with the previous log correlation and analysis methods

	Previous approaches	Proposed approach
Target	◇ Detection of intrusion attempt ◇ Forensic evidence gathering at the time of administrator(user) report	◇ Detection of host infringement ◇ Forensic evidence collection at the time of infringement occurrence
Analysis method	◇ Correlation analysis among host event logs ◇ Correlation analysis among IDS event logs	◇ Correlation analysis among event logs of both IDS and host ◇ Calculation of the infringement severity of hosts
Effect	◇ Accurate intrusion attempt detection i.e., low FP and FN	◇ Measurement for the degree of real infringement of hosts ◇ Low loss of the forensic evidence ◇ Reduction of evidence for analysis
Limitation	◇ Loss of evidence for both analysis and proofs ◇ Dummy analysis for not infringement but intrusion attempt ◇ No way to collect any volatile data	◇ No way to collect some volatile data

2 Infringement Decision and Forensic Evidence Collection

Now, we describe how to calculate the value of infringement severity of the hosts for each attack step and which information should be collected. We assume the general multi-step attack shown in Fig.2, for example, DDoS whose final attack is given as Denial of Service(DoS).

2.1 Overall Description of the Analysis Objects and Its Procedure

As shown in Fig.3, the evidence for analysis is collected to a collaborative analysis server which monitors IDSes and the observable and protective hosts. The

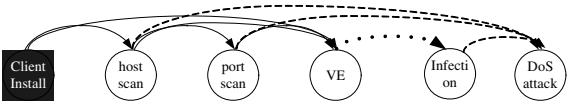


Fig. 2. Infringement represented by stages and the possible attack flow, where the sold line implies the beginning stage of infringement; the dotted line implies the infringement path selection on networks; the dashed line implies a final attack; Client is an attacker; DoS is assumed to be an example of a multi-step attack. For example, Host scan→Port scan→NETBIOS SMB NT Trans NT CREATE oversized Security Descriptor attempt(represents the solid line up to this part)→Trin00 daemon install(represents the dotted line)→DoS attack(represents the dashed line).

Table 2. Definition for the parameters or terms to the proposed approach

Parameters(terms)	Description
state(s, IP)	State for infringement decision at stage s for the host with ip address as IP, where {the evidence for analysis at stage s, the forensic evidence for stage s, infringed host IP(port), attacker IP(port), patterns for event analysis}, s=1∼5 or HS(:1), PS(:2), VE(:3), D(:4), DoS(:5)
state(s, ss, IP)	Sub-state ss for infringement decision at stage s of host IP when there are different evidence for analysis that shows the same symptom of infringement at stage s, where same as state(s, IP)
v(s, IP)	The value of infringement severity of host IP at stage s
v(s, ss, IP)	The value of infringement severity of host IP for sub-state ss at stage s
pre-condition	Conditions that the host infringement succeeds
post-condition	Evidence for analysis of the infringement
$c_k(s)$	k_{th} condition for the infringement decision at stage s , where (pre-condition, post-condition)
$c_m(s, ss)$	m_{th} condition for the analysis of sub-state ss at stage s , where (pre-condition, post-condition) and s, m={1,2,...}
C(s)	The set of pre-conditions at stage s composed of $c_k(s)$ and $c_m(s, ss)$, where {$c_k(s)$, $c_m(s, ss)$}
$\|C(s)\|$	Number of the elements in C(s)
wt(s)	Weighting factor for stage s which is tunable
wt(s, k)	Weighting factor for k_{th} condition at stage s which is tunable
wt(s, ss, m)	Weighting factor for m_{th} condition of sub-state at stage s which is tunable
Lo_N	Number of files for analysis of infringement
Li_N	Number of lines in each log file
S(s)	State information at stage s for all the hosts where, {analysis time, v(s, IP), total value of infringement severity at stage s for all the hosts}

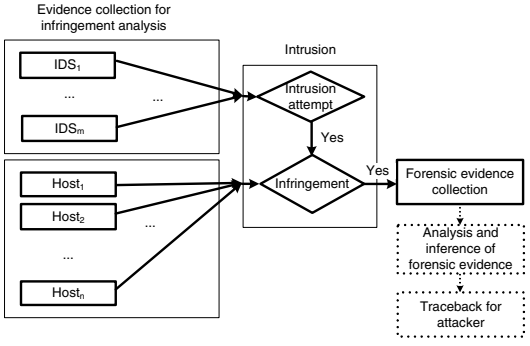


Fig. 3. Diagram of the proposed approach for infringement decision and forensic evidence collection

proposed approach analyzes the event logs of IDSes for detection of intrusion attempt. The event logs and configuration informations of the hosts are used to analyze the degree of infringement severity. From the beginning stage to the end stage of a multi-step attack as shown in Fig.2, the approach identifies the infringement for stage s of the hosts and store the evidence for analysis as the state information. The state information includes the followings: who did the malicious behavior such as hacking(Attacker IP(port)), what one did(the forensic evidence at stage s), how one did(the evidence for analysis), where one did(Infringed host IP(port)), when one did(time information of log). And then, if the value of infringement severity exceeds a criterion for the forensic evidence collection at each stage, the sever requests hosts to send the forensic evidence selectively and stores them to the data storage.

2.2 Calculation of the Value of Infringement Severity and Collection of the Forensic Evidence

Based on the parameters or terms that we define in Table.2, we now describe how to calculate the value of infringement severity of the hosts and how to collect the forensic evidence. We assume that the timer of the IDSes and hosts are synchronized and the time stamps of logs from the IDSes and hosts are reliable.

- **Step 1.** Preprocessing
  After we periodically, with time period T, collect the state information, i.e., state(s, IP) and sub-state(s, ss, IP), for analysis from the IDSes and the hosts, we classify them into the corresponding information for each stage.
- **Step 2.** Calculation of the degree of infringement severity
  Considering the time sequence of the evidence for analysis and the state information at each stage, the Severity Management Server, called SMS, calculates the degree of infringement severity of each host at each stage.

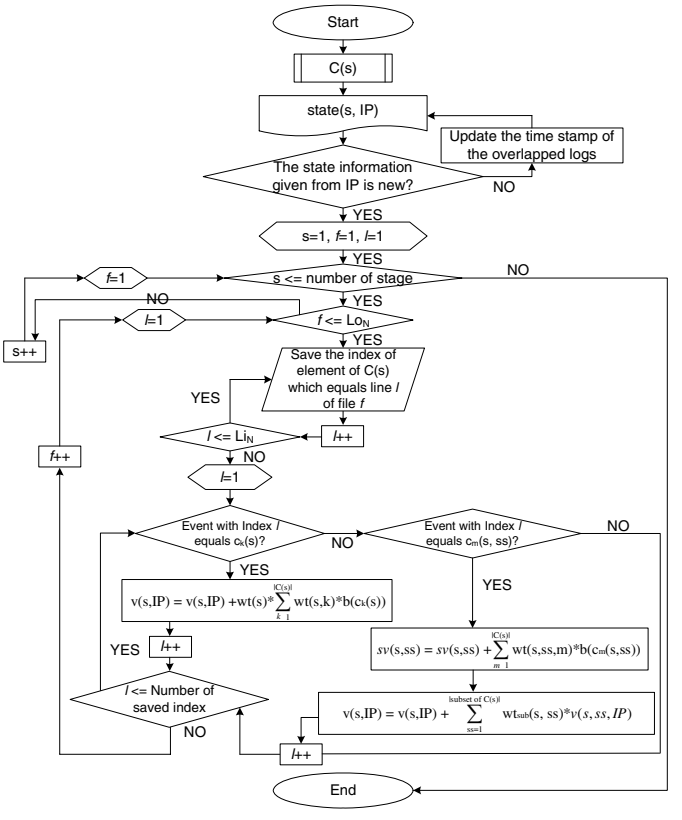


Fig. 4. Flowchart for the infringement severity calculation at stage s for the host with IP address as IP, where the computational complexity of the algorithm is given as $\mathrm{O}(Li_N)$ because $Lo_N \ll Li_N$ and (number of indices given from the analysis of a file) $\ll Li_N$

In Fig.4, we describe how to calculate the degree of the infringement severity in detail. If the received file is first given from the host with ip address as IP, we compare the evidence for analysis of state(s, IP) with C(s) and calculate the degree of infringement severity of the host with ip address as IP at each stage. If l_{th} line of log file f(post-condition) matches to one of $c_k(s)$(pre-condition), we calculate the degree of infringement severity by using the following equation (1):

$$v(s, IP) = v(s, IP) + wt(s) * \sum_{k=1}^{|C(s)|} wt(s, k) * b(c_k(s)), \tag{1}$$

where the binary variable $\mathrm{b}(c_k(s))$ is given as 1 if a pre-condition equals a post-condition and 0 otherwise, and $\sum_{k=1}^{|C(s)|} wt(s, k) = 1$. On the other hand, if l_{th} line of log file f matches to one of $c_m(s, ss)$, we calculate the

degree of infringement severity by using equation (2). We note that the same information for infringement at each stage can be given from different log files with the different degree of reliability. The case frequently happens in the process of infection. For example, we assume that an attacker tries to infect host A with Linux OS. The attacker executes some active processes and leaves some traces. This trace may be found by executing a command, 'ps -atn'. However if a new process is executed from this host, this information can not be found by executing the command (low reliability). On the other hand, IDS can have a trace that a specific string found in the daemon of attack tool is recorded, for instance '*HELLO*' message. This information does not change even if the other infection strings are detected by IDS(high reliability). So, we need to set the different weighting factor for each case, i.e., the weighting factor for the analysis result with higher reliability should be larger than that with low reliability. We calculate the degree of infringement severity of each host by considering the subsets of C(s) grouped according to the degree of reliability of records, where N_s means the number of subsets in C(s) and N_{ss} the number of elements in a subset.

$$v(s, IP) = v(s, IP) + \sum_{ss=1}^{N_s} wt(s, ss) * v(s, ss, IP) \tag{2}$$

$$v(s, ss, IP) = v(s, ss, IP) + \sum_{m=1}^{N_{ss}} wt(s, ss, m) * b(c_m(s, ss)), \tag{3}$$

where the binary variable b($c_m(s, ss)$) is given as 1 if a pre-condition equals a post-condition or 0 otherwise, and $\sum_{m=1}^{N_{ss}} wt(s, ss, m) = 1$.

Related to the weighting factors, we note the followings. If the decision at a stage is dominant to make an infringement decision for the final attack, the weighting factor of the stage, wt(s), has a larger value compared to the weighting factor of other stages. Also, if the decision based on a subset is more important than others, the weighting factor of the subset, wt(s, ss), is larger than others. Similarly we set larger values to more important conditions than others. Generally, as the infringement stage is close to the final stage, the weighting factors become larger and the weighting factor of the final stage equals to the sum of other weighting factors ahead of it. We also note that if the same symptom from the same log files is founded with the time difference, we only store the new information.

– **Step 3.** Forensic evidence collection
Now, referring to the following criteria for the value of the infringement severity given from step 2, we collect the forensic evidence for each stage from the infringed hosts.
  - **Case 1. The infringement of a specific observable host:** $min.th(s) < v(s, IP) < avg.th(s)$, where $min.th(s)$ is calculated by min[wt(s)*wt (s,k)] or min[wt(s,ss)*wt(s,ss,m)] and $avg.th(s)$ by wt(s)*(average value of wt(s,k) with respect to k).

We collect the forensic evidence for stage s from a specific observable host with ip address as IP .

- **Case 2. The infringement of a specific observable host and its surrounding observable hosts:** $avg.th(s) < v(s, IP)$ **or** $sum.min.th(s) < \sum_{IP} v(s, IP) < sum.avg.th(s)$, where $sum.min.th(s)$ is calculated by the summation of $min.th.(s)$ and $sum.avg.th(s)$ by the summation of $avg.th(s)$.

  We collect the forensic evidence for the final stage from a specific observable host with ip address as IP and for stage s from the specific host and its surrounding observable hosts. If $avg.th(s) < v(s, IP)$ is satisfied, it means that post-conditions more than one which match to pre-conditions or at least one post-condition which matches to a pre-condition with larger weighting factor is found from analysis. Thus, we need to suspect the possibility of the final attack against the specific observable host with ip address as IP and of the infringement for stage s against the specific host and its surrounding observable hosts.
- **Case 3. The infringement of the observable and protective hosts:** $sum.avg.th(s') < \sum_{s=1}^{s'} v(s, IP)$ **or** $total.avg.th(s') < \sum_{IP} \sum_{s=1}^{s'} v(s, IP)$, where $total.avg.th(s')$ is calculated by the summation of $sum.avg.th(s)$ for all the observable and protective hosts.

  We collect the forensic evidence by state s' from all the observable hosts and by the final stage from the protective host. It means that all the observable and protective hosts have high possibility of the infringement by stage s'.

Now, in Table.3, we summarize which evidence for proofs should be collected for each stage selectively.

Table 3. The forensic evidence collected from a host for each stage, where we assume DoS as an example of a multi-step attack

Stage	Forensic evidence
HS	Nothing
PS	date and time at the beginning of data collection(BT), open TCP/UPD ports, date and time at the end of data collection(ET)
VE	BT, the list of active processes, the useful information for the system such as information of CPU, the version of OS, the operation time of system, the domain name and the host name, information of all swap partitions, all file systems, information for the mounted file systems, ET
Infection(D)	BT, image of physical memory, the list of modules in kernel memory, active and doubt list of processes, the useful information for the system, the cache table collected from arp and routing tables, information of all swap partitions, all file systems, ET
DoS	BT, the cache table collected from arp and routing tables, ET

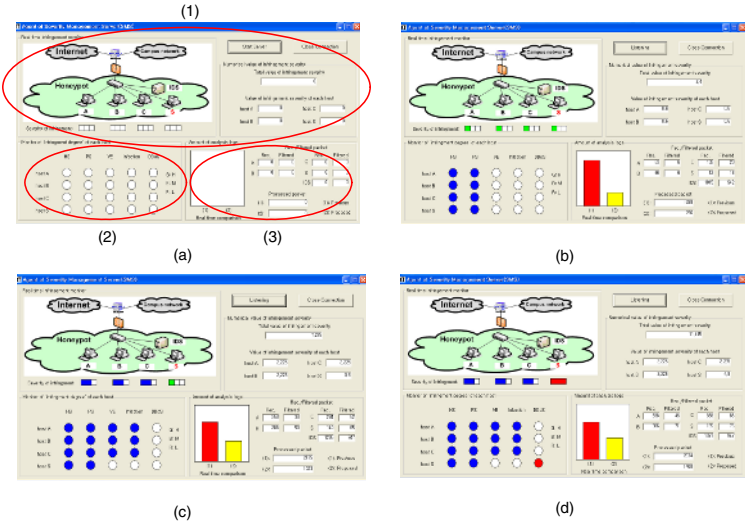


Fig. 5. Sequential operation of the prototype software under the attack scenario: (a) shows the configuration of the prototype software where, (a)-(1) shows the real-time degree of infringement severity of each host in terms of total value of infringement severity for stages, (a)-(2) the real-time degree of infringement severity for various stage of each host and (a)-(3) the real-time amount of evidence for analysis collected from the hosts. (b), (c) and (d) show a sequential operation of the prototype software under the attack scenario. (b) shows the operation in HS and PS, (c) in VE and Infection and (d) in DDoS.

3 Implementation

In this section, we describe the performance evaluation results of the proposed approach. To evaluate the performance of the proposed approach, we implemented the Honeypot between the campus network and Internet as shown in Fig.5-(a)-(1) and executed the DDoS attack against the Honeypot. We configureed the Honeypot with one protective host, i.e., host S, and three observable hosts, i.e., host A, B and C. By implementing the agent for analysis as shown in Fig.5, we correlate the state information, i.e. state(s, IP), from the hosts. As an example, we describe the process of the experiment in detail in Appendix.

In the process of the experiment, we reduce can FN and FP for infringement severity decision by using the following techniques. To reduce FN for infringement severity decision, we detect malicious packets by using the exact matching algorithms in the SNORT. And, we collaborate the event logs from hosts and the event logs from IDS. Also, to reduce FP for infringement severity decision, we use the patterns of shorter length for each signature group. And, we assign the small weighting factors for the conditions whose pre-conditions equal post-conditions including the shorter patterns, otherwise large weighting factors. Additionally,

we set a small weighting factor for the condition which matches the evidence for analysis which can frequently change by the normal behavior.

3.1 Infringement Monitoring

As shown in Fig.5-(a), we implemented the prototype software of SMS. The monitor of the agent consists of three parts. Fig.5-(a)-(1) is the panel which shows the degree of infringement severity in terms of summation for stage values of each host. Fig.5-(a)-(2) shows the degree of infringement severity for various stage of each host. Fig.5-(a)-(3) compares the amount of logs for the proposed approach with those for the previous approach which gathers the forensic evidence. From Fig.5-(b) to Fig.5-(d), we show the real operation of this agent. While an attacker executes HS and PS for host A, B, C and S, as shown in Fig.5-(b), the colors of HS and PS indicators located at Fig.5-(a)-(2) change from white(zero) to blue(medium severity). And then, while the attacker proceeds to VE and infection for host A, B and C, as shown in Fig.5-(c), the colors of VE and D indicators change from white(zero) to blue(medium severity). Finally, when the attacker proceeds to DDoS, the color of DDoS indicator for host S changes from void to red(high severity).

Another indicator located at Fig.5-(a)-(1) changes its color according to the degree of total infringement severity for stages of each host. For HS and PS, the color of the indicator changes from white to green. And then, for VE and D, the color of the indicator changes from green(low severity) to blue(medium severity). Finally, the color of the indicator for host S changes from green to red. Thus, by referring to the colors of the indicators located at Fig.5-(a)-(1) and Fig.5-(a)-(1), we can monitor the degree of infringement severity for observable and protective hosts.

3.2 Performance Evaluation

We evaluated the performance of the proposed approach by changing the time period T. We compare the amount of data collected from the hosts for two cases, i.e., the previous approach which gathers the evidence for analysis and the proposed approach which collects the evidence for analysis. We show the comparison results under the attack scenario in Fig.6 and Fig.7. For T=30sec, a total of 199213 bytes of data and 80719 bytes(1681 bytes/T) of data for the previous and proposed approaches are gathered and collected, respectively. Also, for T=10sec, a total of 100624 bytes(708 bytes/T) of data for the proposed approach are collected. As a result, under the same attack scenario, we get 60% and 49.5% evidence reduction for analysis so that it can reduce the analysis overhead caused by the dummy data. But the proposed approach may increase the average load of SMS because of the frequent access from hosts. On the other hand, it can reduce the peak load of SMS by reducing the amount of evidence for analysis. Thus, to operate the agent efficiently, we need to consider the size of analysis network. In the experiment, no variation of the system resource usage such as CPU and memory was found but we could reduce the amount of evidence for analysis.

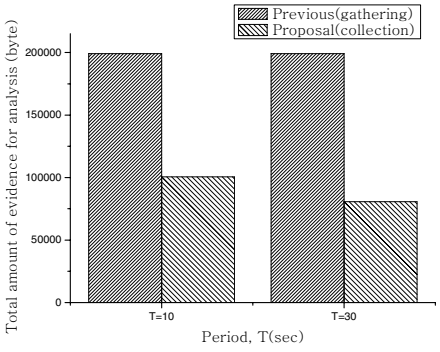


Fig. 6. Comparison of the total amount of evidence for analysis by varying the period: previous(gathering) vs. proposal(collection)

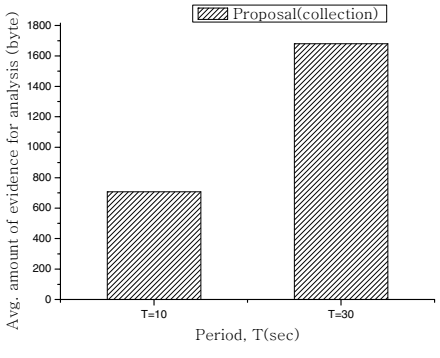


Fig. 7. Comparison of the average amount of evidence for analysis for different period of the proposed approach

4 Conclusion

In this paper, we proposed an efficient forensic evidence collection scheme based on the new approach of the detection of host infringement severity. For a multi-step attack, the proposed approach collects the evidence for analysis such as vulnerabilities, system logs and event logs from hosts and event logs from IDS. From them, it makes a decision on the infringement of hosts based on the value of infringement severity for each stage of a multi-step attack. From the decision on real host infringement, the proposed approach collects the forensic evidence at the time of infringement occurrence selectively. Also, it reduces the amount of evidence for analysis. Thus, the proposed approach gives a guideline to determine the forensic evidence collection at the time of host infringement selectively.

References

[Sn1] Snort $v2.0$, an open source network intrusion detection system, http://www.snort.org
[AS1] ASAX: Software Architecture and Rule-Based Language for Universal Audit Trail Analysis, 1992.
[AS2] Advanced Security Audit Trail Analysis on (ASAX also called SAT-X), 1994.
[SK1] S. Templeton and K. Levit: "A requires/provides model for computer attacks", In Proc. of New Security Paradigms Workshop, p31 38, September, 2000.
[BH1] B. Morin and H. Debar. Correlation of intrusion symptoms: "An application of chronicles", In Proceedings of the 6th International Conference on Recent Advances in Intrusion Detection(RAID03), September, 2003.
[Em1] P. A. Porras and P. G. Neumann: "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances", National Information Systems Security Conference, 1997.
[BM1] Benjamin Morin and al.: "M2D2: a formal data model for IDS Alert Correlation", Proceedings of RAID 2002,Zurich, Switzerland, October, 2002.
[HA1] Hervé Debar, Andreas Wespi: "Aggregation and Correlation of Intrusion Detection Alerts", in proceedings of RAID, 2001.
[FA1] Frederic Cuppens, Alexandre Miege: "Alert Correlation in a Cooperative Intrusion Detection Framework", in proceedings of IEEE S&P, 2002.
[Ch1] A tool to locally check for signs of a rootkit, http://www.chkrootkit.org/
[Ma1] Mariusz Burdach: "Forensic Analysis of a Live Linux System, Pt. 1,2", http://www.securityfocus.com/, 1997.
[Nm1] nmap-3.93, a free open source utility for network exploration or security auditing, http://www.insecure.org/nmap/
[Ne1] Nessus 2.2.8, the network vulnerability scanner, http://www.nessus.org/
[Ip1] iplog 2.2.3, a TCP/IP traffic logger, http://www.freshports.org/net/iplog/

A Appendix

A.1 Experiments

To analyze the performance of the proposed approach, we implemented the software agent over the client-server model. We constructed a SMS with CPU of clock speed 2.7GHz and DDRRAM 1Gb. To consider the graphical viewing of the operation, we implemented the agent of the SMS over windows XP. For a client, we made it operate over Pentium 3 CPU of clock speed 700MHz and SDRAM 256MB. To expose vulnerabilities of the client Apache, PHP 4, Mysql and Zeroboard 4.1pl5 over Linux RedHat 7.0 were operated. Based on these system configurations, we executed the experiments under the traditional DDoS attack scenario composed of the following sequential proceeding: Attacker finds the victims(HS and PS up to this step)→Attacker installs Master program to a host→Attacker installs Daemon program to other hosts(VE and D up to this step)→Attacker executes DDoS attack(Final attack up to this step).

- **Scenario step 1. Host and Port Scan:** An attacker scans the observable hosts, i.e., host A, B and C. By using Nmap[Nm1], the attacker tries to know whether hosts A, B and C are alive and what is the type of OS of the hosts(HS), and which port is open(PS).

- **Scenario step 2. Vulnerability Exploit and Infection:** Now, based on the results from HS and PS, the attacker installs an attack tool, i.e., DDoS Trin00 daemon, over the observable hosts. That is, by using Nessus[Ne1], the attacker gathers the vulnerability information of the hosts. And then, based on collected vulnerabilities, the attacker installs a DDoS tool for each host. Host A is used as Master and host B and C as Daemon.
- **Scenario step 3. Final attack:** The attacker executes DDoS attack from host A, B and C to the protective host S.

The results of the experiment are as follows. Now, we describe the result of the experiment based on the event logs of the SNOT and the event logs and the configuration informations of hosts.

We could calculate the value of infringement severity for HS and PS as follows.

- **Event logs related to HS**
  - SNORT log
    (*time) Auth.Alert IDS snort: [1:402:7]* ***ICMP Destination Unreachable Port Unreachable*** *[Classification: Misc activity] [Priority: 3]:* {*ICMP*} *host A → attacker*
- **Event logs related to PS**
  - SNORT log
    (time) Auth.Alert IDS snort: [122:3:0] ***(portscan) TCP Portsweep PROTO255*** *attacker → host A*
  - iplog[Ip1]
    (time) TCP: ***ssh connection attempt*** *from attacker:43116*
  - Open port information collected from host A in advance
    (time) ***22/tcp open ssh***
    111/tcp open rpcbind
    139/tcp open netbios

From the SNORT logs, HS and PS from attacker to host A were detected(intrusion attempt). Based on the event logs from hosts, it was verified for hosts to be really infringed because the scanned port to host A was really open at host A as shown in the event logs of hosts(possibility of real infringement). From these evidences for analysis, when wt(HS)=1, wt(PS)=2, wt(HS, 1)=0.4 and wt(PS, 1)=0.3 are given, v(HS, host A) can be obtained by $1*(0.4*1+\sum_{k=2}^{|C(1)|} wt(1,k)*0)$=0.4 and v(HS, host B) can be obtained by $2*(0.3*1+\sum_{k=2}^{|C(2)|} wt(2,k)*0)$=0.6. Each value is located between min.th.(HS or PS)(=0.1 or 0.2) and avg.th.(HS or PS)(=0.5 or 1). Thus, we collected the forensic evidences for stage HS and PS from host A. With a small time difference, we got the similar results for host B and C. Here, C(1) was given as $\{c_1(1), c_2(1)\}$ and C(2) was given as $\{c_1(2), c_2(2)\}$, where $c_1(1)$ was given as (Observable IP, IP scanned by attacker), $c_2(1)$ as (Observable subnet list, subnet list scanned by attacker), $c_1(2)$ as (Open port collected from each host in advance, Port detected by iplog for each host) and $c_2(2)$ as (Open port collected from each host in advance, Port detected by the SNORT for each host).

We could calculate the value of infringement severity for vulnerability exploit and infection to the hosts by using the following evidences for analysis.

- **Event logs related to VE**
  - SNORT log
    (time) Auth.Alert IDS snort: [1:3018:1] ***NETBIOS SMB NT Trans NT CREATE oversized Security Descriptor attempt*** *[Classification: Generic Protocol Command Decode] [Priority: 3]: TCP attacker: 2612 → host A:139*
- **Event logs related to infection**
  - SNORT log
    (time) Auth.Alert IDS snort:[1:232:5] ***DDOS Trin00 Daemon to Master *HELLO* message detected*** *[Classification: Attempted Denial of Service] [Priority: 2]: UDP attacker:1995 → host A:31335*

From the SNORT log, VE attempt from attacker to host A was detected(intrusion attempt) and from the vulnerability scanning of host A executed by using Nessus in advance, we detected 40 vulnerabilities with the importance *info*, 14 with the *medium* and 18 with the *high*, where the importance of vulnerability can be *info*, *medium* or *high*. By comparing with the SNORT log, it was verified that vulnerability of 139 port was one of 40 vulnerabilities with the importance *info* at host A(possibility of real infringement). From this evidence for analysis, when wt(VE)=3 and wt(VE, 2)=18/72 are given, v(VE, host A) can be obtained by $3 * (wt(VE, 1) * 0 + (18/72) * 1 \sum_{k=3}^{|C(3)|} wt(3, k) * 0)$=3/4. The value was located between min.th.(VE)(=0.3) and avg.th.(VE)(=1.5).

Related to infection, it was found that a new active process operates at host A which was connected to the daemon with strings found in DDoS tools(possibility of real infringement). And the specific strings which were exchanged among daemons were detected by IDS(intrusion attempt and possibility of real infringement). Based on these evidences for analysis, when wt(D,2,3)=0.3 and wt(D,3,1)=0.5 are given, v(D,2,host A) can be obtained by $wt(D, 2, 1) * 0 + wt(D, 2, 2) * 0 + 0.3 * 1$=0.3, and similarly v(D,3,host A) can be obtained by $0.5 * 1$=0.5. The value was located between min.th.(VE)(=0.3) and avg.th.(VE)(=5). Thus, we collected the forensic evidences for stage VE and infection from host A. With a small time difference, we got the similar results for host B and C.

Here, C(3) was given as $\{c_1(3), c_2(3), c_3(3)\}$ and C(4) was given as $\{c_1(4, 1), c_2(4, 1), c_1(4, 2), c_2(4, 2), c_3(4, 2), c_1(4, 3)\}$, where $c_1(3)$ was given as (Vulnerability information collected from each host(*high*), Vulnerability information scanned by attacker(*high*)), $c_2(3)$ as (Vulnerability information collected from each host(*medium*), Vulnerability information scanned by attacker(*medium*))m $c_3(3)$ as (Vulnerability information collected from each host (*info*), Vulnerability information scanned by attacker(*info*)), $c_1(4, 1)$ as (Open port collected from each host in advance, Open port collected from each host periodically), $c_2(4, 1)$ as (access information for files collected from each host in advance, access information for files collected from each host periodically), $c_1(4, 2)$ as (Active port collected from each host in advance, Active port collected from each host periodically), $c_2(4, 2)$ as (Active process collected from each host in advance, Active port collected from each host periodically), $c_3(4, 2)$ as (strings of executable

files, strings of daemon executing at each host) and $c_1(4,3)$ as (strings given and received among attack tools, strings detected by IDS, e.g., *HELLO* message).

Finally, we could calculate the value of infringement severity for DDoS attack as follows.

- **Event logs related to the final attack, i.e., DDoS**
  - SNORT log
    (time) Auth.Alert IDS snort: [1:2339:2] ***TFTP NULL command attempt [Classification: Potentially Bad Traffic]*** *[Priority: 2]: UDP host B:2029 → host S:69*

As shown in the SNORT log, DDoS attack from host B to host S was detected (intrusion attempt and possibility of real infringement). Based on the evidence for analysis, when wt(DDoS,1)=0.7 and wt(DDoS,2)=0.3 are given, v(DDoS,host S) can be obtained by $16 * (0.7 * 1 + 0)$=11.2 and was larger than avg.th.(DDoS)(=0.8). Also, attacks from host A and C were detected so that the sum of severity values until stage DDoS exceeded total.avg.th.(=16). It means that the real attack is affecting the operation of host S seriously. Here, C(5) was given as $\{c_1(5), c_2(5)\}$, where $c_1(5)$ was given (-, the Snort for bad traffic) and $c_2(5)$ as (port number of the specific DDoS daemon given from rand() function, port scanned from attacker for each host). Thus, we collected the forensic evidences from host S.

A Copy Protection Technique Using Multi-level Error Coding

Chen-Yin Liao, Jen-Wei Yeh, and Ming-Seng Kao

Department of Communication Engineering, National Chiao-Tung University, Hsinchu, 30050, Taiwan

Abstract. A novel copy protection scheme for optical disks is proposed. Three error mechanisms – error insertion, error correction and error propagation – are included in the proposed scheme, which lead to a sharp cutoff in the detection probability of an encryption key. This cutoff behavior could be employed to effectively prevent bit-by-bit copying of optical disks. The proposed scheme can be easily implemented in common players, being a simple and effective copy protection technique**.**

Keywords: copy protection, multi-level error coding, error-control coding.

1 Introduction

As digital data can be perfectly reproduced without degradation, copy protection of optical disks had attracted much research interest [1-4]. Most of the reported copy protection schemes aimed at encrypting the disk and/or the content, while the recorder was less considered. Because either physically-encrypted disks or software-encrypted contents (e.g. digital watermarking) cannot effectively prevent bit-by-bit copying, mass unauthorized copies can be made with low-cost recorders [5]. There were several schemes being designed against bit-by-bit copying. For example, the "disk wobble" technology proposed by Philips reshaped the grooves inside optical disks so as to prevent duplication of stored data [6]. The "optical fingerprinting" scheme designed by NEC prevented disk copying by modifying the reflectivity of some selected bits [7]. Although these schemes can prevent bit-by-bit copying via common recorders, their drawback is the requirement of special players equipped with sophisticated detection circuits, which limits their practical use.

Error control coding had been extensively used in communications to improve system performance [8]. Here we adopt it alternatively to prevent bit-by-bit copying of optical disks. To the authors' best knowledge, no similar approaches had been proposed before. In our approach, an encryption key is used to scramble the source content, and a multi-level error coding is subsequently designed to encode the encryption key. Next, some special bits called X-bits are introduced, being inserted into the encoded codeword block as potential errors. As will be clear later, wing to the multi-level error coding and the inserted X-bits, a sharp cutoff exists in the detection probability of the encryption key. This cutoff behavior sets a tight bound on the reflectivity of X-bits, thereby being able to effectively prevent piracy.

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296 , pp. 222 – 232, 2006.
© Springer-Verlag Berlin Heidelberg 2006

2 Multi-level Error Coding

In the proposed approach, we apply the multi-level error coding to encode an encryption key as the first step to protect unauthorized copy. The multi-level error coding is composed of L levels with each level consisting of the word-extension process and the error-control coding process to be described below.

2.1 Word Extension

In our scheme, a binary word consisting of v bits (v>1) is called a unit-word, being the basic unit in the signal processing. Let $a = (\alpha_1, \alpha_2, ..., \alpha_v)$ and $b = (\beta_1, \beta_2, ..., \beta_v)$ be unit-words, where $\alpha_i, \beta_i \in (0,1)$. The mod-2 addition of a and b is given as

$$a \oplus b = (\alpha_1 \oplus \beta_1, \alpha_2 \oplus \beta_2, ..., \alpha_v \oplus \beta_v), \quad (1)$$

where the symbol " $\oplus$ " denotes mod-2 addition..

Let $a_1, a_2, ..., a_m$ be m different unit-words having the following property:

$$a_1 \oplus a_2 \oplus .... \oplus a_m = (0,0,..,0), \quad (2)$$

Eq.(2) means that the mod-2 addition of $a_1, a_2, ..., a_m$ results in the all-zero unit-word. For a given m, it is easy to generate a set of $\{a_1, a_2, ..., a_m\}$ which satisfies (2).

Supposed that y is an arbitrary unit-word. We can extend y to be a codeword D composed of m unit-words, i.e. $D = \{d_1, d_2, ..., d_m\}$, where d_i is a unit-word. This is called the word-extension process, which will be performed as below.

Let $\{a_1, a_2, ..., a_m\}$ be a specific set of m unit-words satisfying (2) and m is an odd integer. The word-extension process is given as

$$d_i = y \oplus a_i, \qquad i = 1,2,..., m, \quad (3)$$

Using (3) we can easily obtain D for a given y. On the other hand, if D is given, y can be obtained via the reverse of word-extension as below:

$$y = d_1 \oplus d_2 \oplus .... \oplus d_m, \quad (4)$$

Since m is an odd number, it is not difficult to verify (4) by using (2) and (3). The codeword D obtained from (3) is named as the extended codeword (EC) of y.

2.2 Error Control Coding

As described before, a unit-word y can be extended to be a codeword D consisting of m unit-words. The codeword D can be further encoded by using forward-error-correcting (FEC) codes.

Reed-Solomon code (RS-code) is a well-known non-binary FEC code in which the basic unit in the coding process is a symbol consisting of several bits. Here RS-code with v-bit symbol is employed, i.e., a unit-word is a symbol in the RS-coding. We apply (s,m) RS-code to encode D to be a codeword E consisting of s unit-words. This

codeword E is called the FEC-coded codeword (FC). Thus, with the word-extension/error-control coding process, we can transfer a unit-word y to be an FC with inherent error-correcting capability.

2.3 Multi-level Error Coding

In the proposed copy protection technique, an encryption key (K) consisting of k unit-words is used to scramble the source content. This key will be protected from bit-by-bit copying via the following multi-level error coding as well as the X-bit coding to be introduced in Section 3.

In the beginning, a (n, k) RS-code with v-bit symbol is used to encode K to be a codeword C consisting of n unit-words, where $C = (c_1, c_2, ..., c_n)$ and c_i is a unit-word. The codeword C will be further encoded by the multi-level error coding.

In the 1st-level, each unit-word in C is first word-extended to be a 1st-level EC and then further encoded via (s,m) RS-code to obtain the corresponding 1st-level FC. As there are n unit-words in C, the total number of 1st-level FC's originated from C is $N_1 = n$, with each consisting of s unit-words.

Let E be one of the 1st-level FC's just obtained. In the 2nd-level, each unit-word in E is similarly encoded as before to generate the corresponding 2nd-level EC and 2nd-level FC. Because E is composed of s unit-words, there are s 2nd-level FC's generated from E. As there are N_1 1st-level FC's, the total number of 2nd-level FC's generated after the 2nd-level is $N_2 = N_1 \cdot s = n \cdot s$, each again consists of s unit-words.

The same process as described above will be carried out in all the L levels to accomplish the multi-level error coding. After the Lth-level, we will obtain N_L Lth-level FC's, where $N_L = n \cdot s^{L-1}$. Thus, with the multi-level error coding, we actually encode K to be a big codeword consisting of N_L Lth-level FC's, whereas each Lth-level FC consists of s unit-words and has the inherent error-correction capability. These Lth level FC's will be further encoded by X-bits before stored in the optical disk.

3 X-Bit Coding

After the multi-level error coding, we obtain N_L Lth-level FC's which will be further encoded by X-bits. The encoding of X-bits is the second step to accomplish the proposed copy protection scheme.

3.1 The X-Bit

Let A_0 and A_1 be the signal levels corresponding to binary 0 and 1 stored in the optical disk, respectively. Those bits with signal levels A_0 or A_1 are called normal bits. On the other hand, an X-bit has the signal level as

$$A_X = \frac{A_0 + A_1}{2} + \chi = \mu + \chi, \tag{5}$$

where μ is the mean of A_0 and A_1, while χ is a random variable accounting for the level fluctuation of X-bits during manufacture process. In practice, X-bit can be obtained by making its reflectivity close to the mean reflectivity of normal 0-bit and 1-bit.

In the proposed scheme, X-bits are contained in the N_L Lth-level FC's and stored in the lead-in sector of the disk. X-bits can be detected with the double-read process, i.e., the lead-in sector containing X-bits would be read twice. For a given bit, if outcomes of two reading processes both are 1, it is recognized as a 1-bit; if both are 0, it is taken as a 0-bit; if two outcomes are different, an X-bit is assumed. Note that only the lead-in sector has to be read twice, so the time required for the double-read process is little. Moreover, common players can be used to detect X-bits without difficulty.

Assume the random variable χ in (5) be uniformly distributed within the interval $[-\alpha, \alpha]$, where α denotes the maximum deviation of A_X with respect to μ. In this case, the error probability in detecting X-bit via the double-read process can be calculated as [9]

$$P_{e,X}(\alpha) = 1 - \frac{1}{2\alpha}\int_{-\alpha}^{\alpha} \operatorname{erfc}\left(\frac{\chi}{\sqrt{2}\sigma_n}\right) \cdot \left[1 - \frac{1}{2}\operatorname{erfc}\left(\frac{\chi}{\sqrt{2}\sigma_n}\right)\right] d\chi, \tag{6}$$

where erfc(·) denotes the complementary error function and σ_n^2 is the variance of noise encountered in the reading process.

Fig. 1 illustrates $P_{e,X}$ as a function of the normalized parameter $\alpha' = \alpha / \sigma_n$. We find that $P_{e,X}$ increases with α' and has the minimum value of 0.5 when $\alpha' = 0$. The result reveals that most X-bits cannot be successfully detected via the double-read process. This is the preferred property, since X-bits can be served as potential errors in the optical disk.

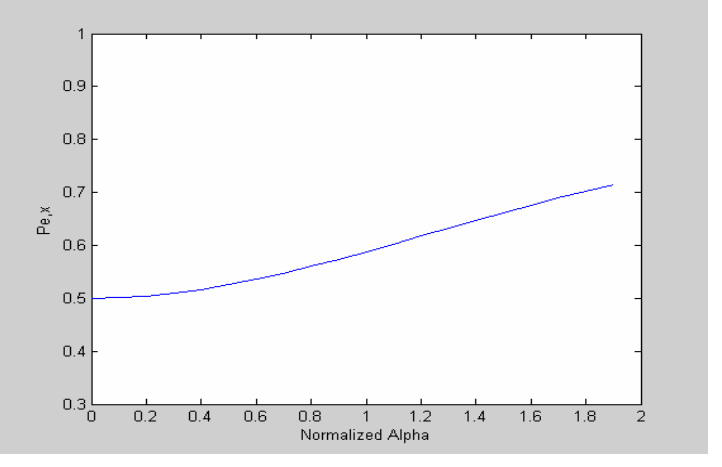


Fig. 1. $P_{e,X}$ as a function of the normalized parameter α'

3.2 X-Bit Coding Process

As described above, there are N_L Lth-level FC generated via the multi-level error coding. Let $Y = (y_0, y_1, ..., y_{s-1})$ be an Lth-level FC, where y_j is a unit-word. Within the s unit-words in Y, merely u of them will be encoded by X-bits. Those unit-words being X-bit encoded are called X-words, which will be served as potential errors in Y. We take $u = t + b$, where t is the error-correcting capability of the (s,m) RS-code, given as t=(s-m)/2, and b is an integer.

The u X-words within Y are assigned by a simple algorithm which uses the first unit-word y_0 as the indicator. Let $0 \le h \le 2^v$ be a decimal number corresponding to the binary content of y_0, then X-words in Y are indicated as

$$r_i = (i \cdot h) \mod s, \quad i=1,2,\ldots,u, \tag{7}$$

Eq.(7) means that $(y_{r_1}, y_{r_2}, ..., y_{r_u})$ within Y will be encoded as X-words. If $r_i = 0$, we take $r_i = 1$ to avoid y_0 being coded as an X-word. Moreover, if $j > i$ and $r_j = r_i$, then $r_j = r_i + 1$ is assumed. Thus we can always have u X-words in Y.

Let $y = (b_1, b_2, ..., b_v)$, $b_i \in \{0,1\}$, be a unit-word in Y to be encoded as a X-word $\xi = (e_1, e_2, ..., e_v)$, $e_i \in \{0,1,X\}$. The coding can be accomplished by a mapping table such that there is a one-to-one mapping between y and ξ. Let β denote the number of X-bits in ξ. For a given v, if the following inequality

$$C_\beta^v \cdot 2^{v-\beta} > 2^v \quad \Rightarrow \quad C_\beta^v > 2^\beta, \tag{8}$$

is satisfied, we can always design a mapping table which has one-to-one mapping between y and ξ (details of the mapping table is omitted here). Thus, with X-bit coding, $(y_{r_1}, y_{r_2}, ..., y_{r_u})$ in Y will become X-words.

After X-bit coding, each Lth level FC will contain u X-words plus $s-u$ normal unit-words, while an X-word contains β X-bits and $v-\beta$ normal bits. All the N_L Lth-level FC's will be stored in the lead-in sector of the optical disk. As the error probability in detecting X-bit is high, the error probability in detecting X-words is high as well. However, due to the multi-level error coding, it is still possible to recover the encryption key through RS-decoding.

4 Decoding Process

4.1 X-Bit Decoding

As described above, there are N_L Lth-level FC's stored in the lead-in sector of the optical disk. This disk can be read by common players in which a special decoding

algorithm was installed. When the disk is played, the decoding algorithm will guide the player to read the lead-in sector twice in order to decode the encryption key.

We begin with the decoding of X-bits. Let $W = (w_0, w_1, ..., w_{s-1})$ be an Lth-level FC consisting of s unit-words with u of them being X-words. In the first reading process, all the s unit-words in W are detected, including those X-words. For simplicity, we assume all the normal bits be correctly detected without error, while detail performance analysis will be given in the next section.

After the first reading process, the locations of X-words within W are indicated by the first unit-word w_0 via (7). Although the locations of X-words are known, the β X-bits inside an X-word must be detected via the second reading process. Let w_j be one of the X-words in W to be decoded. If all the β X-bits in w_j are successfully detected in the second reading process, the original content of w_j will be obtained via the mapping table mentioned in Sec. 3.2. If one or more than one X-bits are not detected, the decoding algorithm will assume a random decoded word to w_j (with the probability of $1/2^v$ to simulate the original content of w_j).

After X-bit decoding, W became a decoded Lth-level FC within which some of the s unit-words inside W might be erroneous. In particular, most of the errors are coming from u X-words but not normal words. However, it is still possible to recover the corresponding Lth-level EC, if the number of erroneous words in **W** is within the error correcting capability of the RS-code.

4.2 Multi-level Decoding

Let W' be an X-bit decoded Lth-level FC, which may contain some errors within the s unit-words. In the Lth-level decoding, we perform RS-decoding to recover the corresponding Lth-level EC. If the number of erroneous words in W' is within the error correcting capability of the (s,m) RS-code, the decoded Lth-level EC will be correct; otherwise, it will be erroneous. Thus, after the Lth-level decoding, we obtain N_L Lth-level EC, with some of them might be erroneous.

In the (L-1)th-level decoding, each unit-word in an (L-1)th-level FC can be obtained from the corresponding Lth-level EC via the reverse of word-extension process given in (4), and then an (L-1)th-level FC can be obtained from s Lth-level EC's. Thus N_{L-1} (L-1)th-level FC's will be obtained from N_L Lth-level EC's. Next, each (L-1)th-level FC is RS-decoded to obtain the corresponding (L-1)th-level EC. Again if the number of erroneous unit-words in an (L-1)th-level FC is within the error correcting capability of RS-code, the decoded (L-1)th-level EC will be correct. Otherwise, it will be erroneous.

The same process described above can be carried out until the 1st-level to obtain n 1st-level EC's. Then, using the n 1st-level EC's, the codeword C can be regained via the reverse of word-extension. Finally, the encryption key (K) can be recovered via the (n,k) RS-decoding. This completes the decoding process.

5 Performance Analysis

Let P_e denote the error probability of detecting a normal bit, then the probability of correctly detecting a normal unit-word is given as $P_n = (1 - P_e)^v$. Meanwhile, as there are β X-bits and $v - \beta$ normal bits in an X-word, the probability of correctly decoding an X-word is approximately expressed as

$$P_d \approx (1 - P_{e,X})^{\beta} (1 - P_e)^{v-\beta} + [1 - (1 - P_{e,X})^{\beta}] \cdot \frac{1}{2^v}, \tag{9}$$

where the first term in the right-hand side of (9) is the probability that all the X-bits and normal bits in the X-word of concern are correctly detected, while the second term is the probability that at least one X-bit is not detected, but the random decoded word assumed by the decoding algorithm happens to simulate the correct content.

If W is an Lth-level FC consisting of u X-words and s-u normal words, the probability that exactly r erroneous words occurring in W after X-bit decoding is given as

$$P_L(r) = \sum_{i=0}^{r} [C_i^u (1 - P_d)^i P_d^{u-i}] \cdot [C_{r-i}^{s-u} (1 - P_n)^{r-i} P_n^{s-u-r+i}], \tag{10}$$

Let Y be the Lth-level EC obtained from W through RS-decoding and t be the maximum number of correctable errors in the (s,m) RS-code. If $r \le t$, then Y can be correctly decoded. Hence the probability of correctly decoding Y is written as

$$P_L = \sum_{r=0}^{t} P_L(r), \tag{11}$$

Next, we analyze the performance of the (L-1)th-level. Assume $Z = (z_1, z_2, ..., z_S)$ be an (L-1)th-level FC consisting of s unit-words. In the decoding process, each z_i is obtained from the corresponding Lth-level EC via the reverse of word-extension process indicated in (4). Thus the correct probability of z_i is equal to that of the corresponding Lth-level EC, i.e. P_L. Therefore, the probability that exactly r erroneous words occurring in Z is written as

$$P_{L-1}(r) = C_r^s (1 - P_L)^r P_L^{s-r}, \tag{12}$$

If $r \le t$, the (L-1)th-level EC corresponding to Z can be correctly decoded through RS-decoding. Thus the correct probability of decoding an (L-1)th-level EC is calculated as

$$P_{L-1} = \sum_{r=0}^{t} P_{L-1}(r) = \sum_{r=0}^{t} C_r^s (1 - P_L)^r P_L^{s-r}, \tag{13}$$

The relationship of (13) can be similarly applied to the other levels. In general, we have

$$P_{j-1} = \sum_{r=0}^{t} C_r^s (1-P_j)^r P_j^{s-r}, \quad j=2,3, \ldots L, \tag{14}$$

where P_j and P_{j-1} are probabilities of correctly decoding the jth-level EC and the (j-1)th-level EC, respectively.

Fig. 2 shows P_j versus α' at different decoding levels with L=4. We find that all P_j's intercept at a special point, denoted as (α'_C, P_{cutoff}), where P_{cutoff} is defined as the cutoff probability. Also notice that the curve of P_1 (the correct probability of decoding a 1st-level EC) approaches a rectangular-shape with a sharp cutoff at $\alpha' = \alpha'_C$. In the region $\alpha' < \alpha'_C$, we have $P_1 \to 1$; while in the region $\alpha' > \alpha'_C$, we have $P_1 \to 0$.

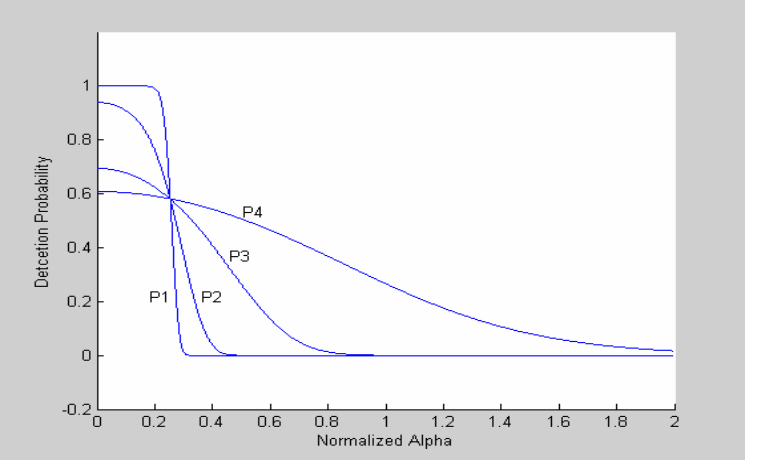


Fig. 2. P_j versus α' at different decoding levels, with L=4, v=5, $\beta = 1$, b=13, s=31, m=5 and $P_e = 10^{-5}$

The results of Fig. 2 can be explained as follows. At $\alpha' = \alpha'_C$, we have $P_j = P_{j-1} = P_{cutoff}$, indicating that the decoding process does not affect the performance. In the region $\alpha' < \alpha'_C$, we have $P_{j-1} > P_j$, revealing that the mean number of erroneous words in the (j-1)th-level EC is decreased so that P_{j-1} increases. In contrast, we have $P_{j-1} < P_j$ in the region $\alpha' > \alpha'_C$, indicating that the mean number of erroneous words in the (j-1)th-level EC is increased, i.e., error propagation occurs in this region. Owing to the strong error-correcting capability of RS-codes in the region $\alpha' < \alpha'_C$ and the fast growing error propagation effect in the region $\alpha' > \alpha'_C$, eventually we obtain the special curve of P_1.

After the 1st-level decoding, we have n 1st-level EC's with correct probability P_1. Then the original codeword C can be obtained through the reverse of word-extension process, and the encryption key K is recovered through the (n,k) RS-decoding. As the maximum number of allowable errors in an (n,k) RS-code is $(n-k)/2$, the probability of correctly recovering K is expressed as

$$P_{key} = \sum_{r=0}^{(n-k)/2} C_r^n (1-P_1)^r P_1^{n-r}, \quad (15)$$

Fig. 3 illustrates the relationship between P_{key} and α' for different values of b. Again all the curves approach the rectangular shape with a sharp cutoff. In each curve, if α' is below the cutoff level, K is guaranteed to be correctly decoded since $P_{key} \to 1$. Otherwise, it is almost impossible to decode K since $P_{key} \to 0$. We also find that the parameter b has significant affect on the cutoff level, which can be used to achieve the desired value of α'_C.

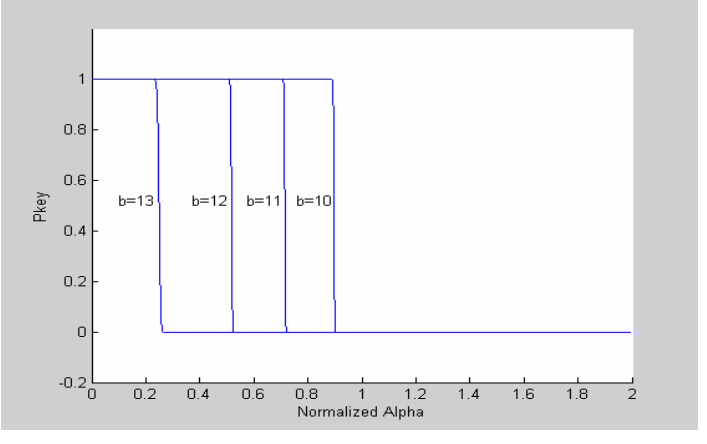


Fig. 3. P_{key} versus α' for different RS-codes, where L=3, v=5, $\beta = 1$, s=31, m=5, n=31, k=11 and $P_e = 10^{-5}$

In Fig. 3, we have $\alpha'_C \approx 0.22$ for the case of $b = 13$, indicating that K can be successfully recovered only if $\alpha' < 0.22$, i.e. $\alpha < 0.22\sigma_n$. Recall that α is the maximum deviation of A_X with respect to μ while σ_n is the standard deviation of noise encountered in the reading process, the above requirement sets a tight bound on the signal level of X-bits since σ_n is usually very small. For authorized disks pressed by well-equipped factories, this specification can be met by special equipments with ultra-high precision in making X-bits, i.e., the reflectivity of X-bits should be within the tight specification imposed by $\alpha < 0.22\sigma_n$. In contrast, it will be very difficult to reproduce X-bits without such equipments. If these ultra-high precision equipments

are offered by few companies and their access is well controlled, then bit-by-bit copying will be effectively prevented.

6 Discussion

The cutoff phenomenon illustrated above is accomplished by the multi-level error coding which includes three error mechanisms, i.e., error insertion, error correction and error propagation. The introduction of X-bits is to artificially insert errors into the disk, being used to prevent bit-by-bit copying. The error correction capability provided by RS-codes is able to recover the encryption key if X-bits are precisely made. Meanwhile, the phenomenon of error propagation is implicitly present in the decoding process. The multi-level error coding results in a tight bound on the reflectivity of X-bits, being a severe obstacle for bit-by-bit copying.

In the proposed scheme, the locations of all X-bits in the disk can be known via repeated read-detect process and then the original content of the encryption key can be known. However, knowing the key is of no use to make a copied disk be successfully played. As the decoding algorithm installed in the firmware of players cannot be modified, a copied disk without X-bits will be randomly decoded so that the source content will be lost. Using the proposed scheme, casual piracy via common recorders is definitely prohibited since X-bits cannot be reproduced. On the other hand, professional piracy will be prevented as well, if the access of those ultra-high precision equipments in making X-bits is well controlled. Moreover, as the double-read process necessary to detect X-bits can be easily implemented in common players, the proposed scheme is a feasible and practical copy protection technique.

7 Conclusions

A copy protection scheme based on multi-level error coding is proposed. We introduce X-bits into the optical disk serving as potential errors. We further design the multi-level error coding scheme cooperated with X-bit coding to achieve a sharp cutoff in the detection probability of encryption key. This cutoff behavior sets a tight bound on the reflectivity of X-bits, thereby making bit-by-bit copying of optical disks very difficult. The multi-level error coding consists of three error mechanisms, being an effective approach to prevent unauthorized copying. In the future, we will look for other possible codes and improve the coding algorithm to further control the cutoff level so as to realize a robust copy protection technique.

References

1. J. A. Bloom, I. J. Cox, T. Kalker, J.-P. M. G. Linnartz, M. L. Miller, C. B.S. Traw, “Copy protection for DVD video,” *Proceedings of the IEEE,* vol. 87, (1999) 1267 – 1276.
2. M. Maes, T. Kalker, J.-P. M. G. Linnartz, J. Talstra, F. G. Depovere, J. Haitsma, “Digital watermarking for DVD video copy protection,” *IEEE Signal Processing Magazine*, vol. 17, (2000) 47–57.

3. A. V. Spesivtsev, A. J. Krutjakov, V. V. Seregin, V. A. Sidorov, V. A. Wegner, "Software copy protection systems: structure, analysis, attacks," *1992 International Carnahan Conference on Security Technology*, (1992) 179 – 182.
4. H. Morito, M. Roe, E. S. Lee, "Digital copy protection scheme using recording medium identifier," in *Proc. IEEE International . Workshops on Parallel Processing,* (1999) 174–178.
5. D. S. Wallach, "Copy protection technology is doomed," *Computer,* vol. 34, (2001) 48-49.
6. M. J. Geier, "Lights, camera, controls! " *Computer,* vol. 40, (2003) 28-31.
7. N. R. Potlapally, "Optical fingerprinting to protect data: a proposal," *Computer,* vol. 35, pp.23-28, Mar. 2002
8. S. Lin and D. J. Costello, Error Control Coding, ,Pearson Prentice Hall,: 2nd ed., 2003, ch.7.
9. S. Simon, Communication Systems, John Wiley & Sons, 4th ed., 2001, ch.4

Digital Rights Management with Right Delegation for Home Networks

Heeyoul Kim[1], Younho Lee[1], Byungchun Chung[1], Hyunsoo Yoon[1], Jaewon Lee[2], and KyungIm Jung[2]

[1] Network and Security Laboratory
Division of Computer Science
Korea Advanced Institute of Science and Technology (KAIST)
{hykim, yhlee, bcchung, hyoon}@nslab.kaist.ac.kr
[2] Software Laboratories
Samsung Electronics CO., LTD., Suwon, Korea
{jaewon613.lee, kyung}@samsung.com

Abstract. The purpose of digital rights management (DRM) is to protect the copyrights of content providers and to enable only designated user to access digital contents. For a user to share the contents among all his devices in the home network, several domain-based approaches that group multiple devices into a domain have been proposed. In these approaches, however, each device in a domain has equivalent rights on all contents although certain contents require an access control between the devices. In this paper, a new DRM system for home networks is presented. This system enables access control on the contents by a right delegation strategy with proxy certificates. Moreover, it also provides additional functionalities, including restricted sharing and temporal sharing of contents, which are necessary for ordinary scenarios in home networks.

Keywords: DRM, domain, right delegation, proxy certificate.

1 Introduction

Digital rights management (DRM) is the term referring to any of several technologies used to enforce pre-defined policies controlling access to music, movies, or other digital data. The main purpose of DRM is to protect the copyrights of digital content providers and to enable distribution and access control of the content. Since the advent of personal computers, digital contents have become easy to copy an unlimited number of times without any degradation in the quality of subsequent copies. The popularity of the Internet and file sharing tools have made the distribution of copyrighted digital media files simple. Therefore, a novel DRM system is necessary for the benefit of content providers.

One of the first and most widely contested DRM systems was the Content Scrambling System (CSS) used to encode DVD movie files [2, 3]. This system was developed by the DVD Consortium as a tool to influence hardware manufacturers to produce only systems which did not include certain features. By releasing the encryption key for CSS only to hardware manufacturers who agreed not to

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 233–245, 2006.
© Springer-Verlag Berlin Heidelberg 2006

include features such as digital-out, which would allow a movie to be copied easily, the DVD Consortium was essentially able to dictate hardware policy for the DVD industry.

From the consumers' point of view, they have a tendency to dislike complex and confused restrictions. This aptitude obstructs the growth of DRM market. Moreover, consumer's right of using legally acquired contents may be harmed by arbitrary limitations. Especially, if someone legally purchases a digital content, he wants to play the content freely on any of his multiple devices such as PC, PDA, and MP3 Player. If he should purchase the content per each device separately, he not only feels inconvenience to using the content but also may spend extra money.

The concept of Authorized Domain (AD) has been presented to resolve such problems [8, 10, 12]. The devices of a consumer organize a group, named authorized domain, and the access right on the content is given to the group instead of each device. Once he purchases the content, he can play it on any of devices registered in the domain. Especially, Popescu et al. [11] present a security architecture for AD with compliance checking protocol which is targeted on home network environments. In these systems, a domain is considered as an entity, and contents are shared with all devices in a domain. Thus, each device has equivalent access rights on the contents, just if it is registered. However, certain contents such as adult contents, which are not allowed to children, are to be managed with access control.

In this paper, we present a DRM system for home networks based on the right delegation strategy that controls the access rights of each device. A domain manager stands for all devices within its home network, and it obtains access rights on all digital contents for the home network. When a proper device wants to access a content, corresponding access right is delegated from the manager to the device. To check the validity of right delegation, a proxy certificate is utilized in the system. Moreover, the system provides additional functionalities including restricted content sharing and temporal content sharing.

The rest of the paper is organized as follows. In Section 2, previous DRM systems are reviewed and the motivation is described. Also, the functional requirements for home networks are described. In Section 3, a right delegation strategy with proxy certificate is explained. In Section 4, a DRM system to which the delegation strategy is applied is provided. In Section 5, two revocation mechanisms of proxy certificates are discussed. In Section 6, we show that the provided DRM system satisfies the functional requirements. Finally, some conclusions are made in Section 7.

2 Related Work and Motivation

Until the Authorized Domain (AD)-based DRM approaches are proposed [6, 8, 10, 12, 14, 16], there have been few approaches to address the DRM in home networks. The AD-based DRM technology enables all devices in the same domain to use and share the DRM contents freely. A domain can be either a home

network, a personalized network, or any network which has several rendering devices such as PC, MP3 Player, PDA, and Video Player.

To the best of our knowledge, the most recent work considering DRM in home networks is Popescu et. al.'s approach [11]. In their work, there exists a Domain Manager (DM) per each home network which has a role of the registration and revocation of compliant devices in AD. DM also manages the keys to be used by each compliant device for the authentication and encryption of the DRM content.

xCP [16], proposed by IBM, is an AD-based DRM architecture that employs the broadcast encryption for secure content distribution and membership management. Due to the use of broadcast encryption, xCP needs expensive cost to revoke the members in a domain because the size of the new Media Key Block, which is used to exclude the keys of the revoked devices, is relatively large.

Other recent approaches, such as Windows Media DRM and IBM Electronic Media Management System, neither target on the DRM in home networks nor provide a domain-based management functionality [13, 1, 9, 4, 21, 7].

OMA (Open Mobile Alliance) DRM standard [10] also provides AD functionality. In OMA DRM, the centralized Rights Issuer (RI) manages all domains respectively. All compliant devices should negotiate with the RI to be registered in a domain. This centralized AD management gives much burden on the RI. Moreover, if a compliant device cannot be connected with the RI either directly or indirectly via a proxy device, it cannot be registered to or revoked from the domain. Since RI can acquire the information about all registered compliant devices in each domain, the privacy of each AD can be disturbed.

In the first subsection, we describe an overview of OMA DRM system because our work is based on the OMA DRM system. In the second subsection, the motivation of our work is provided. Finally, the third subsection provides the functional requirements that the proposed DRM system should meet.

2.1 OMA DRM System

The purpose of OMA DRM system is to enable the distribution and consumption of the DRM content in a controlled manner. The content is distributed and consumed on authenticated devices according to the access rights prescribed by the content owners. This system provides mechanisms for secure authentication of devices, secure packaging, and secure transfer of both DRM content and access rights. The system relies on the existence of a Public Key Infrastructure (PKI). The following functional entities consist of the DRM system:

- **Content Provider (CP):** The content provider offers digital contents such as movies and audio contents. The contents are transferred not to the users directly, but to the content issuer.

- **Content Issuer (CI):** The content issuer has a responsibility for transferring the contents to the DRM agents. For secure transferring, the content is re-packaged so that only the purposed agent can obtain the content.

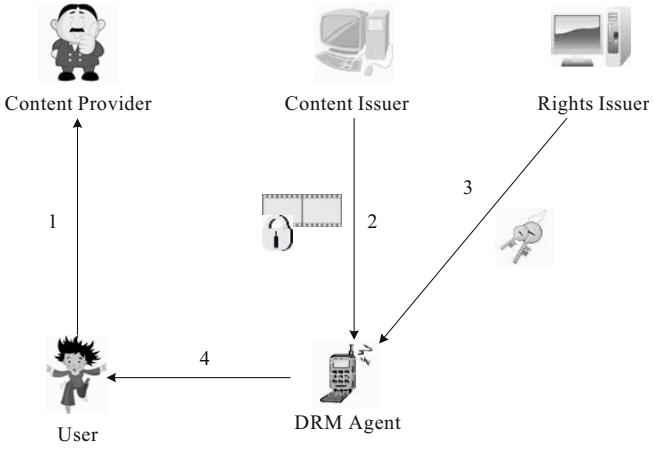


Fig. 1. Overview of OMA DRM system

- **Rights Issuer (RI):** The rights issuer assigns access rights to a DRM content, and generates a Rights Object (RO) for the DRM agent that legally purchases the content. The RO governs how the content can be used by the DRM agent.

- **DRM Agent:** The DRM agent is a trusted entity in a device. The agent is responsible for enforcing access rights specified in the RO.

Fig. 1 shows the overview of OMA DRM system, which works as follows:

1. A user pays to the content provider for a DRM content he wants.
2. The content issuer transfers a protected content in which the content is encrypted with a Content Encryption Key (CEK).
3. The rights issuer generates an RO which contains a CEK. Then the RO is transferred securely to the user's DRM agent.
4. Then DRM agent obtains the DRM content by decrypting the protected content with the CEK in RO. It also checks the access rights specified in the RO. Then, the agent plays the content.

2.2 Motivation

To the best of our knowledge, there are no previous AD-based approaches that give each device in a domain a separated access right for DRM contents; in other words, all compliant devices in the same domain have the same access right.[1]

[1] We only deals with the domain based rights object. Actually, if a device gets a rights object only for itself, the privacy problem is not happened. However, the content which corresponds to the rights object cannot be shared or redistributed without reissuing the rights object by RI; it is not scalable and burdensome to RI.

This causes a number of privacy and security problems. For example, suppose that a user A registers a friend B's device in her home network for B to listen her favorite music.[2] In this case, if the previous approaches are applied, B can access A's all DRM contents in her home network as well as the favorite music file that A is allowed for B to listen. Thus, B can see any secret DRM contents that A does not want to share.

Therefore, in order to remove these potential security and privacy problems, a novel method that can control the access rights of each device per DRM content should be provided. Our main objective in this work is to resolve this problem.

2.3 Functional Requirements

In addition to addressing the drawback described in the previous subsection, the proposed DRM system aims to provide the following functional requirements.

- **Local domain licensing:** The RI can delegate the right of local domain licensing to the Local Domain Manager (LDM).
- **Contents local sharing:** The contents which belong to a device can be shared with other devices in the same local domain.
- **Transferring RO:** The RO which belongs to a device D1 can be transferred to another device D2. After transferring, D1 should not be able to use the transferred RO.
- **Restricted Sharing:** It should be provided how to share a content with only designated devices in the local domain.
- **Temporal Sharing:** A device D1 can temporally share a content with the device D2 placed near D1, although it is not registered in the local domain.
- **Revocation:** RI can revoke the right of local domain licensing that is given to the LDM.

3 Right Delegation by Proxy Certificate

In this section, we provide a right delegation method with proxy certificate. The first subsection describes an overview of proxy certificate, and the second subsection describes how to apply the proxy certificate to the localized right delegation for DRM.

[2] We assume that the temporal access right of digital contents can be provided to unlicensed party. This temporal access right can be controlled with the various ways such as proximity test and access time limitation.

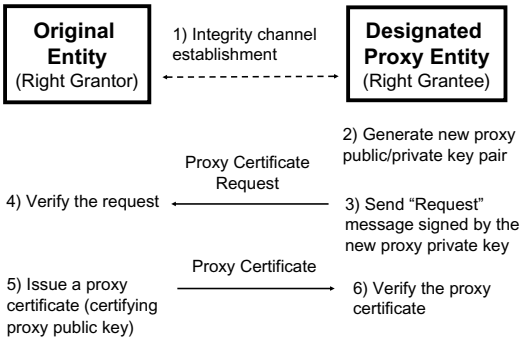


Fig. 2. Delegation protocol in X.509 proxy certificate profile

3.1 Overview of Proxy Certificate

The proxy certificate profile is an extension of ITU-T X.509 Public Key Infrastructure [18, 19]. It enables an entity (Right Grantor) to delegate all or a part of its right to another entity (Right Grantee). Currently, it is implemented for the GSI (Globus Security Infrastructure), which is included in the Globus Project [20], the world wide research project for computational grids. A proxy certificate includes the issuer name, subject name, description of delegated access rights, and the proxy public key which is used to exercise the delegated right. It is signed by the Right Grantor to prevent malicious modification.

The delegation protocol in X.509 proxy certificate is motivated from Neuman's work [15] which proposed the certificate-based right delegation method between objects. The detailed protocol description is shown in Fig. 2. After the protocol, the Right Grantee keeps the proxy certificate which has following properties:

1. Dynamic: The delegation can be established and canceled by a right grantor or grantee dynamically.
2. Light-weight: The delegation procedure needs just small computation and communication resources.
3. Restricted delegation: It is possible that the right grantor delegates only what the right grantee needs. Delegation of excessive rights causes more damage if the right grantee is exposed to an outside attacker.
4. Repeated delegation: It can be happened that the delegated right can be re-delegated to another entity.

3.2 Application to DRM

The proxy certificate in our work gives a domain manager an ability to locally license the devices in a domain to render the DRM contents. To exercise the localized licensing, the manager first gets a proxy certificate that represents the licensing ability is delegated from the RI. The RI can put restrictions on the delegated licensing right by describing them in the issued proxy certificate.

After receiving the proxy certificate, the manager can grant a delegated access right on the DRM content to an individual device. As the domain manager can give various access rights to each device, the access control functionality for the devices in the same domain is possible. With the granted access right and the proxy certificate, each device in the domain can render the DRM content corresponding to the right after verifying them.

4 DRM System for Home Networks

4.1 Overview

In this section, we propose a new DRM system for local domain management in home network environments. A home network is a good example of a localized domain. However, our system is not restricted to only home network environments.

A home network interconnects electronic products/systems such as PCs, mobile phones, digital audio/video, or digital TV, enabling remote access to and control of those products/systems, and any available content such as music, video, or data [17]. The typical home network consists of a home gateway that connects inside home network to the outside public network, in-home intranets such as phone-line, power-line, or wireless network, and home devices that involve home networking facilities.

As we pointed out previously, the concept of AD-based DRM is insufficient for the local domain environments such as home networks. Therefore, we propose a new concept of "local domain management with delegation". To support this delegation concept, a new functional entity, namely Local Domain Manager (LDM), is required. The LDM is delegated the rights of issuing Delegated RO (DRO) from the RI, so that it should be a trusted entity. The LDM also manages the membership of devices within its local domain.

The LDM should be secure and reliable, and it also should have an "always-on" property. Moreover, it should have reasonable computing power because the generation of DRO requires time-consuming public-key operations. The home gateway satisfies these requirements of the LDM. Therefore, we consider that a home gateway plays the role of an LDM, which is shown in Fig. 3.

In Fig. 3, we show a flow of a DRM content from the CI to the DRM agent of a device in a home network. When a user pays to the CP for a DRM content he wants, the CI transfers a protected content with a legal RO to the DRM agent. This content should be also played in other devices that are members of home network. To support this capability, the RI delegates the right of issuing DROs within the limits of home network use. For this, the RI transfers a Local Domain RO (LDRO) for the purchased content, which is only accessible to the LDM, and a proxy certificate together. When a device in the home network requests an RO for that content, the LDM generates a DRO, which is only accessible to the DRM agent in that device, and transfers both this DRO and the related proxy certificate. The DRM agent verifies the legality of the proxy certificate

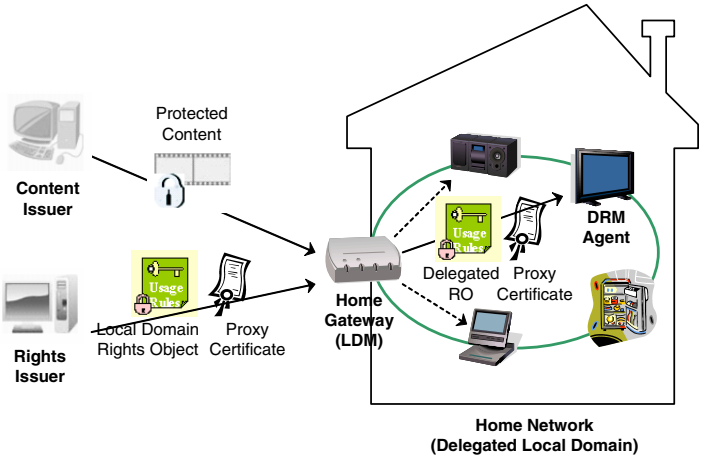


Fig. 3. DRM functional architecture for delegated local domain

and the satisfaction of specified constraints described in the DRO. When all requirements are satisfied, the DRM agent plays the content.

In the following subsections, we present more detailed processes of issuing proxy certificate by an RI, issuing DRO by an LDM, and verifying both DRO and proxy certificate by a device.

4.2 Issuing Proxy Certificate for RO

Let the RI have a public-private key pair (PK_{RI}, SK_{RI}) and its certificate $Cert_{RI}$. Let the LDM also have a public-private key pair (PK_{LDM}, SK_{LDM}) and its certificate $Cert_{LDM}$.

After a legitimate payment process, the LDM obtains a protected content which is encrypted with a CEK from the content issuer. At this point of time, he has no access right on the content, although he holds the protected content, because he does not hold a corresponding RO.

The RI issues two kinds of credentials about the content: a rights object associated with the content, and a proxy certificate for LDM. The rights object specifies that the LDM has a right to use the content. The proxy certificate specifies that the LDM can issue DROs for domain devices of his own. The detailed protocol is described as follows.

1. RI $\leftrightarrow$ LDM
   The RI and LDM establish a mutual authentication protocol which is provided in X.509 standard.
2. LDM $\rightarrow$ RI : $\{PK_{PI}\}_{SK_{PI}}$
   The LDM generates a public-private key pair (PK_{PI}, SK_{PI}) for proxy issuing DROs. The public key PK_{PI} is signed by the private key SK_{PI} and sent to the RI.

3. RI → LDM : $\{LDRO\}_{SK_{RI}}, \{Cert_{PI}\}_{SK_{RI}}$
   The RI generates an LDRO for the content. The LDRO specifies permissions and constraints associated with the usage of the content. The CEK is encrypted with the public key PK_{LDM}, and it is also contained by the LDRO. The RI checks the validity of the signature $\{PK_{PI}\}_{SK_{PI}}$ with PK_{PI}. Then, the RI issues a corresponding proxy certificate $Cert_{PI}$ including PK_{PI}. The certificate specifies permissions and constraints to issue DROs associated with the LDRO. The LDRO and $Cert_{PI}$ are signed by SK_{RI} respectively, and sent to the LDM. From the LDRO, the LDM obtains the CEK and can access the protected content. Then he verifies the proxy certificate.

Fig. 4 shows an example of an LDRO and corresponding proxy certificate. The content identifier field in both the LDRO and certificate binds them to the associated DRM content.

Local domain RO
Issuer : RI Subject : LDM Content Identifier : 00000001 Validity period : 06.3.1~07.2.28 Usage count : 10 Encryption key : $\{CEK\}_{PK_{LDM}}$
Signature by SK_{RI}

Proxy Certificate
Issuer : RI Subject : LDM Content Identifier : 00000001 Public key : PK_{PI} Validity period : 06.3.1~06.3.31 Delegation count : 3
Signature by SK_{RI}

Fig. 4. Issuing local domain RO and proxy certificate

4.3 Issuing Delegated RO by LDM

Suppose that a user wants to play the purchased content with one of his devices in the home network. The device D has a public-private key pair (PK_D, SK_D) and a certificate binding the public key with the device. The protected content is provided from the LDM or from other devices by the super-distribution. To play the content, D should acquire the access right on the content.

After the LDM and D authenticate each other with certificates, the LDM issues a DRO (e.g., in Fig. 5) to the device by the following steps:

1. The LDM obtains the CEK from the corresponding LDRO by decrypting it with SK_{LDM}.
2. The CEK is encrypted with D's public key PK_D. Then the ciphertext is contained in the DRO.
3. The LDM describes D's delegated access rights in the DRO within the scope of his own rights.
4. The DRO is signed by the proxy private key SK_{PI}. Then the DRO is sent to the device with the proxy certificate issued by RI.

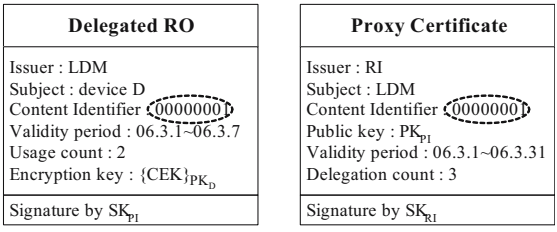


Fig. 5. Issuing delegated RO for a device

4.4 Verifying DRO and Proxy Certificate

A device in home network obtains a protected content, a DRO to the content, and a proxy certificate by the above steps. Let assume that the DRM agent in the device has a list of trusted RIs' certificates. Before playing the content, the agent checks the validity of both the DRO and the proxy certificate by the following steps:

1. The agent verifies the signature in the proxy certificate with PK_{RI} in $Cert_{RI}$. It also checks the validity period and recognizes delegated access rights specified in the proxy certificate. Then, it obtains PK_{PI}.
2. The agent checks whether the content identifier in the DRO matches the content identifier in the proxy certificate. It also checks the validity period in the DRO. With PK_{PI}, it also verifies the signature in the DRO. Then, it obtains CEK by decrypting $\{CEK\}_{PK_D}$ with SK_D.
3. The agent checks whether the received access rights and the validity period in the DRO are within the scope of the proxy certificate. Then, it decrypts the protected content with CEK, and plays the content.

For example, Fig. 5 shows that now the device D obtains the capability to play the content, whose identifier is 00000001, at most twice during one week.

5 Revocation Mechanisms

To provide the revocation functionality that is described in Section 2.3, the proposed DRM system presents two revocation methods: the proactive method and the reactive method.

5.1 Proactive Method

In this approach, the RI issues the proxy certificate to the LDM with only a short validity period. If the period is passed and the proxy certificate is expired, the LDM requests the RI to renew the proxy certificate. If the requested LDM is known as a compromised one, the RI can revoke the LDM by not renewing the proxy certificate. Otherwise, the RI renews the proxy certificate of the LDM and sends the renewed proxy certificate to the LDM.

From the view of the revocation, this approach is very efficient as the certificate revocation list is not needed. However, since the LDM frequently accesses the RI to renew its proxy certificate, a reliable communication channel between the LDM and the RI is required. Moreover, if a device is out of the home network but is preserving its local domain membership, the device can play the DRM content only until the current proxy certificate is valid.

5.2 Reactive Method

In this approach, the RI periodically issues the proxy certificate revocation list (PCRL) which is signed by the RI, and sends it to each LDM. Before rendering a new DRM content, an individual device requests the most recent PCRL. The LDM is enforced to send the PCRL to the device because it is regarded as a compromised one if it refuses to send the recent PCRL, or sends old PCRL.

This approach gives a heavy PCRL management cost to the RI. However, it is more flexible than the proactive method: although the device is out of its home network, it can receive a PCRL by the nearest LDM because it can verify the validity of the PCRL by checking the signature of the RI in PCRL.

6 Meeting the Functional Requirements

It is described how the proposed DRM system meets the functional requirements in Section 2.3. Additionally, the comparison results between the proposed system and the previous works is shown in Table 1.

- **Local domain licensing:** By issuing the proxy certificate to the LDM, the RI can delegate the local domain licensing rights to the LDM. Since the only valid LDMs that have valid proxy certificates can exercise the local domain licensing, the proposed system meets this requirement.

- **Contents local sharing:** As the LDM can issue a DRO with the proxy certificate that has been previously issued by the RI, all devices in the same local domain can acquire the right of rendering the DRM contents. Thus, all devices in the same local domain can share the DRM content that corresponds to the issued DRO.

- **Transferring an RO:** If a device D1 wants to transfer a DRO to anther device D2 in the same local domain, D1 can achieve this by requesting the transference to the LDM and erasing both the content and corresponding DRO. After the request, the LDM issues a new DRO and sends both the DRO and its corresponding content to D2.

- **Restricted Sharing:** As the LDM can issue a DRO to each device respectively, only the intended devices to play the DRM contents can obtain the access rights on the contents. Moreover, as the LDM can assign various access conditions to each DRO, each device has only the access right that is specified in the DRO issued to it. Thus, this requirement is preserved.

– **Temporal Sharing:** As the LDM can limit the validity period of a DRO and it also can distinguish the member devices in the local domain from the others, the proposed DRM system provides this requirement.

– **Revocation:** As mentioned in Section 5, this functionality can be provided in the proposed system.

Table 1. Functionality comparison results between the proposed system and the previous systems

Functionality	Popsescu [11]	OMA DRM [10]	Proposed system
Local domain licensing	×	×	◯
Contents local sharing	◯	◯	◯
Transferring an RO	×	◯	◯
Restricted Sharing	×	×	◯
Temporal Sharing	×	◯	◯
Revocation	◯	N/A†	◯

†As RI manages all local domains in the OMA DRM, the LDM does not exist.

7 Conclusion

In this paper, a new DRM system for local domain management has been presented in which access control on the contents is enabled by the right delegation strategy with proxy certificates. Moreover, the system also provides additional functionalities which are necessary for ordinary scenarios in home networks. Although we applied the delegation strategy to OMA DRM, this strategy can be applied to any other DRM systems easily.

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] Aegis DRM. http://www.aegisdrm.com/.
[2] J. Bloom, I. Cox, T. Kalker, J. Linnartz, M. Miller, and C. Traw, "Copy protection for DVD video," In Proc. IEEE, vol. 87, no. 7, pp. 1267–1276, July. 1999.
[3] Content scrambling system. http://www.dvdcca.org/css/.
[4] IBM. IBM Electronic Media Management System (EMMS). http://www-306.ibm.com/software/data/emms.

[5] F.A. Stevenson, "Cryptanalysis of Contents Scrambling System," DVD-Copy.com, Nov. 1999.
[6] Call for proposals for content protection & copy management technologies, July 2001.
[7] Real Networks. Helin DRM.
[8] S.A.F.A. van den Heuval, W. Jonker, F.L.A.J. Kamperman, and P.J. Lenoir, "Secure Content Management in Authorized Domains," In Proc. IBC 2002, pp. 467–474, Sept. 2002.
[9] Fraunhofer Institute. Light Weight DRM (LWDRM). http://www.lwdrm.com/.
[10] OMA. Open mobile alliance. http://www.openmobilealliance.org.
[11] B.C. Popescu, B. Crispo, F.L.A.J. Kamperman, A.S. Tanenbaum, "A DRM Security Architecture for Home Networks," In Proc. of 4th ACM Workshop on Digital Rights Management, Oct. 2004.
[12] S. Sovio, N. Asokan, and K. Nyberg, "Defining Authorization Domains Using Virtual Devices," In SAINT Workshops 2003, pp. 331–336, 2003.
[13] S. Michiels, K. Verslype, W. Joosen, and B. D. Decker, "Towards a Software Architecture for DRM", In ACM *DRM'05*, pp. 65–74, 2005.
[14] Smartright techinical white paper, "http://www.smartright.org/images/SMR/content/SmartRight_tech_whitepaper_jan28.pdf," Jan. 2003.
[15] B. Neuman. "Proxy-based authorization and accounting for distributed systems," Proc. 13th International Conference of Distributed Computing Systems, pp. 283–291, 1993.
[16] xCP Cluster Protocol. http://www.almaden.ibm.com/software/ds/ContentAssurance/papers/xCP_DVB.pdf.
[17] R. Holtz et al., Guide to Home Networks. http://www.ce.org/networkguide/default.asp
[18] L. Perlman, V. Welch, I. Foster, C. Kesselman and S. Tuecke. "Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile," RFC 3820, 2004.
[19] Housley et. al., "Internet X.509 Public Key Infrastructure," RFC 2459, The Internet Engineering Task Force (IETF), 1999, avaiable at http://www.ietf.org/rfc/rfc2459.txt
[20] The Globus Alliance, Globus Project, http://www.globus.org
[21] Microsoft Corporation. Windows Media DRM (WMDRM), 2005.

Fake Iris Detection Based on Multiple Wavelet Filters and Hierarchical SVM

Kang Ryoung Park[1,2], Min Cheol Whang[1], Joa Sang Lim[1], and Yongjoo Cho[1]

[1] Division of Media Technology, Sangmyung University,
7 Hongji-Dong, Jongro-Gu, Seoul, Korea
[2] Biometrics Engineering Research Center,
134 Sinchon-Dong, Seodaemun-Gu, Seoul, Korea

Abstract. With the increasing needs for higher security level, biometric systems have been widely used for many applications. Among biometrics, iris recognition system has been in the limelight for high security applications. Until now, most researches have been focused on iris identification algorithm and iris camera system. However, after the recent report of attacking iris recognition system by fake iris such as printed, photography and contact lens iris has been disclosed, the importance of fake iris detection is much increased.

So, we propose the new method of detecting fake iris. This research has following three advances compared to previous works. First, to detect fake iris, we check both the size change of pupil and the change of iris features in local iris area (near pupil boundary) by visible light. Second, to detect the change of local iris features, we used multiple wavelet filters having Gabor and Daubechies bases. Third, to enhance the detecting accuracy of fake iris, we used a hierarchical SVM (Support Vector Machine) based on extracted wavelet features.

Keywords: Iris Recognition, Fake Iris Detection, multiple wavelet filters, hierarchical SVM.

1 Introduction

Iris recognition system has been in the limelight for high security biometric applications [1][2][4][7]. Iris is the region which exists between sclera and pupil [1]. Its main function is to contract or dilate the pupil in order to adjust the penetrated light amount into the retina. Iris patterns are highly detailed and unique textures that almost remain unchanged from 6 month of age to death. After the recent report of attacking iris recognition system by fake iris such as printed, photography and contact lens iris has been disclosed, the importance of fake iris detection is much increased [15].

Fake iris detection is to detect and defeat a fake (forgery) iris image. In previous research, Daugman proposes the method of using FFT (Fast Fourier Transform) in order to check the high frequency spectral magnitude in the frequency domain, which can be observed distinctly and periodically from the print iris pattern because of the characteristics of the periodic dot printing [1][2][16]. However, the high frequency component cannot be detected in case that input printed iris image is blurred

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 246 – 256, 2006.
© Springer-Verlag Berlin Heidelberg 2006

purposely (by clever attacker) and the fake iris may be accepted as live one, consequently. Another method of fake iris detection was introduced by iris camera manufacturer. They use the method of turning on & off illuminator and checking specular reflection on a cornea. However, such a method can be easily deceived by using the printed iris image with cutting off printed pupil region and seeing through by attacker's eye, which can make corneal specular reflection [15]. Another research [27] proposed the method of using the dilation and contraction of pupil according to environment light stimulus. However, such a method cannot detect the fake iris made by (semi-transparent) patterned contact lens. That is because the iris region of the contact lens is semi-transparent and dilation & contraction is also visible in such case in spited of fake iris.

Another approach using Purkinje image was shown [28], but it cannot detect fake iris such as patterned contact lens. Advanced methods using visible lights were introduced [29][30]. However, they used single wavelet filter and local change of iris features could not be accurately detected especially in case of fake iris such as patterned contact lens. In addition, they used one level SVM for classification of fake and live iris image and the classification complexity was too great for SVM to correctly classify the samples, consequently.

As another countermeasure, we can consider multimodal biometric system. Multimodality means combining several biometric traits such as face and iris recognitions [2]. This concept is reported to increase the accuracy of the system in terms of EER as well as the resistance to counterfeiting attempts, simply because all traits should be spoofed simultaneously. However, total cost and system complexity are inevitably much increased due to the combination of more than two biometric systems.

To overcome such problems, we propose the new method of detecting fake iris based on multiple wavelet filters and a hierarchical SVM.

2 The Proposed Fake Iris Detection Method

2.1 Proposed Iris Camera and Controlling Illuminator

In case of the user with glasses, single IR-LED (InfraRed Light Emitting Diode) or visible illuminator can make large specular reflection (on glasses surface) which hides the whole iris region. In such cases, our system cannot recognize user and detect fake iris. So, we use dual illuminators. The IR pass filter is attached in front of iris camera lens in order to exclude the external visible light. The dual visible light illuminators are only used for making pupil's size change and in such a case, the IR-LED illuminator of the same side is turned on also, because the iris image only by visible light cannot be seen due to the IR pass filter.

So, our system controls the illuminators synchronized with CCD output signal [12]. When a user approaches in the operating range of the iris camera, our iris system perceives it. Then, our system controls (On/Off control) the IR-LED and visible light illuminator selectively. In our system, the IR-LED illuminator is composed of two wavelength of 760 and 880 nm. Each wavelength illuminator (760 or 880 nm) can be turned on selectively. After the iris recognition system is started, our system turns on

the left illuminator (760 + 880 nm) and performs the operation of capturing focused iris image. From that, focused and clear iris image can be captured and iris identification is performed. However, in case of users with glasses, the large specular reflection can happen on the glasses surface and in this case, the identification may be failed. Then, our system turns off the left IR-LED illuminator and turns on the right one and the same procedure is iterated. Then, the specular reflection does not happen in iris region and iris identification is successful. After that, our system turns on the right visible light for about 1 sec and checks the change of pupil's size for detecting fake iris. Detain accounts are shown in following section.

2.2 Checking the Change of Pupil's Size by Visible Light

By checking the change of pupil's size, we can detect the fake iris such as the 2D/3D printed/photograph iris, artificial eye and opaque contact lens. That is because such fake iris images do not show the change of pupil's size by the visible light. However, a live iris shows the distinctive change of pupil's size by visible light as shown in Fig. 1 [29]. Detail experimental results are shown in section 3.

To check the change of pupil size, we firstly detect the inner & outer boundary of iris by circular edge detection [1]. Then, we calculate the ratio of pupil radius to iris radius from iris images captured in case that visible light is off and on (Fig. 1 (a), (b)) respectively. If the variation of ratio does not exceed in the predetermined threshold, we regard the input iris image as fake one and vice versa. One thing to be considered is that the visible light can make dazzling to user's eye and it becomes severe in case that much visible light passes through pupil and approaches the retina. So, we reduce the dazzling by making the angle between the line of sight and the illuminating line bigger (more than 5 degree), which can lessen the volume of visible light passed through the pupil. Also, experimental results showed that with the blue light, the change of pupil's size can be seen most actively and we used the blue light illuminator [30].

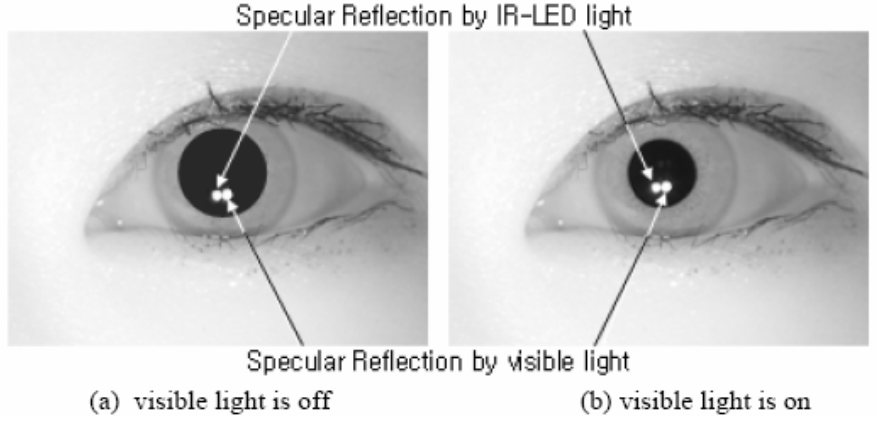


Fig. 1. Example of the change of pupil's size in case of live iris

However, because semi-transparent lens has the complex structure of transparent iris area and semi-transparent iris patterns on it, the change of pupil's size can be observed through transparent iris area. So, the fake detection method by checking the change of pupil's size cannot detect such a fake iris [29].

So, to overcome such problems, we propose the enhanced method of checking the change of iris features in the local iris area (adjacent of pupil boundary as shown in Fig. 3 and 4). As shown in Fig. 3, the iris pattern of live iris is dilated and contracted in case of the change of pupil's size and it is like rubber band model [1]. So, the iris pattern is not disappeared or appeared. However, the iris pattern of fake iris is not dilated and contracted like that of live iris. That is, the some iris pattern is hidden by the dilated pupil boundary as shown in Fig. 4 (b) [29][30]. So, we propose the enhanced method of checking the iris feature changing in the local iris area (adjacent of pupil boundary) and detecting the pattern contact lens.

In details, iris and pupil boundary was detected by circular edge detection [1]. In general, eyelash and eyelid areas are occluded in iris region and they should be removed for detecting the local change of iris features. For upper and lower eyelids are also located by eyelid detection mask and parabolic eyelid detection method [1][31]. Then, we determine the eyelash candidate region based on detected iris & pupil area and detect the eyelash region [32] as shown in Fig. 2(b). After that, the detected circular iris region is normalized as rectangular shape as shown in Fig. 2(c) [13]. In general, each iris image has variations about the length of its iris outer boundary. That is because there exists the size variation of iris per user (it is reported that the diameter of iris is about 10.7 ~ 13mm). Another is because the captured image size of iris may be changed according to Z distance between camera and eye. So, we adjust the length of iris outer boundary into 256 pixels by stretching and interpolation as shown in Fig. 2(c) and it is defined as 256 sectors. Then, the normalized iris image is divided as 8 tracks as shown in Fig. 2(c) and in each track, the weighted mean of gray level based on 1-D Gaussian kernel is calculated per each sector. By using the weighted mean of gray level, we can reduce the effect caused by iris segmentation error. Consequently, we can obtain the normalized image of 256*8 pixels.

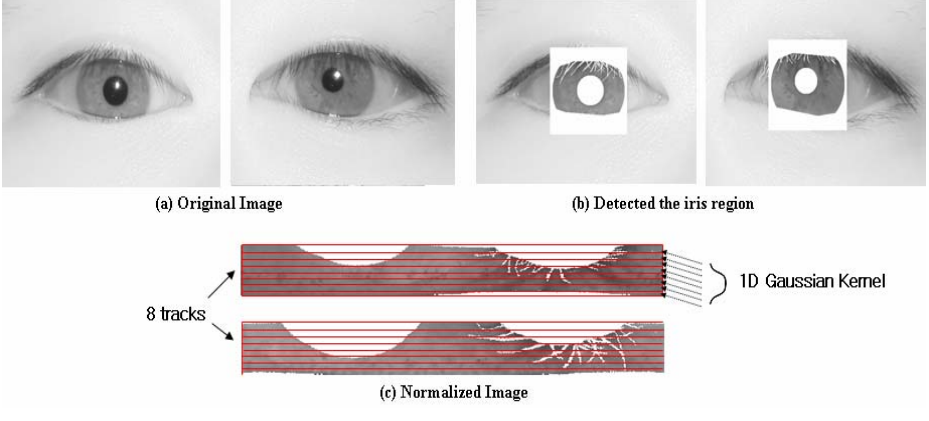


Fig. 2. Normalized Iris Image

Then, we apply Daubechies and Gabor wavelet filters in 4 tracks in the local iris area (adjacent to pupil boundary) in case that visible light is on. After that, the extracted iris feature values in 4 tracks are compared those extracted in case that visible light is off as shown in Fig. 3(b) and 4(b). The variations of the iris feature values in local 4 tracks are inputted to SVM classifier. From that, we determine the live and fake iris.

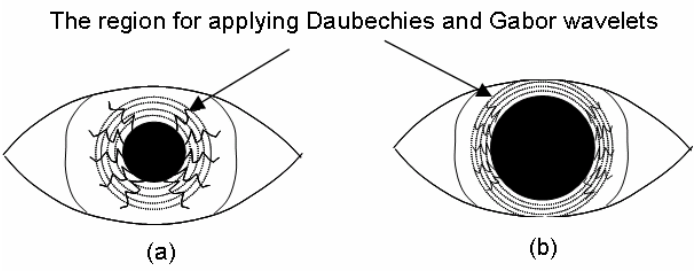


Fig. 3. The local iris region for applying Gabor and Daubechies wavelet (Live Iris)

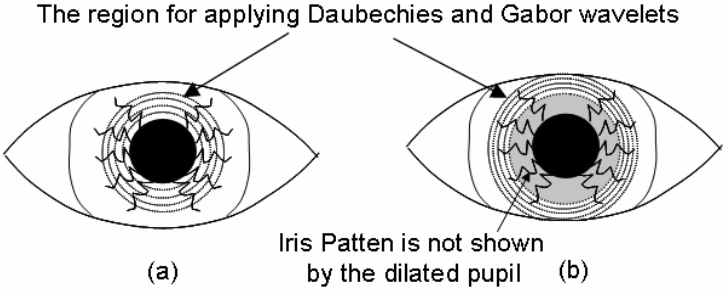


Fig. 4. The local iris region for applying Gabor and Daubechies wavelet (Fake Iris by Patterned Lens)

2.3 Iris Feature Extraction by Gabor and Daubechies Wavelet Filtering

As mentioned before, we use Gabor [30] and Daubechies wavelet filtering [29] in order to extract the iris feature information. Gabor wavelet was commonly used for iris recognition. The optimal kernel size and frequency of Gabor filter were determined to obtain minimum EER (Equal Error Rate) by testing with our live and fake iris DB. EER is the error rate in case that FAR (False Acceptance Rate) is same to that of FRR (False Rejection Rate). FAR is the error rate of accepting fake iris as live one. FRR is the error rate of rejecting live iris as fake one. After Gabor filtering, we can obtain amplitude and phase information. In conventional iris recognition, they used the phase information such as iris code bits. However, in our experiment, the EER in case of using cosine distance based on amplitude was smaller than that hamming distance based on phase.

To obtain more information in local iris area, we also used Daubechies wavelet filter [29]. Daubechies's wavelet is reported to have good localization trait and power of high texture classification [22]. In addition, Daubechies's wavelet has the characteristics of the orthogonality and factorization and provides compact support, but is not symmetric. The iris region in 4 tracks as shown in Fig. 2 is passed through low-pass and high-pass filters to generate the low-low, low-high, high-low and high-high subbands, The decomposition process is recursively applied on the low frequency channel to obtain the lower resolution subbands. For iris features, we use two features i.e. standard deviation and energy from the grey-level histogram of the subbands [19]. In this paper, we decompose each iris region into three level Daubechies's tap-4 filter which resulted in 12 subbands from which to extract iris features. In addition, we divide the subband images into local windows in order to get robust feature sets against shift, translation and noisy environment. Instead of traditional pyramid-structured wavelet transform, wavelet packet is used as a feature extraction method in this paper. Wavelet packet is very effective for texture classification [21] because their finite duration provides frequency and spatial locality.

After extracting the mean and standard deviation by Daubechies wavelet packet, we performed the feature normalization, because features with greater values have stronger influence in the classification process than that with small values. If we have N available data, we denote the i^{th} feature from the j^{th} pattern by x^{j}_{i}. For the normalization, we use the following equation [22].

$$\widehat{x}_i^j = \frac{x_i^j - \mu}{\sigma_i} \tag{1}$$

The mean $\widehat{x}_i$ and variance σ_i^2 of i^{th} feature are given by

$$\begin{aligned} \widehat{x}_i &= \frac{1}{N}\sum_{j=1}^{N} x_i \\ \sigma_i^2 &= \frac{1}{N}\sum_{j=1}^{N}(x_i^j - \widehat{x}_i)^2 \end{aligned} \tag{2}$$

After normalization, all features have zero mean and unit variance. For the second feature, we calculate the energy (the energy from the grey-level histogram) of each subband images.

2.4 Pattern Matching by Hierarchical SVM

With the transformed iris region in 4 tracks as shown in Fig. 2 and detected features (the change of amplitudes) by the Gabor wavelet and (the change of standard deviation and the energy of each subband) by the Daubechies wavelet, we use SVM (Support Vector Machine) to determine live or fake iris. SVMs have been recently proposed as a new technique for solving pattern recognition problems [23][24]. SVMs perform pattern recognition between two point classes by finding a decision surface determined by certain points of the training set, termed as Support Vectors (SV) and SVs are regarded as data which are difficult to be classified among training. At the same time, the decision surface found tends to have the maximum distance between

two classes. In general, it is reported that its classification performance is superior to that of MLP (Multi-Layered Perceptron). Especially, when plenty of positive and negative data are not obtained and input data is much noisy, the MLP cannot show the reliable classification results. In addition, MLP requires many initial parameter settings and it is usually performed by user's heuristic experience.

In this paper, we use a polynomial kernel of degree 5 for SVM in order to solve non-linearly separable problem. That is why the dimension of input data is big, so we use the polynomials of high degree. In this case, the problem is defined as 2 class problem. The first class shows the live iris and the second one does the fake iris. It is reported that the other inner products such as RBF, MLP, Splines and B-Splines do not affect the generation of support vector [25][29].

To reduce the complexity of SVM classification, we used following hierarchical method. At first, two SVMs are used for the detected features by Gabor and Daubechies wavelet, respectively. In general, the output values of SVM are represented as continuous one. In our case, the output values close to 0 represent live iris and those close to 1 do fake iris. Then, each output value of SVM was inputted to next SVM for final classification of live and fake iris. This method using hierarchical SVM can make classification problem be simple and consequently it can reduce the classification complexity compared to one SVM as shown in Fig.5.

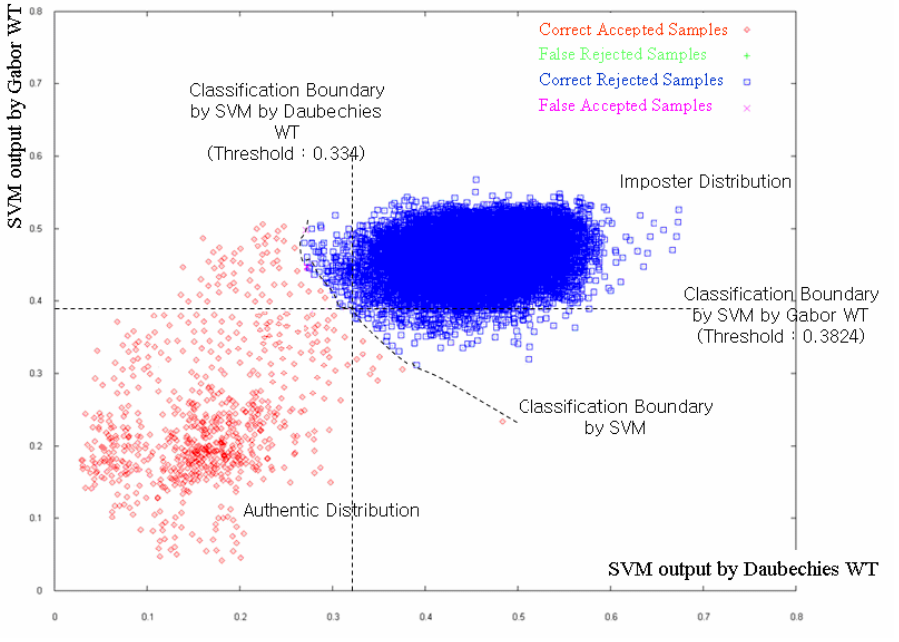


Fig. 5. Hierarchical SVM classification

Our experimental results comparing the polynomial kernel to MLP for SVM kernel show the same results. The C factor affects the generalization of SVM and we use 10,000 as C factor, which was selected by experimental results. We got 300 live iris

image frames (10 frames * 30 persons) and 200 fake iris images (20 frames * 10 fake iris) for SVM training and testing.

3 Experimental Results

For experiments, live irises were acquired from 30 persons (15 persons without glasses and 15 persons with glasses). Each person tried to recognize 10 times and total 300 iris images were acquired to test our algorithm. The color of pupil for all test data is black. According to field test, we could know the normal approaching speed of general user to iris camera is about 10 cm/sec (We measured the speed by Polhemus sensor [33]). In addition, we acquired total 10 fake iris samples for testing. They were composed of 3 samples for 2D printed/photographed iris image on planar or on/with convex surface. Also, 3 samples were acquired for 3D artificial eye. And 4 samples were for 3D patterned contact lens. With each sample, we tried to 20 times to spoof our counterfeit iris detection algorithm.

Experimental results showed the FAR was 0% (0/200) and the FRR was 0.33 % (1/300), but the FRR became 0 % allowing for the second trial. Here, the FAR means the error rate of accepting the fake iris as the live one. And the FRR means the error rate of rejecting the live iris as the fake one.

In case of only using Gabor wavelet, FAR was 1% (2/200) and FRR was 0.67% (2/300). When using only Daubechies wavelet, FAR was 0.5% (1/200) and FRR was 0.67% (2/300). According to results, we can know that Daubechies wavelet can show the better performance of extracting local iris features than Gabor filtering in case of using small region. Especially, we only extract the iris features in 4 tracks(as shown in Fig. 2) and the localization accuracy of iris / pupil boundary can affect the performance much more than using whole 8 tracks. Considering such condition, Daubechies wavelet shows the better performance than that by Gabor wavelet, because the mean value (Eq.(2)) extracted by Daubechies wavelet can reduce the effect by inaccurate localization of iris / pupil boundary.

In addition, in case of using one SVM for classification instead of hierarchical SVMs, the FAR was 0.5% (1/200) and FRR was 0.33% (1/300). From that, we can know the proposed method of using multiple wavelet filters and hierarchical SVMs show better performance compared to that of using single wavelet or one SVM classifier.

In case of using MLP for pattern matching instead of SVM, the error rate was increased. The MLP showed the FAR of 1.5 % (3/200) and the FRR of 1 % (3/300). In addition, the classification time using SVM was so small as 12 ms in Pentium-III 866Mhz.

Comparing to the fake iris detection method by Daugman [1][2][16], Daugman's method showed the FRR was 1 % (=3/300), but the FAR was over 51 % (102/200).

Though we tested 300 and 200 data for live and fake iris respectively, it is difficult to assert that the data set can represent the general characteristics of whole live and fake iris. Also, the error may be increased in case of using more data set. So, it is required to evaluate the performance by theoretical and we distribute the extracted feature value of live and fake iris into feature space. Then, we take mapping the feature value distributions of live and fake iris into two 2D Gaussian functions. With the generated functions and the experimental decision surface by SVM, the FAR/FRR

can be calculated by theoretical. Theoretical evaluation showed that the FAR and the FRR were 0.001 % and 0.28 %, respectively.

The total time for fake iris detection is taken 1,633 ms (on average), which includes 1,051 ms for turning on visible light and 582 ms for the processing.

In the next experiment, we measure the FAR (the error rate of accepting the fake iris as the live one) and FRR (the error rate of rejecting the live iris as the fake one) with distance between the input iris and camera. FAR(%) was 0 in all distances. FRR(%) was 0.21(at 20mm), 0.2(at 25mm), 0.21(at 30mm), 0.22(at 35mm), 0.2(at 40mm), 0.19(at 45mm), 0.19(at 50mm), 0.2(at 55mm), 0.19(at 60mm) and 0.2(at 65mm), respectively

From that, the FAR and FRR are almost same according to the distance between input iris and the iris camera. In the next experiment, we measure the FAR and FRR according to the change of environmental lighting condition with fluorescent lamp.

FAR(%) was 0 in all lighting conditions. FRR(%) was 0.21(at 250 Lux.), 0.19(at 500 Lux.), 0.2(at 750 Lux.), 0.2(at 1000 Lux.), 0.19(at 1250 Lux.), respectively. From that, the FAR and FRR are almost same according to the change of environmental lighting. That is because our iris camera has the IR pass filter and the functionality of AE (Auto Exposure).

4 Conclusions

For higher security level of iris recognition, the importance for detecting fake iris is much highlighted recently. In this paper, we propose the new method of checking the dilation and contraction of pupil size and the change of iris features by visible light.

To enhance the performance of our algorithm, we should have more field tests and consider more countermeasures against various situations and counterfeit samples in future. Also, the method for reducing processing time should be researched for user's convenience. In addition, we plan to research about the changing profile of pupil size according to time for more robust fake detection.

Acknowledgements

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Biometrics Engineering Research Center (BERC) at Yonsei University.

References

[1] John G. Daugman, "High Confidence Visual Recognition of Personals by a Test of Statistical Independence," IEEE Trans. Pattern Anal. Machine Intell., vol.15, no.11, pp.1148-1160, 1993

[2] http://www.iris-recognition.org (accessed on 2006. 08. 24)

[3] Keith Jack, Video Demystified. Harris, 1996.

[4] Anil K. Jain, Biometrics: Personal Identification in Networked Society. kluwer academic publishers, 1998.

[5] Smart Cards and Biometrics in Privacy-Sensitve Secure Personal Identification Systems. A Smart Card Alliance White Paper, May 2002
[6] Ramesh Jain, Machine Vision, McGraw-Hill International Edition, 1995
[7] Tony Mansfield, etc, "Biometric Product Testing Final Report," Draft 0.6, National Physical Laboratory, March 2001
[8] Steven C. Chapra, Raymond P. Canale, Numerical Methods for Engineers, McGraw-Hill International Editions, 1989
[9] Rafael C. Gonzalez, etc, Digital Image Processing, Addison-Wesley, 1992
[10] D. Ioammou, W. Huda, A. F. Laine, "Circle Recognition through a 2D Hough transform and Radius Histogramming," Image and Vision Computing, vol.17, pp.15-26, 1999
[11] Kang Ryoung Park, "Facial and Eye Gaze detection," LNCS, vol.2525, pp.368-376, 2002
[12] Kang Ryoung Park, Jaihie Kim, "A Real-time Focusing Algorithm for Iris Recognition Camera," IEEE Transactions on System, Man and Cybernatics, Part C, vol. 35, no. 3, pp.441-444, August 2005
[13] Hyun-Ae Park, Kang Ryoung Park, "Iris Recognition Based on Score Level Fusion by Using SVM," Pattern Recognition Letters, submitted
[14] Steven C. Chapra et al., Numerical Methods for Engineers, McGraw-Hill, 1989
[15] http://www.heise.de/ct/english/02/11/114/ (accessed on 2006. 08. 24)
[16] Daugman J, "Demodulation by complex-valued wavelets for stochastic pattern recognition," International Journal of Wavelets, Multi-resolution and Information Processing, vol.1, no.1, pp.1-17, 2003
[17] Vogel, et al. "Optical Properties of Human Sclera and Their Consequences for Transscleral Laser Applications," Lasers in Surgery and Medicine, vol.11(4), pp.331-340, 1991
[18] J. Deng et al., "Region-based Template Deformation and Masking for Eye Feature Extraction and Description," Pattern Recognition, vol.30(3), pp.403-419, 1997
[19] G. Kee, Y. Byun, K. Lee and Y. Lee, "Improved Technique for an Iris Recognition System with High Performance," AI 2001: Advances in Artificial Intelligence, pp. 177-188, 2001
[20] S. G. Mallet., "A Theory for Multi-resolution Signal Decomposition: The Wavelet Representation," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.11(4), pp.674-693, 1989
[21] R. E. Learned, W. C. Karl and A. S. Willsky, "Wavelet Packet based on Transient Signal Classification," Proc. of IEEE Conference on Time Scale and Time Frequency Analysis, pp.109-112, 1992
[22] Jain Jang, Kang Ryoung Park, Jinho Son, Yillbyung Lee, "Multi-unit Iris Recognition by Image Checking Algorithm," Lecture Notes in Computer Science (ICBA 2004), July 2004
[23] Vapnik, The Nature of Statistical Learning Theory, New York: Springer Verlag, 1995
[24] Vapnik, Statistical Learning Theory, Wiley-Interscience publication, 1998
[25] Saunders, Support Vector Machine User Manual, RHUL, Technical Report, 1998
[26] J. Daugman, "Biometric Decision Landscape," Technical Report No. TR482, University of Cambridge Computer Laboratory, 2000
[27] Matsushita, "Iris Image Capturing Device and Iris Image Authentication Device," Japanese Patent (Issued Number : 2002-247529)
[28] Eui Chul Lee, Kang Ryoung Park, Jaihie Kim, "Fake Iris Detection By Using the Purkinje Image," Lecture Notes in Computer Science (ICB'06), vol.3832, pp.397-403, January 5-7, 2006
[29] Kang Ryoung Park, "Robust Fake Iris Detection," Lecture Notes in Computer Science (AMDO), vol.4069, pp.10-18, July, 2006

[30] Kang Ryoung Park, "New Automated Iris Image Acquisition Method," Applied Optics, vol. 44, no. 5, pp.713~734, Feb. 2005
[31] Y.K Jang et al., “Robust Eyelid Detection for Iris Recognition,” Journal of Institute of Electronics Engineers of Korea, submitted
[32] Byung Joon Kang, Kang Ryoung Park, “A Study on Fast Iris Restoration Based on Focus Checking,” Lecture Notes in Computer Science (AMDO), vol.4069, pp.19~28, July, 2006
[33] http://www.polhemus.com (accessed on 2006. 08 24)

Multi-block Collisions in Hash Functions Based on 3C and 3C+ Enhancements of the Merkle-Damgård Construction*

Daniel Joščák and Jiří Tůma

Charles University in Prague,
Faculty of Mathematics and Physics, Department of Algebra,
Sokolovská 83, 186 75 Prague 8, Czech Republic
joscd1am@karlin.mff.cuni.cz, tuma@karlin.mff.cuni.cz

Abstract. At the ACISP 2006 conference Praveen Gauravaram et al [2] proposed 3C and 3C+ constructions as enhancements of the Merkle-Damgård construction of cryptographic hash functions. They conjectured these constructions improved multi-block collision resistance of the hash functions. In this paper we show that the recently found collision attack on MD5 can be easily extended to the 3C and 3C+ constructions based on the MD5 compression function. In fact we show that if an algorithm satisfying some mild assumptions can find multi-block collisions for the Merkle-Damgård construction then it can be easily modified to find multi-block collisions for the 3C and 3C+ constructions based on the same compression function.

Keywords: hash functions, multi-block collision attack, 3C and 3C+ constructions.

1 Introduction

Research in the design and analysis of cryptographic hash functions has been very active since Wang at al [7] published their first collision search algorithm for the MD5 hash function. Collision search algorithms for other hash functions have been discovered, in particular for SHA-0, see [1], [8]. An algorithm for finding collisions in SHA-1 that is significantly more efficient than the generic birthday attack was announced in [9].

In the light of these attacks Gauravaram et al [2] have proposed a slight modification to the Merkle-Damgård construction for an improved protection against many known attacks on MD based hash functions. Their idea is to add additional registers that would collect xors of all chaining variables. After the message is processed the content of additional registers is padded to provide one more message block and the extra block is used as an input for the last calculation of the compression function. Thus the original MD construction remains and the extra security is supposed to be provided by the additional registers, see Figure 1.

* Research was supported by the Institutional grant MSM0021620839.

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 257–266, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Since the 3C construction contains the original MD contruction, any n-block internal collision for the 3C construction must be in fact an n-block collision of the MD construction based on the same compression function. However, because of the extra use of the compression function at the end of the 3C construction one cannot claim that an n-block collision for the 3C construction must be also an n-block collision of the MD construction. To find an n-block collision (where $n \geq 2$) for the 3C construction that is not an n-block collision for the MD construction based on the same compression function would require to find a collision in the compression function with different IV's and possibly different input blocks.

In this paper we show that the 3C construction does not increase significantly resistance against multi-block collisions. In fact, under very mild assumptions we prove that if there is an algorithm that finds n-block collisions for the MD construction based on a compression function, then one can easily find $(2n)$-block collisions for the 3C construction based on the same compression function and $(2n + 1)$-block collisions for its modification called 3C+. Our theorem can be applied in particular to the MD5 compression function.

We also observe that the 2-block collisions for the SHA-0 hash function published in [8] are in fact also 2-block collisions for the 3C construction based on the same collision function.

The paper is organized as follows. In section 2 we discuss the 3C and 3C+ design principles, in section 3 we point out a few important properties of the recent 2-block collision attacks on MD5. In Section 4 we prove two simple general theorems how multi-collision attacks on the MD construction can be extended to multi-collision attacks on the 3C and 3C+ constructions. We conclude the paper in section 5. In the appendix we present concrete examples of colliding messages for the 3C and 3C+ constructions based on the compression function of MD5.

2 Description of 3C and 3C+

The 3C construction is a generic construction designed as an enhancement to the Merkle-Damgård construction with the idea to increase its resistance against multi-block collision attacks. One of the main properties of the 3C construction is that it is as efficient as the standard hash functions when it is instantiated with the compression functions of any of these hash functions.

The 3C construction accumulates every chaining state of the MD construction by xoring it to the register already containing xor of all previous chaining states.

Thus if IV_i is the chaining variable obtained as the result of i-th iteration of the compression function (IV_0 is the initialization vector), then the value of the additional accumulation registers (denoted by C_i) after the i-th iteration of the compression function is $C_1 = \mathrm{IV}_1$ and

$$C_i = C_{i-1} \oplus \mathrm{IV}_i = \mathrm{IV}_1 \oplus \mathrm{IV}_2 \oplus \cdots \oplus \mathrm{IV}_i$$

for $i = 2, \ldots, L$, where L is the number of blocks of the message. The authors also suggest in their paper [2] that different variants can be obtained for 3C

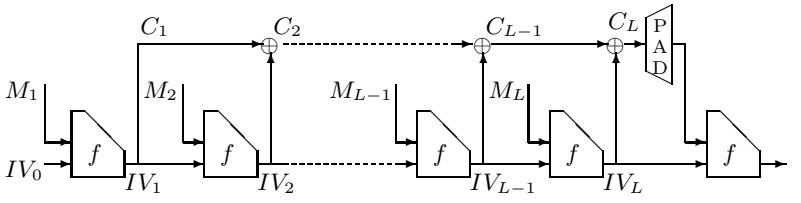


Fig. 1. 3C construction of hash function

by replacing the xor function in the accumulation chain by other non-linear functions.

The 3C+ construction is a different modification of the 3C construction in which yet another chain D_i, $i = 1, \ldots, L$ of additional registers accumulating the values of chaining variables is added. This time $D_1 = \mathrm{IV}_0$ and

$$D_i = D_{i-1} \oplus \mathrm{IV}_i = \mathrm{IV}_0 \oplus \mathrm{IV}_2 \oplus \cdots \oplus \mathrm{IV}_i$$

for $i = 2, \ldots, L$. Thus

$$D_i = C_i \oplus \mathrm{IV}_1 \oplus \mathrm{IV}_0$$

for every $i = 2, \ldots, L$.

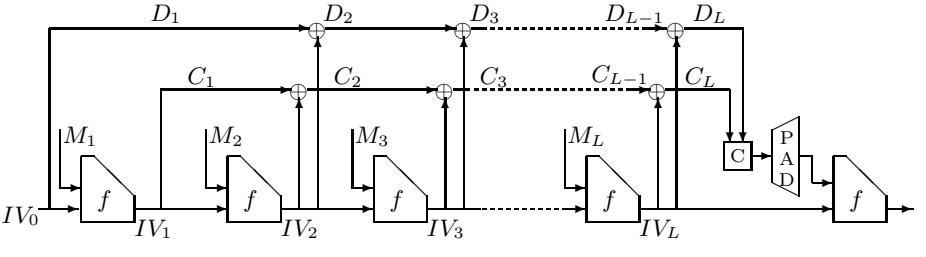


Fig. 2. 3C+ construction of hash function

3 Recent Multi-block Collision Attacks

The hash function MD5 uses four 32-bit registers to keep the value of each chaining variable IV_i. We denote them by $\mathrm{IV}_{i,0}, \mathrm{IV}_{i,1}, \mathrm{IV}_{i,2}, \mathrm{IV}_{i,3}$. Thus

$$\mathrm{IV}_i = (\mathrm{IV}_{i,0} || \mathrm{IV}_{i,1} || \mathrm{IV}_{i,2} || \mathrm{IV}_{i,3}).$$

Wang et al presented in [7] an algorithm for finding 2-block collisions in the hash function MD5. Their algorithm works for an arbitrary initialization vector IV_0. If $(M_1||M_2)$ and $(M_1'||M_2')$ are two colliding messages found by their algorithm then the modular differences of the chaining variables after processing the first blocks M_1 and M_1' are

$$\begin{aligned}
\Delta_{i,0} &= \mathrm{IV}'_{1,0} - \mathrm{IV}_{1,0} = 2^{31} \\
\Delta_{i,1} &= \mathrm{IV}'_{1,1} - \mathrm{IV}_{1,1} = 2^{31} + 2^{25} \\
\Delta_{i,2} &= \mathrm{IV}'_{1,2} - \mathrm{IV}_{1,2} = 2^{31} + 2^{25} \\
\Delta_{i,3} &= \mathrm{IV}'_{1,3} - \mathrm{IV}_{1,3} = 2^{31} + 2^{25}, \qquad (1)
\end{aligned}$$

where

$$\begin{aligned}
\mathrm{IV}_1 &= f(\mathrm{IV}_0, M_1) \\
\mathrm{IV}'_1 &= f(\mathrm{IV}_0, M'_1)
\end{aligned}$$

and f is the compression function used in MD5.

Wang et al in [7] also presented a set of so-called sufficient conditions for registers in computation of $f(\mathrm{IV}, M_1)$ to produce the first blocks of a pair of colliding messages. These conditions in fact were not sufficient and various authors, e.g. [5] [6] offered their sets of sufficient conditions. For our purposes only the conditions for IV_1 are important and these conditions were the same for all authors. In fact, we need only four of the sufficient conditions for IV_1 and these four conditions are described in the Table 1.

Table 1. Prescribed conditions for IV_1

$IV_{1,0}$	....
$IV_{1,1}$	......0. ....
$IV_{1,2}$	.....01. ....
$IV_{1,3}$	......0. ....

The exact value of $\mathrm{IV}_1 \oplus \mathrm{IV}'_1$ then follows from given modular differences (1) and prescribed conditions for IV_1 in the Table 1. Thus $\mathrm{IV}_1 \oplus \mathrm{IV}'_1$ is a constant independent of the initialization vector IV_0 and the first blocks M_1 and M'_1 of the colliding messages $(M_1||M_2)$ and $(M'_1||M'_2)$.

Table 2. Prescribed δ for IV_1

$\delta_{1,0} = \mathrm{IV}_{1,0} \oplus \mathrm{IV}'_{1,0}$	10000000 00000000 00000000 00000000
$\delta_{1,1} = \mathrm{IV}_{1,1} \oplus \mathrm{IV}'_{1,1}$	10000010 00000000 00000000 00000000
$\delta_{1,2} = \mathrm{IV}_{1,2} \oplus \mathrm{IV}'_{1,2}$	10000110 00000000 00000000 00000000
$\delta_{1,3} = \mathrm{IV}_{1,3} \oplus \mathrm{IV}'_{1,3}$	10000010 00000000 00000000 00000000

The collision finding algorithm for SHA-0 by Wang et al [8] also finds 2-block colliding messages but the structure of the messages in this attack is different than in the case of MD5. The first blocks of the colliding messages $(M_1||M_2)$ and $(M_1||M'_2)$ are the same and serve to obtain the chaining variable IV_1 satisfying the conditions sufficient for finding the second blocks M_2 and M'_2. The algorithm again works for an arbitrary IV_0.

In another paper [9] Wang et al propose an algorithm for finding 2-block collisions in SHA-1 that is faster than the generic birthday attack. Although no real collisions in SHA-1 have been found so far, the form of colliding messages of the proposed attack is in fact the same as in the case of MD5. It means that the algorithm should work for any IV_0 and $\mathrm{IV}_1 \oplus \mathrm{IV}'_1$ should be a constant independent of IV and M_1 and M'_1.

4 Multi-block Collision Attacks on 3C and 3C+

The idea of the attack on the 3C construction when the compression function is the same as in MD5 is very simple. First, we find 2-block colliding messages $(M_1||M_2)$ and $(M'_1||M'_2)$ in MD5 using the attack by Wang et al [7]. Then we take the chaining variable $\mathrm{IV}_2 = \mathrm{IV}'_2$ as the initialization vector for the second run of the Wang et al algorithm. In this way we obtain another pair of messages $(M_3||M_4)$ and $(M'_3||M'_4)$. The 4-block messages $(M_1||M_2||M_3||M_4)$ and $(M'_1||M'_2||M'_3||M'_4)$ then form a collision for the 3C construction based on the MD5 compression function. The scheme of the attack and the distribution of differences are shown on Figure 3.

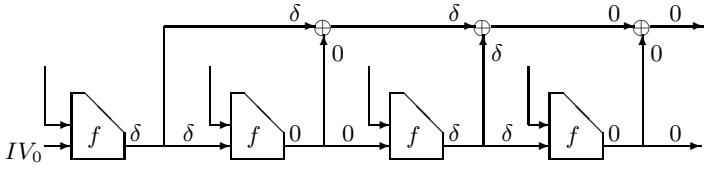


Fig. 3. 4-block internal collision attack on 3C without the final processing

A formal verification of the idea is contained in the following theorem.

Theorem 1. *Let H be an MD hash function based on a compression function f. Suppose that for some $n \geq 2$ there exists an algorithm finding n-block collisions for H that works for any initialization vector IV_0 and has the property that $\mathrm{IV}_i \oplus \mathrm{IV}'_i$ for $i = 1, \ldots, n$ is a constant independent of IV_0 and the actual colliding messages (but can be dependent on i). Then there exists an algorithm that finds $(2n)$-block collisions for the 3C construction based on the same compression function f.*

The running time of the algorithm for finding collisions in the 3C construction is twice the running time of the algorithm for finding collisions in the MD construction using the same compression function.

Proof. Let $(M_1||M_2||\cdots||M_n)$ and $(M'_1||M'_2||\cdots||M'_n)$ be two colliding messages obtained by the first run of the algorithm finding collisions in H. Thus $\mathrm{IV}_n = \mathrm{IV}'_n$. We use this value as the initialization vector for the second run of the collision search algorithm for H. We obtain another pair of colliding messages $(M_{n+1}||M_{n+2}||\cdots||M_{n+n})$ and $(M'_{n+1}||M'_{n+2}||\cdots||M'_{n+n})$. We denote the

chaining variables in the second run of the algorithm by IV_{n+i} and IV'_{n+i} for $i = 1, \ldots, n$.

By our assumption on the collision search algorithm for H we can write

$$\mathrm{IV}_i \oplus \mathrm{IV}'_i = \mathrm{IV}_{n+i} \oplus \mathrm{IV}'_{n+i}$$

for every $i = 1, \ldots, n$. Thus we obtain

$$C_{2n} = \bigoplus_{i=1}^{2n} \mathrm{IV}_i$$

and

$$C'_{2n} = \bigoplus_{i=1}^{2n} \mathrm{IV}'_i.$$

Hence

$$\begin{aligned} C_{2n} \oplus C'_{2n} &= \bigoplus_{i=1}^{2n} \mathrm{IV}_i \oplus \bigoplus_{i=1}^{2n} \mathrm{IV}'_i = \bigoplus_{i=1}^{2n} (\mathrm{IV}_i \oplus \mathrm{IV}'_i) \\ &= \bigoplus_{i=1}^{n} (\mathrm{IV}_i \oplus \mathrm{IV}'_i) \oplus (\mathrm{IV}_{n+i} \oplus \mathrm{IV}'_{n+i}) \\ &= 0. \end{aligned}$$

Since $\mathrm{IV}_{2n} = \mathrm{IV}'_{2n}$, the messages $(M_1||\cdots||M_n||M_{n+1}||\cdots||M_{2n})$ and $(M'_1||\cdots||M'_n||M'_{n+1}||\cdots||M'_{2n})$ form a collision for the 3C construction based on f. □

In Section 3 we explained that the Wang et al [7] collision search algorithm for MD5 satisfied the assumptions of Theorem 1 for $n = 2$. Thus there exists an algorithm finding 4-block collisions in the 3C construction based on the MD5 compression function. The fastest implementation of the Wang et al algorithm known in the moment of writing the paper is by Klima [4] and finds collisions in MD5 in about 30 seconds in average. Thus at this moment collisions in the 3C construction based on the MD5 compression function can be found within a minute.

As for the 3C construction based on the SHA-0 compression function there is no need for running the algorithm twice to obtain a collision. Since the collision search algorithm for SHA-0 finds colliding messages of the form $(M_1||M_2)$ and $(M_1||M'_2)$, we get $\mathrm{IV}_1 = \mathrm{IV}'_1$ and $\mathrm{IV}_2 = \mathrm{IV}'_2$, thus $C_2 = C'_2$. Hence the SHA-0 collisions found by the algorithm are simultaneously collisions for the 3C construction based on the SHA-0 compression function.

Since the theoretical algorithm for finding collisions in SHA-1 proposed by Wang et al in [9] also satisfies the assumption of Theorem 1 running the algorithm twice should again produce a 4-block collision in the 3C construction based on the SHA-1 compression function.

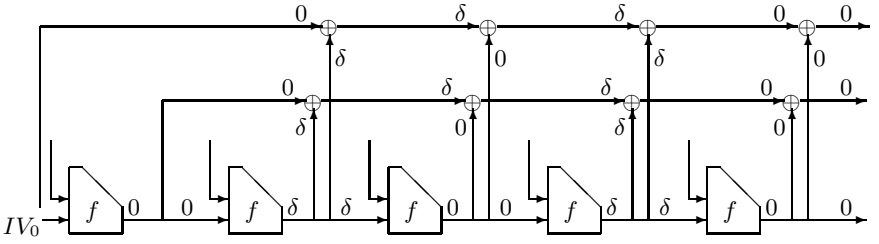


Fig. 4. 5-block collision attack on 3C+ without the final processing

The Figure 4 shows how a 5-block collision for the 3C+ construction based on the $MD5$ compression function can be found. The only difference with the collision search algorithm for the 3C construction is that we start with an arbitrary message block M_1, calculate the value of the compression function for the block with given IV_0 to obtain a new initialization vector IV_1 and then we run the collision search algorithm for the 3C construction with the initialization vector IV_1. We obtain messages $(M_1||M_2||M_3||M_4||M_5)$ and $(M_1||M_2'||M_3'||M_4'||M_5')$ such that $C_5 = C_5'$ and $\mathrm{IV}_5 = \mathrm{IV}_5'$. Since $D_5 = C_5 \oplus \mathrm{IV}_1 \oplus \mathrm{IV}_0$ and $D_5' = C_5' \oplus \mathrm{IV}_1 \oplus \mathrm{IV}_0$, we obtain also $D_5 = D_5'$.

From this observation one obtains the following theorem.

Theorem 2. *Suppose there exists an algorithm finding k-block collisions in the 3C construction based on a compression function f. Then there exists an algorithm for finding $(k+1)$-block collisions in the 3C+ construction based on the same compression function f.*

The running time of the algorithm for the 3C+ construction is equal the running time of the algorithm for the 3C+ construction plus the running time of the one calculation of the compression function.

5 Conclusion

In this paper, we have shown how to find collisions for 3C and 3C+ constructions based on a compression function f provided a collision search algorithm for the MD construction based on f is known. We also present concrete collisions for the 3C and 3C+ constructions based on the MD5 compression function.

Acknowledgment

The authors thank to Praveen Gauravaram for sending them the preliminary version of [2] and [3] which motivated this paper.

References

1. Eli Biham, Rafi Chen, Antonie Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. *Collisions of SHA-0 and reduced SHA-1.* In Ronald Cramer, editor, Advances in Cryptology – EUROCRYPT 2005, volume 3494 of Lecture Notes in Computer Science, pages 36–57. Springer, 2005.

2. Praveen Gauravaram, William Millan, Ed Dawson, and Kapali Viswanathan. *Constructing Secure Hash Functions by Enhancing Merkle-Damgård Construction.* In Lynn Batten, Reihaneh Safavi-Naini, editors, Information Security and Privacy, volume 4058 of Lecture Notes in Computer Science, pages 407–420. Springer, 2006.
3. Praveen Gauravaram, William Millan, Ed Dawson, and Kapali Viswanathan. *Constructing Secure Hash Functions by Enhancing Merkle-Damgård Construction (Extended Version).* Information Security Institute (ISI), Queensland University of Technology (QUT), number QUT-ISI-TR-2006-013, `http://www.isi.qut.edu.au/research/publications/technical/qut-isi-tr-2006-013.pdf`, July 2006.
4. Vlastimil Klima. *Tunnels in Hash Functions: MD5 Collisions Within a Minute*, Cryptology ePrint Archive: Report 105/2006, http://eprint.iacr.org/2006/105.
5. Jie Liang, Xuejia Lai *Improved collision attack on hash function MD5*, Cryptology ePrint Archive: Report 425/2005, http://eprint.iacr.org/2005/425.
6. Jun Yajima, Takeshi Shimoyama. *Wangs sufficient conditions of MD5 are not sufficient*, Cryptology ePrint Archive: Report 263/2005, http://eprint.iacr.org/2005/263.
7. Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. *Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD.* Cryptology ePrint Archive, Report 2004/199, 2004. http://eprint.iacr.org/2004/199.
8. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. *Effcient collision search attacks on SHA-0.* In Victor Shoup, editor, Advances in Cryptology - CRYPTO – 05, volume 3621 of Lecture Notes in Computer Science, pages 1–16. Springer, 2005, 14–18 August 2005.
9. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. *Finding collisions in the full SHA-1.* In Victor Shoup, editor, Advances in Cryptology - CRYPTO – 05, volume 3621 of Lecture Notes in Computer Science, pages 17–36. Springer, 2005, 14–18 August 2005.
10. Xiaoyun Wang and Hongbo Yu. *How to Break MD5 and Other Hash Functions.* In Ronald Cramer, editor, Advances in Cryptology – EUROCRYPT 2005, volume 3494 of Lecture Notes in Computer Science, pages 19–35. Springer, 2005.

A Examples of Collisions

Table 3. Collision in 3C invoked with MD5 compression function

IV	0x67452301	0x10325476	0x98badcfe	0xefcdab89
M_1	0x4e1a8245	0x5fe0e55d	0xfe3faa53	0x0d8546b3
	0x18ccad34	0xac0bae59	0xd59d3352	0x4805693e
	0x06342cd5	0x81b41206	0x83c2bea3	0x8fd22557
	0xc41a4cd6	0x9e9a4fe1	0x818ae34d	0x1a97e731
N_1	0x4e1a8245	0x5fe0e55d	0xfe3faa53	0x0d8546b3
	0x98ccad34	0xac0bae59	0xd59d3352	0x4805693e
	0x06342cd5	0x81b41206	0x83c2bea3	0x8fd2a557
	0xc41a4cd6	0x9e9a4fe1	0x18ae34d	0x1a97e731
IV_1	0xadebbbec	0xc85d058e	0xa2672e58	0xb91d144b
IV'_1	0x2debbbec	0x4a5d058e	0x24672e58	0x3b1d144b
$\mathrm{IV}_1 \oplus \mathrm{IV}'_1$	0x80000000	0x82000000	0x86000000	0x82000000
M_2	0x06faa233	0x1c84a4bf	0xf38ee5f1	0x08deb9af
	0x467ad36b	0x4c900712	0xd6a37d26	0x11f6de56
	0x8577e045	0x299991d5	0x5940588e	0x3fd25887
	0x301fc8fa	0x77dc0e81	0xe8c1a1a7	0x13d51d82
N_2	0x06faa233	0x1c84a4bf	0xf38ee5f1	0x08deb9af
	0xc67ad36b	0x4c900712	0xd6a37d26	0x11f6de56
	0x8577e045	0x299991d5	0x5940588e	0x3fd1d887
	0x301fc8fa	0x77dc0e81	0x68c1a1a7	0x13d51d82
$\mathrm{IV}_2 = \mathrm{IV}'_2$	0xa918ce8d	0xb7ea0df6	0x69bdb806	0x713af4de
M_3	0xcd71fe0c	0x58d0f463	0xa9399e1d	0x7db79e98
	0x3622a432	0x736cb277	0x011cb460	0x6a04e9b4
	0x06332d55	0x23f47e02	0x799ab597	0xd3ba5325
	0xb9e866e6	0xde6b9cd3	0xde6cebbb	0x0b4c3783
N_3	0xcd71fe0c	0x58d0f463	0xa9399e1d	0x7db79e98
	0xb622a432	0x736cb277	0x011cb460	0x6a04e9b4
	0x06332d55	0x23f47e02	0x799ab597	0xd3bad325
	0xb9e866e6	0xde6b9cd3	0x5e6cebbb	0x0b4c3783
IV_3	0x2b30549a	0x089c590a	0x52710661	0x6932f794
IV'_3	0xab30549a	0x8a9c590a	0xd4710661	0xeb32f794
$\mathrm{IV}_3 \oplus \mathrm{IV}'_3$	0x80000000	0x82000000	0x86000000	0x82000000
M_4	0x96ded638	0x4c1be33a	0xd46e6a5f	0xdbc8da73
	0x473af92b	0x4d0da98e	0x56dd6d3e	0xd19e7bd1
	0x53f857cd	0x4c25f191	0x918be4da	0xc09e206c
	0x320b28d4	0xcc6c0e7a	0x68515c76	0x57840834
N_4	0x96ded638	0x4c1be33a	0xd46e6a5f	0xdbc8da73
	0xc73af92b	0x4d0da98e	0x56dd6d3e	0xd19e7bd1
	0x53f857cd	0x4c25f191	0x918be4da	0xc09da06c
	0x320b28d4	0xcc6c0e7a	0xe8515c76	0x57840834
$\mathrm{IV}_4 = \mathrm{IV}'_4$	0x6a1a021a	0xc81fe980	0x88e1db5b	0x512e7c88

Table 4. Collision in 3C+ invoked with MD5 compression function

IV	0x67452301	0x10325476	0x98badcfe	0xefcdab89
M_1	0x0634add5	0x4074c002	0x7baaf717	0x0f522d75
	0xbf6ac0ec	0xa4885903	0x7349e78b	0x2aad1b45
	0x281dfb7e	0x173e6c0c	0xab79fc54	0x39453670
	0x44fb372b	0x4d5259c8	0xf7ad2d48	0xd1254b51
N_1	0x0634add5	0x4074c002	0x7baaf717	0x0f522d75
	0xbf6ac0ec	0xa4885903	0x7349e78b	0x2aad1b45
	0x281dfb7e	0x173e6c0c	0xab79fc54	0x39453670
	0x44fb372b	0x4d5259c8	0xf7ad2d48	0xd1254b51
$\mathrm{IV}_1 = \mathrm{IV}'_1$	0xd3f4b63c	0x595f4645	0xa890d3d0	0x9cc907db
M_2	0xa72fc176	0x64b7a050	0xe266ae7a	0x1b21009e
	0xfac1ee4c	0x9e588e8e	0x076d346d	0x805529b7
	0x0633ad55	0x02342602	0x83b4ba0b	0x56d1d924
	0x82d9651a	0xba9c8de6	0xebbbe37e	0xb78c63d5
N_2	0xa72fc176	0x64b7a050	0xe266ae7a	0x1b21009e
	0x7ac1ee4c	0x9e588e8e	0x076d346d	0x805529b7
	0x0633ad55	0x02342602	0x83b4ba0b	0x56d25924
	0x82d9651a	0xba9c8de6	0x6bbbe37e	0xb78c63d5
IV_2	0xead1c69e	0xd19e34c2	0xca2e528e	0xb1790589
IV'_2	0x6ad1c69e	0x539e34c2	0x4c2e528e	0x33790589
$\mathrm{IV}_2 \oplus \mathrm{IV}'_2$	0x80000000	0x82000000	0x86000000	0x82000000
M_3	0x6dbb34a0	0x9c1b815b	0x7ceb8ffd	0x1502296c
	0x467d585b	0x4d0d8038	0xc6db2d16	0x00d11ad5
	0xd2b2eeed	0x4a04145b	0x2f79d4aa	0x00be08a0
	0xf2e830f3	0x10bc0a85	0xe9019cb8	0x4fd512a2
N_3	0x6dbb34a0	0x9c1b815b	0x7ceb8ffd	0x1502296c
	0xc67d585b	0x4d0d8038	0xc6db2d16	0x00d11ad5
	0xd2b2eeed	0x4a04145b	0x2f79d4aa	0x00bd88a0
	0xf2e830f3	0x10bc0a85	0x69019cb8	0x4fd512a2
$\mathrm{IV}_3 = \mathrm{IV}'_3$	0x46321911	0x9d317bd2	0xfde6d50e	0xeb2170d8
M_4	0x122cdc12	0x5f60de22	0xedac78fd	0xf506f854
	0x2b85436b	0x3c980908	0xda4c144d	0x03344bbe
	0x0634ad55	0x0113f402	0x80aab777	0x13888f67
	0xadea26f7	0x623cc142	0x1192759e	0x0e74317c
N_4	0x122cdc12	0x5f60de22	0xedac78fd	0xf506f854
	0xab85436b	0x3c980908	0xda4c144d	0x03344bbe
	0x0634ad55	0x0113f402	0x80aab777	0x13890f67
	0xadea26f7	0x623cc142	0x9192759e	0x0e74317c
IV_4	0x754b85c2	0x45386ef2	0x3adad7b7	0x61523316
IV'_4	0xf54b85c2	0xc7386ef2	0xbcdad7b7	0xe3523316
$\mathrm{IV}_4 \oplus \mathrm{IV}'_4$	0x80000000	0x82000000	0x86000000	0x82000000
M_5	0x65171431	0x2615affc	0x2a2519e7	0xe2e99ce8
	0x44bcf42b	0x4c4def0e	0x47aadd22	0x127d7d56
	0x62bf776d	0x6cc9d58b	0x597058d6	0x602a5867
	0x3e2bc8ce	0xb3ec1267	0x68716155	0x17a50429
N_5	0x65171431	0x2615affc	0x2a2519e7	0xe2e99ce8
	0xc4bcf42b	0x4c4def0e	0x47aadd22	0x127d7d56
	0x62bf776d	0x6cc9d58b	0x597058d6	0x6029d867
	0x3e2bc8ce	0xb3ec1267	0xe8716155	0x17a50429
$\mathrm{IV}_5 = \mathrm{IV}'_5$	0x1453b7b0	0x803e8aee	0xfd85765e	0x176ca5d9

Cryptanalysis of T-Function-Based Hash Functions

Applications to MySQL Password Algorithms

Frédéric Muller[1] and Thomas Peyrin[2]

[1] HSBC, Paris, France
frederic.muller@hsbc.fr
[2] France Télécom R&D, Issy les Moulineaux, France
thomas.peyrin@orange-ft.com

Abstract. T-functions are a useful new tool to design symmetric-key algorithms, introduced by Klimov and Shamir in 2002. They have already been used to build stream ciphers and new applications for block ciphers and hash functions have been recently suggested.

In this paper, we analyze the security of several possible constructions of hash functions, based on T-functions. We show that most natural ideas are insecure. As an application, we describe a practical preimage attack against the dedicated hash function used in the MySQL password-based authentication mechanisms.

Keywords: T-functions, hash functions, MySQL.

1 Introduction

Following many recent cryptanalysis results against popular algorithms like MD5 [23] or SHA-1 [22], the design of secure hash functions is now in question. Some issues have been raised about the theoretical security of the Merkle-Damgård mode of operation [2,4,5] but the heart of the problem seems to be the design of "good" compression functions. The typical design method is to use a block cipher in the Davies-Meyer mode, however the large block length necessary to prevent birthday attacks often requires the use of *ad hoc* block ciphers, instead of relying on AES for instance.

An interesting new research direction is to find new paradigms to build compression functions. As an example, one can mention SMASH, a recent attempt by Knudsen [10] which was later broken by Pramstaller *et al* [20], or the Butterfly transforms [18]. Another interesting idea consists in **building a compression function which relies on another primitive than a block cipher**. Some hash functions already rely on number-theoretic assumptions, however it might also be interesting to find constructions that rely on simple and efficient symmetric-key primitives.

In this paper, we investigate the security of hash functions based on T-functions. Our analysis is motivated by the observation that many popular functions (like MD5 and SHA-1) are close to a T-function (only the circular rotations

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 267–285, 2006.
© Springer-Verlag Berlin Heidelberg 2006

break the triangular structure). Also, we observed that some *ad hoc* designs (like the MySQL dedicated hash function, for instance) were based on T-functions.

First, we investigate several "natural" constructions of T-function-based hash functions that a designer could consider, and we show why they are insecure. This analysis is actually quite simple, but we show that it has a devastating effect when taking the concrete example of the MySQL dedicated hash function, whose compression function is based on a T-function. We describe a practical preimage attack against this hash function, which allows to retrieve instantly the user's passwords.

2 Possible Use of T-Functions in Hash Functions

2.1 Definition of T-Functions

T-functions are a new family of primitives that has recently emerged, especially for stream ciphers design. Initially introduced by Klimov and Shamir in 2002 [7], a T-function maps an n-bit input to an n-bit output, where the i-th bit of the output depends only on the i Least Significant Bits (LSB) of the input. T actually stands for "Triangular", as illustrated in Figure 1.

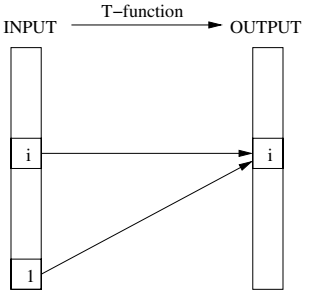


Fig. 1. General structure of a T-function

More precisely, let $x = (x_1, \ldots, x_n)$ be the input of the T-function F, and $y = (y_1, \ldots, y_n)$ its output. The definition implies that y_i must be computable only from $(x_1, \ldots, x_i)$. This idea is quite natural, since most usual software operations are T-functions, like boolean operations (IF, AND, XOR) and arithmetic operations (+, *) taken modulo a power of 2. Besides, T-functions can be composed, while still preserving the Triangular property. Also, provided n is equal to the word size of the underlying processor, implementation of such T-functions in software can be made very efficient.

For practical purpose, it is customary to use a **multi-word T-function**. It is a mapping applied to m words of n bits, which has the property that the i-th bits of each output depends only on the i LSB's of each input. Multi-word T-functions fit better on 32-bit or 64-bit architectures than single-word T-functions.

There has also been some work on suitable properties for T-functions in stream ciphers (like invertibility and cycle length) [8]. More recently, it was suggested to use T-functions to build MDS nonlinear mapping in block ciphers, and self-synchronizing hash functions [9].

2.2 Compression Functions

Typically, a hash function H maps an arbitrary input $M \in \{0,1\}^*$ to a fixed-size output $H(M) \in \{0,1\}^n$. A mode of operation is often provided, in order to handle arbitrary inputs and to extend the domain of a smaller function, also called **compression function** and noted h, which works on fixed-size inputs. Figure 2 summarizes this construction method, also known as Merkle-Damgård construction.

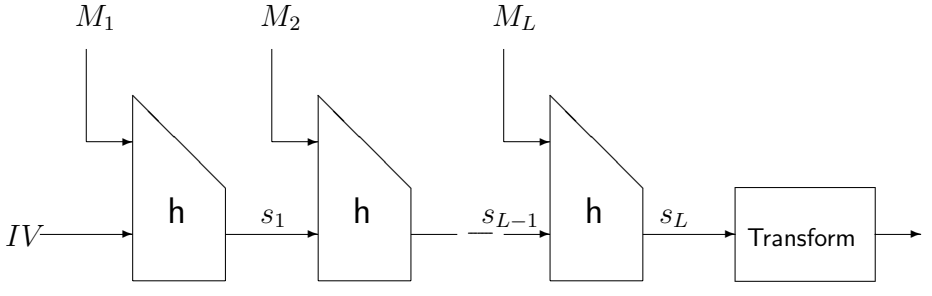


Fig. 2. Description of the hash function H

While there are theoretical background for using it, the final transform is often omitted, and the last chaining variable s_L becomes the hash of the message M. IV is a n-bit constant, which is part of the specifications of the hash function, while the variables $(M_1 \dots M_L)$ (of length t bits each) are the "chunks" of the message. A padding may need to be specified to avoir some attacks and to handle messages whose length is not a multiple of t.

To summarize, the compression function h maps an input of length $t+n$ bits to an output of n bits. A natural idea is to **build a compression function based on a T-function** F. By definition, F has the same input and output domain, so one needs to find a way to decrease the output length : there are several natural ideas to achieve it.

2.3 The Truncation Method

In stream cipher designs, T-functions are thought of as mappings such that knowing the Most Significant Bits (MSB) of the output reveals little information about the input. Roughly, the MSB's are used as the keystream in a stream cipher design. This was the original idea by Klimov and Shamir [8,13], although more complex constructions have been preferred in practice [3,11,14].

In the context of compression function design, the natural idea to decrease the output size of the T-function is therefore to truncate the output's LSB. Call F the T-Function and $t + n$ its domain length. The t LSB's of the output of F are truncated in order to build the compression function h. The output size of h is indeed $t + n - t = n$ bits, as needed (also see Figure 3).

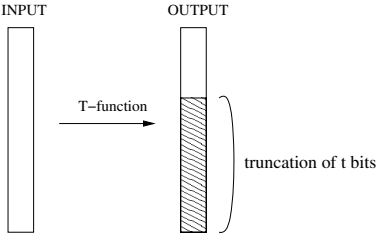


Fig. 3. The truncation method

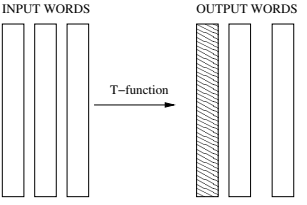


Fig. 4. The reduction method

This idea easily extends to multi-word T-functions. For instance, suppose that $t + n$ is a multiple of 32. Then F is a multi-word T-function operating on $l = (t + n)/32$ words of 32 bits each. To build a compression function h, the t/l LSB's of each output word would be truncated.

2.4 The Reduction Method

Another natural solution is to reduce the number of output words of a (multi-word) T-function. Therefore F becomes a mapping which has different input and output domain. For instance, if both t and n are multiples of 32, one could design a Triangular mapping which takes an input of $(t + n)/32$ words and outputs $n/32$ words only. Such a design can be built from scratch, or from a pre-existing T-function[1] (also see Figure 4).

A good example of this method is the MySQL dedicated hash function, where the input of h is of length 4 words (3 words from the chaining variable, and 1 word which is a chunk of the password) and the output length is 3 words. All operations in h are Triangular, therefore h itself is a T-function (see Section 5.2 for more details).

3 Analysis of the "Natural" Solutions

It is quite clear that all the solutions proposed above do not present a "pseudo-random" behavior. For instance, flipping only the most significant bit(s) of the input will modify only the most significant bit(s) of the output. However, it does not immediately extend to collision or preimage attacks. In this Section,

[1] In this case, it comes down to the truncation method described in Section 2.3, where entire output words are removed, instead of the LSB's of each word.

we exploit the structure of T-functions to propose generic collision and preimage attacks against all the "natural" constructions proposed in Section 2.

3.1 Attacking the Truncation Method

Let h be a compression function built from the truncation of a l-word T-function F. The input x of F is composed by l-bit values noted $x = (x_1 \dots x_{(t+n)/l})$ and its output y is composed by l-bit values noted $y = (y_1 \dots y_{(t+n)/l})$. By definition,

$$h(x) = y' = (y'_1 \dots y'_{n/l}) = (y_{(t/l)+1} \dots y_{(t+n)/l})$$

We describe a **pseudo-preimage attack against** h, *i.e.* for a random input challenge defining an output y' of h, we find an arbitrary input x such that $h(x) = y'$. It is called "pseudo-preimage" attack because there is no constraint on the n input bits which correspond to the chaining variable. Therefore, they are not necessarily equal to the IV of the hash function (see Figure 2). However, it is well-known that such attacks can be extended to preimage attacks against the whole hash function, using a meet-in-the-middle approach [12].

An attacker can compute a pseudo-preimage, *layer by layer*. The sketch of our attack against the truncation method is the following :

- Fix an arbitrary value for $x_1, x_2, \dots, x_{t/l}$ and fix $pos = (t/l) + 1$.
- For all the 2^l candidates of x_{pos}, compute y_{pos} (possible since F is triangular).
- Choose arbitrarily a candidate x_{pos} such that $y_{pos} = y'_{pos-(t/l)}$, increment pos and continue to the next layer.
- If no such candidate exists, return to the first step.
- If $pos = (t+n)/l$, the input $x = (x_1, \dots, x_{(t+n)/l})$ clearly satisfies $h(x) = y'$.

Since there are 2^l possible choices for x_{pos}, it is likely that one of the candidates matches the word $(pos - (t/l))$ of the challenge, at each step. Otherwise we draw at random another value of $x_1, x_2, \dots, x_{t/l}$ until a solution is eventually found. More precisely, we have a probability of P to find a valid candidate at one precise step, where:

$$P = 1 - \frac{\text{number of mappings where } y'_{pos-(t/l)} \text{ does not appear}}{\text{total number of mappings}} = 1 - \left(\frac{2^l - 1}{2^l}\right)^{2^l}.$$

Finally, **this algorithm costs about** $\frac{n}{l} \times 2^l \times P^{-\frac{n}{l}}$ **computation steps to find a pseudo-preimage** while the expected strength is 2^n. We show in Appendix A that $P^{-1/l} < \sqrt{2}$, which validates our attack since $P^{-1/l} < 2$.

Note that this attack can be utilized to find a pseudo-collision, i.e. two different inputs x, x' such that $h(x) = h(x')$. By choosing arbitrarily an input x and running the previously described attack for the challenge $y = h(x)$, we eventually get an input x' such that $h(x) = h(x') = y$. The complexity of this attack is the same as the preimage one while the expected strength is $2^{\frac{n}{2}}$, and we show in Appendix A that $P^{-1/l} < \sqrt{2}$.

3.2 Attacking the Reduction Method

The reduction methods suffer from the same weaknesses as the truncation method, as an attacker can compute a preimage layer by layer. Suppose that the input is composed of l words and the output of $l' < l$ words.

First, for all possible least significant bits of the input (2^l candidates), he computes the least significant bits of the output and searches for a match with the bits from the challenge. If a solution is found, he jumps to the next layer (*i.e.* the second least significant bits), etc .... Since $l > l'$, it is very likely that a solution exists at each step.

With the same reasoning as for the truncation method, we can conclude that **this algorithm costs about $\frac{n}{l'} \times 2^{l'} \times P^{-\frac{n}{l'}}$ computation steps to find a pseudo-preimage**, where:

$$P = 1 - \left(\frac{2^{l'} - 1}{2^{l'}} \right)^{2^l}.$$

Using the proof in Appendix A, we have $P^{-1/l'} < 2$. A good illustration is given in the next Sections with the MySQL dedicated hash function, where $l = 4$ and $l' = 3$.

This attack also leads to a pseudo-collision one, in the same manner as for the truncation method. The complexity of this attack is the same as the preimage one, and $P^{-1/l} < \sqrt{2}$ is a direct consequence of the proof in Appendix A .

3.3 Some Possible Strengthening

All these attacks result from the following fact : due to their triangular structure, T-functions allow to attack some parts independently. More generally, functions for which some parts can be separated independently are bad candidates for hash functions. For example, this problem is present when one wants to build a multiple-length hash function from simple-length primitives [19]. Naturally, one can think of some possible strengthening in order to thwart such attacks :

- the **feed-forward** consists in XORing some input bits to the output of the compression function. It is typically used to break the invertibility of block cipher-based compression functions. The feed-forward is also triangular, so it would not help here.
- the **folded feed-forward** consists in exchanging the least and most significant part of the input before the feed-forward. Such folding has already been used in stream ciphers design to strengthen the output function [6]. Such solutions would require a separate analysis, but we believe trade-off attacks could allow to invert the compression function for less than 2^n work.
- **tweaking** the T-functions to make the most significant bits very difficult to invert, when the least significant bits are hidden. However, this requires major changes to T-function designs.

3.4 Summary

To summarize, most "natural" constructions based on T-function turn out to be weak regarding collision and preimage attacks. The attacks we described are very simple, but they are generic : they are direct consequences of the structure of T-functions. Despite this simplicity, the impact in practice can be devastating. As an illustration, we analyze the dedicated hash function used in MySQL password-based authentication mechanisms, whose compression function is triangular.

4 Application to (Old) MySQL Authentication Mechanisms

MySQL is a very popular database open source technology [16]. It was initially conceived (and is still maintained today) by a Swedish company called MySQL AB. Among many other things, MySQL contains some security mechanisms. They are used to authenticate a client that wishes to connect to the database server. The client and the server initially share **a password** and the authentication is a challenge-response, based on the password only, without the use of any public-key cryptography. The server generates a fresh random challenge, generally called **the salt** and sends it over to the client. Upon reception, the client builds an authentication message (*aka* scrambled message) by mixing the salt and the password and sends its response to the server. Finally, the server recomputes the scrambled message, and compares this result with what he received from the client. Depending on this verification, he accepts or rejects the connection. This protocol is represented in Figure 5.

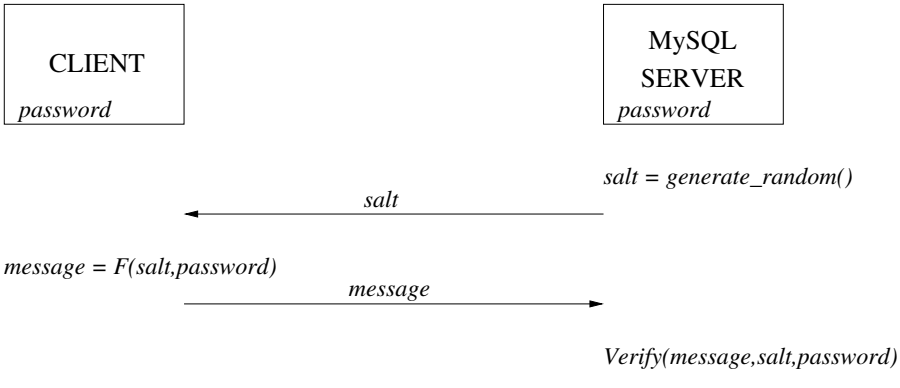


Fig. 5. Password-based Challenge-response Authentication in MySQL

Actually, the real construction is slightly different : indeed, one wants to avoid the loss of all user's passwords in case of intrusion on the server. Therefore, the server stores only a hash of each user's password, instead of the password itself.

A hash function H is specified for that purpose in the MySQL specifications. The function F is applied to $H(password)$ instead of the actual $password$, so what the client and the server both compute is actually :

$$message = F(salt, H(password))$$

As we can see, all an attacker needs to know in order to impersonate the legitimate client is the value of $H(password)$.

Until version 3.23, MySQL used a dedicated hash function H and a dedicated **scrambling** function F. H is a usual iterated construction based on a compression function, while F is a pseudo-random number generator, whose seed is derived from the salt and $H(password)$. The last version using these old mechanisms (3.23.58) was released in September, 2003. Since then, new versions have been developed (4.0 and above), for which they were replaced by constructions based on the popular hash function SHA-1 [17]. Still, the old security algorithms continue to be implemented, to ensure compatibility with old versions. The description of H and F can be obtained on the MySQL website [15], by downloading the source code (in particular, see the `password.c` file). A good informal description of the protocol can also be found at the address `http://www.redferni.uklinux.net/mysql/`.

The old MySQL password algorithms (those used in versions 3.23 and earlier) have already been studied in 2002 [1]. An attack against the F function was proposed, which allows an attacker to recover $H(password)$, provided he intercepts about 10 pairs $(salt, message)$. This is sufficient for practical purpose, which motivated the introduction of new security algorithms. However, **studying the dedicated hash function remains an open research topic**.

In this paper, we propose a preimage attack against this hash function. Although this was not apparently a design criteria, we observe that its compression function is based on a T-function and we apply the attacks described in Section 3. First, we describe the MySQL authentication mechanisms. Then we show an attack that finds preimages for the compression function, and finally we perform a meet-in-the-middle attack against the full hash function. This allows to mount a **square-root dictionary attack** : a password with entropy of n bits is recovered with $2^{n/2}$ computation steps. This attack has been implemented in practice on a standard PC and takes a few seconds, independently of the password length : if the password is too long, an "equivalent" password (i.e. leading to the same hash value) will be retrieved instead.

5 Description of the Mechanisms

5.1 The Old MySQL Hash Function

The old MySQL hash function H takes an input of arbitrary size and outputs two words of 30 bits each. Therefore it is a mapping

$$H : \{0,1\}^* \longrightarrow \{0,1\}^{60}$$

In practice, the input is a password which is split into a sequence of n bytes. No padding is specified, so the input length must be a multiple of 8 bits. H is an iterated hash function. At the i-th iteration, it updates a chaining variable s^i of size 96 bits (the concatenation of 3 words of 32 bits which are called n_1^i, n_2^i and add^i: $s^i = n_1^i||n_2^i||add^i$), by

$$s^i = h(s^{i-1}, c^i)$$

As usual, h is called the **compression function** and c^i is the i-th byte of the password. The initial value s^0 consists of the following constants :

$$\begin{aligned} n_1^0 &= 1345345333 \\ n_2^0 &= 305419889 \\ add^0 &= 7 \end{aligned}$$

When all password bytes have been used, after k iteration of the compression function, the final value of the state is truncated from 96 to 60 bits, by keeping only the 30 least significant bits of n_1^k and n_2^k. These 60 bits constitute the hash of the password. Also refer to Figure 2 for a description of H, where the final transform is a truncation from 96 to 60 bits.

Due to the limited output size, H can offer at most a security of 2^{30} regarding collision attacks and 2^{60} regarding preimage attacks.

5.2 The Compression Function

The compression function h is a mapping

$$h : \{0,1\}^{96} \times \{0,1\}^8 \longrightarrow \{0,1\}^{96}$$

which has the property to combine only triangular operations. More precisely, it mixes basic arithmetic and logical operations :

$$\begin{aligned} n_1^i &= n_1^{i-1} \oplus (((n_1^{i-1} \wedge 63) + add^{i-1}) \cdot c^i + (n_1^{i-1} \ll 8)) \\ n_2^i &= (n_2^{i-1} \ll 8) \oplus n_1^{i-1} \\ add^i &= add^{i-1} + c^i \end{aligned}$$

where the additions are taken modulo 2^{32} and $\ll 8$ denotes a shift by 8 bits to the left[2].

Looking at the chaining variable and the password chunk c^i as input words, this compression function follows exactly the **reduction method** introduced in Section 2.4, with input domain of 4 words and output domain of 3 words. There is a slight technical difference here, since the chaining variable words have length 32 bits, while the password chunk has length of 8 bits only. But this does not change fundamentally the analysis.

[2] This is equivalent to a multiplication by $2^8 = 256$, modulo 2^{32}.

5.3 The Scrambling Function

The scrambling function F operates on a random value *salt* (which is a sequence of bytes, like the password) and on the hash of the password $H(password)$, by

$$message = F(salt, H(password))$$

Actually, F can be seen as a Pseudo-Random Number Generator (PRNG) : its *seed* is derived by :

$$seed = H(salt) \oplus H(password)$$

The scrambled message is then generated from several consecutive output bits of the PRNG. More precisely, *seed* is represented as two words of 30 bits each : $seed_1$ and $seed_2$. The update works as follows :

$$seed'_1 = (3 \cdot seed_1 + seed_2) \bmod (2^{30} - 1)$$
$$seed'_2 = (seed_1 + seed_2 + 33) \bmod (2^{30} - 1)$$

After each update, a pseudo-random output is extracted by :

$$output = \left\lfloor \frac{seed'_1}{34638622} \right\rfloor \in \{0, \ldots, 30\}$$

This output is added to 64 and mapped to a printable character using the ASCII table. This operation is performed l_s times where l_s is the length (in bytes) of the salt. The typical length is $l_s = 8$, although other length (up to $l_s = 20$) are handled by the protocol. A final masking is applied : the random number generator is run one extra round, and the resulting output byte (called *mask*) is XORed to every byte of the message.

The result is the authentication message which is sent by the client to the MySQL server. Depending on the version of MySQL, two security protocols (protocols 9 and 10) are possible. If protocol 9 is used, the scrambling function is slightly weakened : the entropy of the initial state is reduced and the final masking is removed. In this paper, we focus only on protocol 10, *i.e.* the "full" scrambling function, and we assume that $l_s = 8$. The attack would easily translate to situations where $l_s > 8$ or where the "weakened" version is used.

5.4 The New Construction

In more recent MySQL versions (4.0 and above), all these dedicated mechanisms have been replaced by a new scrambling function $F(salt, password)$ based on SHA-1 [17].

$$v = \text{SHA-1}(\text{SHA-1}(password))$$
$$F(salt, password) = v \oplus \text{SHA-1}(salt||v)$$

where $||$ denotes the concatenation. The security of this new mechanism relies on the security of SHA-1. New results have recently been published [22], however they concern collision resistance, while the problem here is essentially preimage resistance.

5.5 The Cryptanalysis Scenario

First, we suppose that an attacker wants to attack the old authentication protocol of MySQL. He listens to communications between the client and the server, and learns pairs $(salt, message)$. His first goal is to retrieve the value of $H(password)$. This information is sufficient to later impersonate the legitimate client. We call it the **unscrambling attack**. It basically comes down to breaking the PRNG, given several output bits. Such an attack was already proposed in [1]. We further analyze the security of the MySQL scrambling function in Appendix B.

If for some reason, the attacker needs to find the actual client's password, he needs to **break the one-wayness** of H, *i.e.* to retrieve *password* from $H(password)$. This scenario can arise after the unscrambling attack, or in case of intrusion on the database server : in this case, the hash of each user's password is compromised.

6 Preimage Attacks Against the MySQL Dedicated Hash Function

In this section, we show how to break the one-wayness of the MySQL dedicated hash function. We are given a hash of the user password and retrieve the password almost instantly. This attack has been implemented on a regular PC (Pentium IV, 2.5 Ghz with 2Gb of RAM), and takes negligible amount of time.

We first show how to compute pseudo-preimage against the compression function, following the framework given in Section 3. Then we perform a *meet-in-the-middle* attack against the whole hash function.

6.1 Breaking the Compression Function

The compression function h is described by the following equations :

$$\begin{aligned} n_1^i &= n_1^{i-1} \oplus (((n_1^{i-1} \wedge 63) + add^{i-1}) \cdot c^i + (n_1^{i-1} \ll 8)) \\ n_2^i &= (n_2^{i-1} \ll 8) \oplus n_1^{i-1} \\ add^i &= add^{i-1} + c^i \end{aligned}$$

where the incoming chaining value $(n_1^{i-1}, n_2^{i-1}, add^{i-1})$ and the output (n_1^i, n_2^i, add^i) are composed with 3 words of length 32 bits and c^i is the password character to be hashed.

This compression function follows the reduction method, introduced in Section 2.4. We apply the pseudo-preimage attack described in Section 3.2 : the general idea is to proceed *layer by layer*. Actually, we can directly write the method that allows to compute all preimages corresponding to a given challenge, noted (n_1', n_2', add'). Each preimage is a pair $\{c, (n_1, n_2, add)\}$ that maps to the

challenge through h. We start by enumerating all possible password characters c. Then, we do the following for each c :

- First, observe that $add = add' - c$
- Guess the 6 LSB of n_1. We get that

$$n_1 \wedge 255 = (n'_1 \wedge 255) \oplus (c \cdot ((n_1 \wedge 63) + add))$$

  So we obtain the 8 LSB of n_1. This must match the previous guess on the 6 LSB, otherwise we reject it. In general, only 1 guess out of $2^6 = 64$ remains at this point.
- The value of $x = ((n_1 \wedge 63) + add) \cdot c$ is known and :

$$n_1 = n'_1 \oplus (x + (n_1 \ll 8))$$

  which allows to derive the value of n_1 byte by byte
- Finally, we derive n_2 by :

$$n_2 \ll 8 = n'_2 \oplus n_1$$

There is no way to determine the 8 MSB of n_2 since they are not used in the compression function. This inversion of h has been implemented on a regular PC and takes a negligible amount of time. In general, the solution obtained is unique, for each guess on c.

6.2 Application to the Full Hash: Meet-in-the-Middle

When it is possible to compute preimages for the compression function, it is well known that a *meet-in-the-middle attack* allows to extend the attack to the full hash (see [21] for instance). Our basic assumption is that the password is an unknown vector of 8 bytes, *i.e.* its entropy is 64 bits. We observe that if the password is longer, it is very likely that we will found an alternative 8-byte password that maps to the correct hash, since the output size is only 60 bits. Then, the framework of the attack is the following :

- Split the search space in two halves (of 4 characters each for instance)
- Start from the initial chaining value, and apply 4 times the compression function, for each possible sequence of 4 characters.
- Start from the password hash and invert 4 times the compression function with the previously explained technique, for each possible sequence of 4 characters.
- Find a match between the forward computation and the backward computation.

The complexity of this attack is only $(2^8)^4 = 2^{32}$ in time and memory. This represents a **square-root dictionnary attack**. Indeed, the complexity is only the square-root of the size of the search space.

6.3 Analysis

We implemented the previous attack, but several modifications were introduced compared to the theoretical analysis :

- First, we did not consider a character as an 8-bit unknown, but we restricted our analysis to a subset of about 100 characters (alphanumerical, plus some usual symbols) that are easily rechable from the keyboard.
- Secondly, instead of splitting the 8 characters in $4+4$, we split it into $3+5$. We stored in memory the candidates for the backward computation (about 100^3 candidates), then we sorted this list (using quick-sort). Finally, we enumerated the list of 100^5 "forward candidates". This is a more reasonable trade-off, since memory requirements are more difficult to satisfy in implementations than time requirements.
- Finally, there is no use in identifying the candidates using the whole n_2 value, since its MSB are not used by the compression function. As a result, we slightly decrease the memory requirements, but we also increase the number of matches we obtain.

Typically, our implementation takes less than one second to retrieve a password randomly chosen by the user. If the password has length > 8 characters, we generally find many alternative candidates.

To give an example, if we pick the password "MySQL123". The hash of the password is $(n_1^k \wedge K = \texttt{0x1b03947c}, n_2^k \wedge K = \texttt{0x1d6d177b})$, with $K =$ `0x3fffffff`. Exploring all passwords of length up to 8, we found two solutions that map to this hash : "MySQL123" and "RGp0mA23". The second candidate is an alternative password that could be later substituted to the original one.

7 Conclusion

We considered several natural solutions to build hash functions based on T-functions, a new and popular primitive in symetric-key designs. Following recent cryptanalysis results, new ideas might be necessary for constructing hash functions, but our results show that **the "natural" solutions based on T-functions fail to provide a satisfactory level of security**. While the attacks we describe are very simple, their practical effect can be devastating.

As an illustration, we focused on the old MySQL password-based authentication mechanisms. Until 2003, they included a dedicated hash function for which no analysis had yet been published. We observed that its compression function is based on a T-function and therefore we apply our generic analysis. We described a square-root dictionnary attack to find preimages. It retrieves the user's password (or an equivalent one if the password is too long from a hash value, in less than a second). These results raise some interesting perspectives about the design of T-function-based hash functions.

Acknowledgement

The authors would like to thank the anonymous referees and Yannick Seurin for their valuable comments and for spotting an error in the original complexity analysis.

References

1. I. Arce, A. Azubel, E. Kargieman, G. Richarte, C. Sarraute, and A. Waissbein. An attack on the MySQL authentication protocol. 2002. Technical report from Core Security Technologies.
2. J-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In V. Shoup, editor, *Advances in Cryptology – Crypto'05*, volume 3621 of *Lectures Notes in Computer Science*, pages 430–448. Springer, 2005.
3. J. Hong, D. Lee, Y. Yeom, and D. Han. A New Class of Single Cycle T-functions. In H. Gilbert and H. Handschuh, editors, *Fast Software Encryption – 2005*, volume 3557 of *Lectures Notes in Computer Science*, pages 68–82. Springer, 2005.
4. A. Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In M. Franklin, editor, *Advances in Cryptology – CRYPTO'04*, volume 3152 of *Lectures Notes in Computer Science*, pages 306–316. Springer, 2004.
5. J. Kelsey and B. Schneier. Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work. In R. Cramer, editor, *Advances in Cryptology – Eurocrypt'05*, volume 3494 of *Lectures Notes in Computer Science*, pages 474–490. Springer, 2005.
6. A. Klimov. *Applications of T-functions in Cryptography.* PhD thesis, Weizmann Institute of Science, 2004. http://www.wisdom.weizmann.ac.il/ ask/.
7. A. Klimov and A. Shamir. A New Class of Invertible Mappings. In B. Kaliski, Ç. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems (CHES) – 2002*, volume 2523 of *Lectures Notes in Computer Science*, pages 470–483. Springer, 2002.
8. A. Klimov and A. Shamir. Cryptographic Applications of T-functions. In M. Matsui and R. Zuccherato, editors, *Selected Areas in Cryptography – 2003*, volume 3006 of *Lectures Notes in Computer Science*, pages 248–261. Springer, 2004.
9. A. Klimov and A. Shamir. New Applications of T-functions in Block Ciphers and Hash Functions. In H. Gilbert and H. Handschuh, editors, *Fast Software Encryption – 2005*, volume 3557 of *Lectures Notes in Computer Science*, pages 18–31. Springer, 2005.
10. L. Knudsen. SMASH - A Cryptographic Hash Function. In H. Gilbert and H. Handschuh, editors, *Fast Software Encryption – 2005*, volume 3557 of *Lectures Notes in Computer Science*, pages 228–242. Springer, 2005.
11. S. Künzli, P. Junod, and W. Meier. Distinguishing Attacks on T-Functions. In E. Dawson and S. Vaudenay, editors, *International Conference on Cryptology in Malaysia (MyCrypt 2005)*, volume 3715 of *Lectures Notes in Computer Science*, pages 2–15. Springer, 2005.
12. A. Menezes, P. van 0orschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.
13. J. Mitra and P. Sarkar. Time-Memory Trade-Off Attacks on Multiplications and T-functions. In P. Lee, editor, *Advances in Cryptology - Asiacrypt'04*, volume 3329 of *Lectures Notes in Computer Science*, pages 468–482. Springer, 2004.

14. F. Muller and T. Peyrin. Linear Cryptanalysis of the TSC Family of Stream Ciphers. In B. K. Roy, editor, *Advances in Cryptology - Asiacrypt'05*, volume 3788 of *Lectures Notes in Computer Science*, pages 373–394. Springer, 2005.
15. MySQL Downloads. See `http://dev.mysql.com/downloads`.
16. MySQL Website. See `http://www.mysql.com`.
17. National Institute of Standards and Technology (NIST). Secure Hash Standard FIPS Publication 180-1, April 1995. Available at `http;//www.itl.nist.gov/fipspubs/fip180-1.htm`.
18. J. Patarin and P. Camion. Design of Near-Optimal Pseudorandom Functions and Pseudorandom Permutations in the Information-Theoretic Model. Cryptology ePrint Archive, Report 2005/135, 2005. `http://eprint.iacr.org/`.
19. T. Peyrin, H. Gilbert, F. Muller, and M. Robshaw. Combining Compression Functions and Block Cipher-Based Hash. In *Advances in Cryptology – Asiacrypt'06*, 2006. To appear.
20. N. Pramstaller, C. Rechberger, and V. Rijmen. Breaking a New Hash Function Design Strategy Called SMASH. In B. Preneel and S. E. Tavares, editors, *Selected Areas in Cryptography – 2005*, volume 3897 of *Lectures Notes in Computer Science*, pages 233–244. Springer, 2005.
21. B. Preneel. *Analysis and design of cryptographic hash functions*. PhD thesis, Katholieke Universiteit Leuven, 1993.
22. X. Wang, Y. Yin, and H. Yu. Finding Collisions in the Full SHA1. In V. Shoup, editor, *Advances in Cryptology – Crypto'05*, volume 3621 of *Lectures Notes in Computer Science*, pages 17–36. Springer, 2005.
23. X. Wang and H. Yu. How to Break MD5 and Other Hash Functions. In R. Cramer, editor, *Advances in Cryptology – Eurocrypt'05*, volume 3494 of *Lectures Notes in Computer Science*, pages 19–35. Springer, 2005.

A Complexity Analysis

We want to show that $P^{-1/l} < \sqrt{2}$, with $P = 1 - \left(\frac{2^l-1}{2^l}\right)^{2^l}$ and $l \geq 1$.

First, we note $f(x) = \frac{2^{x/2}-1}{2^{x/2}}$ and $g(x) = \left(\frac{2^x-1}{2^x}\right)^{2^x}$. With $x \geq 1$, both functions are increasing and if $x \longrightarrow +\infty$, we have:

$$\begin{aligned} f(x) &\longrightarrow 1 \\ g(x) &\longrightarrow \tfrac{1}{e} \end{aligned}$$

Since $f(1) > g(1)$ and $f(2) > \frac{1}{e}$, we conclude that $\frac{2^{l/2}-1}{2^{l/2}} > \left(\frac{2^l-1}{2^l}\right)^{2^l}$, with $l \geq 1$. Thus we have:

$$\begin{aligned} & \tfrac{2^{l/2}-1}{2^{l/2}} && > && \left(\tfrac{2^l-1}{2^l}\right)^{2^l} \\ \Longrightarrow \quad & 2^{l/2}-1 && > && 2^{l/2} \cdot \left(\tfrac{2^l-1}{2^l}\right)^{2^l} \\ \Longrightarrow \quad & 1 && < && 2^{l/2} \cdot (1 - \left(\tfrac{2^l-1}{2^l}\right)^{2^l}) \\ \Longrightarrow \quad & 1 && < && 2^{l/2} \cdot P \\ \Longrightarrow \quad & P^{-1/l} && < && \sqrt{2}. \end{aligned}$$

B Unscrambling

Since the scrambling function is a PRNG, the situation here is close to the **cryptanalysis of a stream cipher**, although the number of available output bits from the PRNG is very limited here. An analysis of the PRNG was already given in [1], which works well in practice. In this section, we propose a more systematic approach to unscramble.

The scrambled message is supposed to have length 8 bytes, and is noted $m = (m_0, \ldots, m_7)$. The intermediate values of the state are noted $(seed_1^i, seed_2^i)$ for $i = 1 \ldots 8$. If we retrieve the initial state, we obtain $H(password)$ by simply XORing $H(salt)$. Unfortunately the problem is under-defined : the initial seed is 60 bits long, while the scrambled message takes only 31^9 possible values. So, in theory, there are :

$$\frac{2^{60}}{31^9} \simeq 2^{15.41}$$

solutions. Several pairs $(salt, message)$ are therefore necessary to determine uniquely $H(password)$.

B.1 Unmasking

The scrambled message m is built from 9 consecutive outputs that we note $O_0, \ldots, O_8$. We have $m_i = O_i \oplus O_8$ for $i \in \{0, \ldots, 7\}$. We first want to find O_8 in order to unmask the real PRNG outputs. We use an important flaw in the underlying PRNG, *i.e.* the fact that **all output sequences are not equally likely**. To observe this phenomenon, we look at the state update equations :

$$seed_1^{t+1} = (3 \cdot seed_1^t + seed_2^t) \bmod (2^{30} - 1)$$
$$seed_2^{t+1} = (seed_1^{t+1} + seed_2^t + 33) \bmod (2^{30} - 1)$$

We **focus on the interval where** $seed_1$ **and** $seed_2$ **lie modulo** 34638622, and define :

$$N_1^t = \left\lfloor \frac{seed_1^t}{34638622} \right\rfloor \text{ and } N_2^t = \left\lfloor \frac{seed_2^t}{34638622} \right\rfloor$$

which are integers in the interval $[0, 30]$. Knowledge of N_1 and N_2 before the update allows to predict with good probability their value after the update. Indeed, :

$$3 \cdot N_1^t + N_2^t \leq \frac{3 \cdot seed_1^t + seed_2^t}{34638622} < 3 \cdot N_1^t + N_2^t + 4$$

Besides

$$N_1^{t+1} = \left\lfloor \frac{(3 \cdot seed_1^t + seed_2^t) \bmod 2^{30} - 1}{34638622} \right\rfloor$$
$$= \left\lfloor \frac{3 \cdot seed_1^t + seed_2^t}{34638622} \bmod 31 \right\rfloor$$

So we have only 4 candidates, *i.e.* $N_1^{t+1} = (3 \cdot N_1^t + N_2^t + i) \bmod 31$, with $i = 0, 1, 2$ or 3, instead of 31 candidates. Similarly, we can identify a reduced number of candidates for N_2^{t+1}, *i.e.* $(3 \cdot N_1^t + 2 \cdot N_2^t + i) \bmod 31$, with $i = 0, 1, 2, 3$ or 4.

The same analysis can be recursively applied to all iterations of h. Since we know the value of $O_i = N_1^i$ we can eliminate an important fraction of the initial guesses, at each step. For $i = 1$, the probability for a guess to be kept is 4 out of 31, since we have 4 candidates for O_1. It is easy to see that when we iterate the process, we get 6 candidates for O_2, ..., and 18 candidates for O_8. Therefore, for each initial guess (31 choices for O_8 and 31 choices for N_2^0), the O_i's must all be among this list of candidates, so the probability p for an initial guess to be accepted is :

$$p = \frac{4}{31} \times \frac{6}{31} \times \ldots \times \frac{18}{31} = 2^{-13.16}$$

This is small enough, since there are $31^2 \simeq 2^{9.9}$ initial guesses. So the good value of the mask O_8 should be uniquely identified by our method. We implemented this attack (see Section B.4) and indeed managed to retrieve the correct mask with high probability. In the (rare) case of a false alarm, we just do not exploit the message, and repeat the attack with another value of the salt.

B.2 Inverting the PRNG

After unmasking the scrambled message, we now have 9 consecutive outputs of the random number generator $(O_0, \ldots, O_8)$ for which we search all the possible initial seeds. Our attack is exactly similar to the unmasking attack, *i.e.* we use **interval bounding techniques**. The only difference is that we use thiner intervals, in order to learn the initial state with better precision. So we wish to split the interval $[0, 2^{30} - 1]$ into m intervals of length l. Luckily, $2^{30} - 1$ is smooth :

$$2^{30} - 1 = 3^2 \times 7 \times 11 \times 31 \times 151 \times 331$$

so we have many possible choices for m and l, such that $m \cdot l = 2^{30} - 1$. Then, we guess the initial value of

$$N_1^t = \left\lfloor \frac{seed_1^0}{l} \right\rfloor \text{ and } N_2^t = \left\lfloor \frac{seed_2^0}{l} \right\rfloor$$

which are now integers in the interval $[0, m]$ so there are m^2 possible guesses. We write :

$$3 \cdot N_1^t + N_2^t \leq \frac{3 \cdot seed_1^t + seed_2^t}{l} < 3 \cdot N_1^t + N_2^t + 4$$

and we observe that $N_1^{t+1} = \left\lfloor \frac{3 \cdot seed_1^t + seed_2^t}{l} \bmod m \right\rfloor$. Therefore we obtain 4 candidates after the first update, ..., and 18 candidates after the 8-th update (exactly like in the previous section).

If l is chosen small enough, these candidates all lie in the same interval modulo 34638622, so they correspond to a unique output of the PRNG. Hence, we can

eliminate a fraction $31^9 \simeq 2^{44.59}$ of the initial guesses by simply testing if they correspond to a valid output sequence. Roughly, we need $18 \cdot l \ll 346386222$ for this to work. A good choice is to pick $m = 3243933$ intervals of length $l = 331$. The number of initial guesses is $m^2 = 2^{43.26}$. In theory, this is sufficient to obtain a unique candidate since $2^{44.59} > 2^{43.26}$.

B.3 Early-Abort

Instead of implementing the basic attack, we used a simple refinement, which consists in **repeatedly refining the interval length** l. We start by working with $m = 31$ and $l = 34638622$ (like in the unmasking attack). We obtain a list of candidate intervals of length l where $seed_1^0$ and $seed_2^0$ may lie. Then, we increase m to $m' > m$, such that m' remains a divisor of $2^{30} - 1$.

We do not need to explore all the intervals of size $\frac{2^{30}-1}{m'}$, but only $\frac{m'}{m}$ sub-intervals, for each solution that was identified with parameters m and l. Thanks to this trick, we avoid an enumeration that would cost $(m')^2$. This **early-abort strategy** allows to dramatically improve the running time of our attack.

B.4 Experimental Results

We implemented this interval bounding technique on a regular desktop PC. We started by splitting the search space into 31 intervals of length 34636833 (unmasking attack). Then, we gradually reduced the size of the intervals. Surprisingly, our experiments revealed some important deviations with the theoretical predictions. Indeed, the number of candidates at all intermediate stages (and in particular at the end of the analysis) is much higher than expected (see Table 1).

Table 1. Experimental results averaged over 1000 different passwords

Interval length l	Number of intervals m	Average number of solutions
34636833	31	1.677
104643	10261	$723.559 \simeq 2^{9.499}$
693	1549411	$138905.785 \simeq 2^{17.084}$
63	17043521	$5788944.172 \simeq 2^{22.465}$
9	119304647	$\simeq 2^{27.842}$
3	357913941	$\simeq 2^{30.983}$
1	$2^{30} - 1$	$\simeq 2^{32.5911}$

In the end, for an arbitrary output sequence, there are much more solutions for the initial seed than the theoretical prediction of :

$$\frac{2^{60}}{31^9} \simeq 2^{15.41}$$

while we observe on average $2^{32.5911}$ solutions. It means that there exist output sequences which are impossible, or with a small number of solutions. It would

be interesting to explore further on this phenomenon. However, our attack still works fine, despite the unexpected deviations : **we manage to enumerate the $2^{32.5911}$ candidates in several minutes on a regular PC**, thanks to the early-abort strategy.

B.5 Analysis

To identify the actual hash from the $2^{32.5911}$ candidates, it is necessary to analyze several pairs $(salt, message)$. Two pairs are not sufficient, because it is very likely to observe collisions between two lists of size $2^{32.5911}$ containing 60-bit values. In theory, 3 or 4 pairs should be sufficient. However, in practice, there remains about 100 candidates, even if we consider 10 pairs. Actually, the same phenomenon was observed in [1].

The underlying reason is that the LSB of $H(password)$ have little influence on the scrambling function. Therefore, even when our prediction on the LSB is incorrect, we have still a good chance to compute the correct scrambled message. On the one hand, this property is a problem for the attacker, since he needs to compute preimages for all candidates in order to retrieve the password. On the other hand, it also means that the attacker has good chance to authenticate correctly even when $H(password)$ is invalid.

Our attack has been implemented. It takes several minutes on a typical PC, while the attack described in [1] requires about one hour running time on a PC. This attack retrieves a hash of the user's password. While this may be enough for practical purpose, it is interesting to analyze the security of the hash function.

Collision Search Attack for 53-Step HAS-160

Hong-Su Cho[1], Sangwoo Park[2], Soo Hak Sung[3], and Aaram Yun[2]

[1] Graduate School of Information Security, Korea University
1, 5-Ga, Anam-dong, Seongbuk-gu, Seoul 136-701, Korea
karma3432@korea.ac.kr
[2] National Security Research Institute,
161 Gajeong-dong, Yuseong-gu, Daejeon 305-350, Korea
{aaram, psw}@etri.re.kr
[3] Department of Computing information & mathematics, Paichai University,
426-6 Doma-dong, Seo-gu, Daejeon 302-735, Korea
sungsh@mail.pcu.ac.kr

Abstract. HAS-160 is a cryptographic hash function which is designed and used widely in Korea. In ICISC 2005, Yun et al. presented a collision search attack for the first 45 steps of HAS-160. In this paper, we extend the result to the first 53 steps of HAS-160. The time complexity of the attack is about 2^{55}.

Keywords: collision search attack, HAS-160, cryptanalysis, hash function.

1 Introduction

One of the main impetus for recent vitalization of hash function research is due to works of Xiaoyun Wang et al.[3,4,5,6]. They produced collision search attacks for MD4, MD5, RIPEMD, HAVAL, SHA-0, and SHA-1, and developed powerful techniques for analyzing currently popular hash functions.

HAS-160 is a cryptographic hash function developed by Chae Hoon Lim et al. It is a Korean industry standard (TTAS.KO-12.0011/R1)[1], and used widely in Korea. The design of HAS-160 is influenced by SHA-1 and MD family hash functions, and it processes messages by 512-bit blocks and produces 160-bit outputs.

Since HAS-160 shares many features with MD family hash functions, the methods developed by Wang is also applicable to the analysis of HAS-160. In ICISC 2005, Yun et al.[7] applied Wang's methods to HAS-160 and found collisions for 45-step version of HAS-160 (out of the full 80 steps).

In this paper, we extend the results of [7], and produce a collision search attack for the first 53 steps of HAS-160. The time complexity of the attack is about 2^{55}, which would be feasible with supercomputers.

No significantly new technique was used other than those in [7], but by selecting message differences judiciously and constructing more complicated first round differential path, we were able to extend the result in [7] from 45 steps to 53 steps.

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 286–295, 2006.
© Springer-Verlag Berlin Heidelberg 2006

In the following, we will first describe HAS-160 briefly in Section 2, and present the collision search attack for 53-step HAS-160 in Section 3. In Section 4, we conclude our paper.

2 Description of HAS-160

HAS-160 is a Merkle-Damgård hash function, and its compression function uses 160-bit chaining values and 512-bit message blocks. The chaining values are represented by five 32-bit variables a, b, c, d, and e, and the 512-bit message block is represented by 16 32-bit words $(m_0, m_1, \ldots, m_{15})$.

HAS-160 applies 80 steps of simple transformations to its chaining values, starting from the IV. One round consists of 20 steps, therefore HAS-160 has 4 rounds in total. Let's denote by $(a_i, b_i, c_i, d_i, e_i)$ the chaining values (a, b, c, d, e) right after step i.

The initial chaining values (IV) are fixed and given as $(a_0, b_0, c_0, d_0, e_0)$= (`0x67452301`, `0xefcdab89`, `0x98badcfe`, `0x10325476`, `0xc3d2e1f0`).

The transformation at step i ($i = 1, \ldots, 80$) can be described as follows:

- $a_i \leftarrow a_{i-1}^{\lll s_{1,i}} + f_i(b_{i-1}, c_{i-1}, d_{i-1}) + e_{i-1} + x_i + k_i$
- $b_i \leftarrow a_{i-1}$
- $c_i \leftarrow b_{i-1}^{\lll s_{2,i}}$
- $d_i \leftarrow c_{i-1}$
- $e_i \leftarrow d_{i-1}$

where $+$ denotes the addition modulo 2^{32}, and $\lll$ the left bit rotation. Here the most significant bit is written at the leftmost position. Therefore $\lll$ moves bits toward the most significant bit (with the wrap-around from the most significant bit). $s_{1,i}$ and $s_{2,i}$ are amounts of bit rotations used in step i. $s_{1,i}$ is dependent on i mod 20 and the values are given in Table 1.

Table 1. The bit rotation s_1

step	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
s_1	5	11	7	15	6	13	8	14	7	12	9	11	8	15	6	12	9	14	5	13

$s_{2,i}$ are dependent on the round, i.e., it is constant in each round, and changes the value as the round is changed. The values are given as follows:

- Round 1: $s_2 = 10$
- Round 2: $s_2 = 17$
- Round 3: $s_2 = 25$
- Round 4: $s_2 = 30$

f_i and k_i represent a boolean function and a constant used in step i, respectively. They are given in the following table.

Table 2. Boolean functions and constants for HAS-160

Round	Step (i)	Boolean function (f_i)	Constant (k_i)
1	$1 \sim 20$	$(x \wedge y) \vee (\neg x \wedge z)$	0
2	$21 \sim 40$	$x \oplus y \oplus z$	0x5a827999
3	$41 \sim 60$	$(x \vee \neg z) \oplus y$	0x6ed9eba1
4	$61 \sim 80$	$x \oplus y \oplus z$	0x8f1bbcdc

Finally, x_i represents the message word used in step i. We have initially given 16 message words $m_0, \ldots m_{15}$. For each round, there are 20 steps and HAS-160 uses 20 message words, one for each step. For this, HAS-160 produces four additional message words out of the original 16 message words, by taking XORs of four of the original message words. We may call the additionally produced words 'dependent words', and the original 16 words 'independent words'. The words are given in Table 3.

Table 3. Message scheduling for HAS-160

i	Round 1	Round 2	Round 3	Round 4
1	$m_8 \oplus m_9$ $\oplus m_{10} \oplus m_{11}$	$m_{11} \oplus m_{14}$ $\oplus m_1 \oplus m_4$	$m_4 \oplus m_{13}$ $\oplus m_6 \oplus m_{15}$	$m_{15} \oplus m_{10}$ $\oplus m_5 \oplus m_0$
2	m_0	m_3	m_{12}	m_7
3	m_1	m_6	m_5	m_2
4	m_2	m_9	m_{14}	m_{13}
5	m_3	m_{12}	m_7	m_8
6	$m_{12} \oplus m_{13}$ $\oplus m_{14} \oplus m_{15}$	$m_7 \oplus m_{10}$ $\oplus m_{13} \oplus m_0$	$m_8 \oplus m_1$ $\oplus m_{10} \oplus m_3$	$m_{11} \oplus m_6$ $\oplus m_1 \oplus m_{12}$
7	m_4	m_{15}	m_0	m_3
8	m_5	m_2	m_9	m_{14}
9	m_6	m_5	m_2	m_9
10	m_7	m_8	m_{11}	m_4
11	$m_0 \oplus m_1$ $\oplus m_2 \oplus m_3$	$m_3 \oplus m_6$ $\oplus m_9 \oplus m_{12}$	$m_{12} \oplus m_5$ $\oplus m_{14} \oplus m_7$	$m_7 \oplus m_2$ $\oplus m_{13} \oplus m_8$
12	m_8	m_{11}	m_4	m_{15}
13	m_9	m_{14}	m_{13}	m_{10}
14	m_{10}	m_1	m_6	m_5
15	m_{11}	m_4	m_{15}	m_0
16	$m_4 \oplus m_5$ $\oplus m_6 \oplus m_7$	$m_{15} \oplus m_2$ $\oplus m_5 \oplus m_8$	$m_0 \oplus m_9$ $\oplus m_2 \oplus m_{11}$	$m_3 \oplus m_{14}$ $\oplus m_9 \oplus m_4$
17	m_{12}	m_7	m_8	m_{11}
18	m_{13}	m_{10}	m_1	m_6
19	m_{14}	m_{13}	m_{10}	m_1
20	m_{15}	m_0	m_3	m_{12}

3 Collision Search Attack

3.1 Notations and Conventions

We would like to find a collision pair (M, M') for 53-step HAS-160, that is, a pair of different messages M, M' with identical hash output of the step 53. We will restrict ourselves to one-block collision pairs. So we may write $M = (m_0, m_1, \ldots, m_{15})$, and $M' = (m'_0, m'_1, \ldots, m'_{15})$. We will further restrict ourselves to the collision pairs satisfying

$$m'_3 - m_3 = m'_6 - m_6 = m'_8 - m_8 = m'_{15} - m_{15} = 2^{31}, m'_i = m_i \ (i \neq 3, 6, 8, 15).$$

The reason for choosing m_3, m_6, m_8, m_{15} and the difference 2^{31} for these words is that, in that case there is no difference in steps between 31 and 53; for each step, either the message word used doesn't contain m_3, m_6, m_8, m_{15}, or the differences are cancelled by the XOR operation. Therefore, if we can find a collision for the first 30 steps, then this naturally extends to the collision pair for the first 53 steps. Since m_6 is the message word for step 54, this cannot be extended further.

Note that in the ICISC 2005 paper [7], similar choice of message word differences were made; in [7], the differences were given at the message words m_3 and m_9, i.e., $m'_3 - m_3 = m'_9 - m_9 = 2^{31}$. This choice was also motivated by the fact that, if one may find an inner collision at the step 24, then no further message differences are given until the step 46, producing collision for 45-step HAS-160.

As notational convention, we will distinguish the hash function operation for the message block M' from that for the message block M by appending $'$ symbol to all the corresponding variables. For example, if a_3 is the chaining value a after step 3 during the HAS-160 computation for the message M, then a'_3 is the corresponding variable for M'.

For a chaining variable v ($v = a_i$, b_i, c_i, d_i, or e_i for some i), the difference at v, denoted by Δv, is defined as

$$\Delta v = v' - v \pmod{2^{32}},$$

i.e., Δv is the difference between the value of v for the message M' and the value of v for the message M, measured by the result of subtraction modulo 2^{32}. The message word differences are defined similarly.

Following Wang's methods, we will keep track of the actual bit positions of the chaining variables where the bits differ, along with the modular differences. Precisely, we give the following definition:

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 states that the j-th position of v is 1.

Finally, for l distinct indices $j_1, \ldots, j_l$, the notation $v[\pm j_1, \pm j_2, \ldots, \pm j_l]$ is a shorthand for

$$v[\pm j_1][\pm j_2]\cdots[\pm j_l] = (\cdots((v[\pm j_1])[\pm j_2])\cdots)[\pm j_l].$$

3.2 Differential Path and Sufficient Conditions

Table 4 represent the differential path we used, and Table 5 represent the set of sufficient conditions for the path. All the chaining values are systematically given by the variables a_i, since it is easy to convert between these variables and other variables $b_i, \ldots, e_i$. For example, $b_i = a_{i-1}$, and $c_i = b_{i-1}^{\lll s_{2,i}} = a_{i-2}^{\lll s_{2,i}}$.

The conditions in Table 5 is a set of sufficient conditions for the path given in Table 4, in the sense that, if all the variables $a_{i,j}$ ($i = 1, \ldots, 25$ and $j = 1, \ldots, 32$) satisfy all the equations in the Table 5 during the hash function computation for a message M, then the differences between the chaining variables for the HAS-160 computations of M and M' will follow the path given in Table 4, and (M, M') will constitute a collision pair for the first 53 steps of HAS-160.

The amounts of bit rotation in HAS-160 (i.e., the parameters $s_{1,i}$, $s_{2,i}$) varies wildly depending on the steps, and this makes finding an appropriate differential path such as Table 4 for HAS-160 much harder than in the case of other hash functions. In case of HAVAL, for example, the amount of bit rotation is constant in each steps. Therefore, one may plan ahead how a bit difference in a chaining variable creates or cancels other bit differences in other chaining variables; since the bit rotation is constant, if a pre-existing bit difference can make another bit difference in other bit position, then even if we move those bit differences a few steps up and down, still this property holds. Therefore we can plan creation and annihilation of bit differences ahead without fixing the actual steps where such differences occur. The only obstruction is that the resulting set of conditions doesn't contain any contradiction. Hence, this gives great flexibility to the designer of such differential paths; one can first plan the bit difference creation/annihilation and later try to posit them in actual steps, and check whether this leads to any contradiction. If no contradiction occurs, then we found a valid differential path. If a contradiction occurs, we try another positioning of the bit differences in other steps. [8]

The bit rotation of SHA-1 is similarly constant. And although the bit rotation of MD5 is not constant, MD5 uses only four amounts of bit rotations in each round. Contrary to these tame bit rotations, the bit rotation of HAS-160 varies at each steps so finding a differential path is much difficult in case of HAS-160. This was essentially done manually by trial-and-error, and the trial-and-error process was not quite systematic or scientific; we didn't try some kind of exhaustive searches. We started the differential path from the start down to the middle of the Round 1, and also went from the Step 30 up to the middle of the Round 1, and tried to meet in the middle. Then we tried tweaking the two threads of the differential path by introducing a few more bit differences, or eliminating some bit differences, until they meet and the resulting set of conditions doesn't contain contradiction.

3.3 Collision Search Algorithm

Using the path given at Table 4, and the sufficient conditions given in Table 5, now we can give a collision search algorithm for 53-step HAS-160.

Table 5 give 434 conditions in total. Therefore, heuristically a random message M will satisfy all the conditions with probability about 2^{-434}, which is clearly worse than the birthday probability of 2^{-80}. Therefore we will use the message modification technique introduced in [3] to enhance the probability.

In order to do this, we need to switch the roles of independent words and dependent words. For example, according to Table 3, x_1 is a dependent word defined by $m_8 \oplus m_9 \oplus m_{10} \oplus m_{11}$, and the words $x_{12}, \ldots x_{15}$ are independent

Table 4. Differential path for 53-step HAS-160

Step	x_i	x'_i	$s_{2,i}$	$s_{1,i}$	a'_i
1	$m_8 \oplus m_9$ $\oplus m_{10} \oplus m_{11}$	$x_1[32]$	10	5	$a_1[-32]$
2	m_0		10	11	$a_2[11, -12]$
3	m_1		10	7	$a_3[18, 19, 20, 21, -22]$
4	m_2		10	15	$a_4[1, \cdots, 16, -17]$
5	m_3	$x_5[32]$	10	6	$a_5[7, 8, -9, 18, -19, 32]$
6	$m_{12} \oplus m_{13}$ $\oplus m_{14} \oplus m_{15}$	$x_6[32]$	10	13	$a_6[3, -4, 17, -20, -21, 22]$
7	m_4		10	8	$a_7[-14, \cdots, -17, 18, 22, 29]$
8	m_5		10	14	$a_8[4, -10, 11]$
9	m_6	$x_9[32]$	10	7	$a_9[13, -15, 16, 18, -19, 30, 31, -32]$
10	m_7		10	12	$a_{10}[-10, -11, 12, -17, -24, 25]$
11	$m_0 \oplus m_1$ $\oplus m_2 \oplus m_3$	$x_{11}[32]$	10	9	$a_{11}[-1, 2, -13, -14, -15, 16, 28, -32]$
12	m_8	$x_{12}[32]$	10	11	$a_{12}[-8, 9, -17, -21, 22, 26]$
13	m_9		10	8	$a_{13}[8, 10, 11, -12, 26, 27, -28]$
14	m_{10}		10	15	$a_{14}[-10, 17, 18, -19, 28, 29, 30, -31]$
15	m_{11}		10	6	$a_{15}[11, \cdots, 14, -15, -16, 20, -23, 26, -27]$
16	$m_4 \oplus m_5$ $\oplus m_6 \oplus m_7$	$x_{16}[32]$	10	12	$a_{16}[3, 5, 6, -7, -11, -12, 13,$ $-20, 21, -22, -31, 32]$
17	m_{12}		10	9	$a_{17}[-11, 18, -20, -26, 27]$
18	m_{13}		10	14	$a_{18}[1, 5, 18, 20, 22, 27]$
19	m_{14}		10	5	$a_{19}[4, 13, 30]$
20	m_{15}	$x_{20}[32]$	10	13	$a_{20}[-4, 11]$
21	$m_{11} \oplus m_{14}$ $\oplus m_1 \oplus m_4$		17	5	$a_{21}[11]$
22	m_3	$x_{22}[32]$	17	11	$a_{22}[-30]$
23	m_6	$x_{23}[32]$	17	7	$\cdot$
24	m_9		17	15	$a_{24}[15]$
25	m_{12}		17	6	$a_{25}[-15]$
26	$m_7 \oplus m_{10}$ $\oplus m_{13} \oplus m_0$		17	13	$\cdot$
27	m_{15}	$x_{27}[32]$	17	8	$\cdot$

Table 4. (*continued*)

Step	x_i	x'_i	$s_{2,i}$	$s_{1,i}$	a'_i
28	m_2		17	14	·
29	m_5		17	7	·
30	m_8	$x_{30}[32]$	17	12	·
31	$m_3 \oplus m_6$ $\oplus m_9 \oplus m_{12}$		17	9	·
32	m_{11}		17	11	·
33	m_{14}		17	8	·
34	m_1		17	15	·
35	m_4		17	6	·
36	$m_{15} \oplus m_2$ $\oplus m_5 \oplus m_8$		17	12	·
37	m_7		17	9	·
38	m_{10}		17	14	·
39	m_{13}		17	5	·
40	m_0		17	13	·
41	$m_4 \oplus m_{13}$ $\oplus m_6 \oplus m_{15}$		25	5	·
42	m_{12}		25	11	·
43	m_5		25	7	·
44	m_{14}		25	15	·
45	m_7		25	6	·
46	$m_8 \oplus m_1$ $\oplus m_{10} \oplus m_3$		25	13	·
47	m_0		25	8	·
48	m_9		25	14	·
49	m_2		25	7	·
50	m_{11}		25	12	·
51	$m_{12} \oplus m_5$ $\oplus m_{14} \oplus m_7$		25	9	·
52	m_4		25	11	·
53	m_{13}		25	8	·
54	m_6	$x_{54}[32]$	25	15	$a_{54}[\pm 32]$

words $m_8, \ldots, m_{11}$. But, changing the perspective, we may regard x_1, x_{12}, x_{13}, and x_{14} as independent words and consider x_{15} as the dependent word which is equal to $x_1 \oplus x_{12} \oplus x_{13} \oplus x_{14}$. In this way we may apply the message modification technique to words x_1, x_{12}, x_{13}, and x_{14}. In this case x_{15} is determined by the choices of the independent words, therefore we cannot apply simple message modification technique to x_{15}.

Similarly, we will redefine the dependent word as the last one of the dependent set of five words. In other words, after the redefinition, the dependent words are x_{11}, x_{15}, x_{16}, x_{20}, and the rest of the message words are independent.

Therefore, we may apply the message modification technique to Steps 1 to 10 with probability 1. At Step 11, the dependent word x_{11} is used, and according to Table 5, there are 27 conditions at the step. So we may satisfy all the conditions

up to Step 11 with probability 2^{-27}. Again we apply the message modification technique to Steps 12 to 14, and at Steps 15 and 16 the dependent words x_{15} and x_{16} are used, and there are 54 conditions in total at Step 15 and Step 16. Therefore, the probability that all the conditions are met at Steps 15 and 16 are 2^{-54}.

If any of the conditions at Step 15 or Step 16 are not satisfied, then we have to go back to Step 12, redefine message words x_{12}, x_{13}, and x_{14}, and retry Steps 15 and 16 to see if all the conditions are met this time. But, note that at Step 12, there are 23 conditions so the number of message words x_{12} which make the corresponding a_{12} to satisfy all the conditions is heuristically about 2^9. Similarly for Step 13, there are about 2^7 correct x_{13}s, and about 2^8 correct x_{14}s. In total, there are about 2^{24} correct (x_{12}, x_{13}, x_{14}) for Steps 12, 13, and 14.

Therefore, even if we backtrack to Step 12 and try all the correct words for Steps 12, 13, and 14, it is possible that some conditions are not met at Step 15 or 16. In this case we backtrack to Step 1 and try again. The probability that all the conditions are met after all these is 2^{-30}.

Table 5. Sufficient conditions for the differential path

Var.	No. of conds.	Sufficient conditions
a_1	7	$a_{1,1} = 1$, $a_{1,2} = 0$, $a_{1,8} = 1$, $a_{1,10} = 0$, $a_{1,11} = 1$, $a_{1,12} = 1$, $a_{1,32} = 1$
a_2	12	$a_{2,3} = 1$, $a_{2,7} = 1$, $a_{2,8} = 0$, $a_{2,9} = a_{1,9}$, $a_{2,10} = 0$, $a_{2,11} = 0$, $a_{2,12} = 1$, $a_{2,25} = 1$, $a_{2,29} = 0$, $a_{2,30} = 0$, $a_{2,31} = 1$, $a_{2,32} = 0$
a_3	25	$a_{3,1} = a_{2,1}$, $a_{3,2} = a_{2,2}$, $a_{3,3} = 0$, $a_{3,4} = a_{2,4}$, ..., $a_{3,6} = a_{2,6}$, $a_{3,7} = 0$, $a_{3,8} = 1, a_{3,9} = 0$, $a_{3,10} = 1$, $a_{3,18} = 0$, ..., $a_{3,21} = 0$, $a_{3,22} = 1$, $a_{3,23} = a_{2,23}$, $a_{3,24} = a_{2,24}$, $a_{3,25} = 0$, $a_{3,26} = a_{2,26}$, ..., $a_{3,31} = a_{2,31}$, $a_{3,32} = 1$
a_4	26	$a_{4,1} = 0$, ..., $a_{4,16} = 0$, $a_{4,17} = 1$, $a_{4,21} = 0$, $a_{4,22} = 1$, $a_{4,25} = 1$, $a_{4,26} = 1$, $a_{4,28} = 0$, ..., $a_{4,30} = 0$, $a_{4,31} = 1$, $a_{4,32} = 1$
a_5	28	$a_{5,4} = 1$, $a_{5,5} = 1$, $a_{5,7} = 0$, $a_{5,8} = 0$, $a_{5,9} = 1$, ..., $a_{5,11} = 1$, $a_{5,12} = 0$, $a_{5,13} = 0$, $a_{5,14} = 1$, $a_{5,15} = 0$, ..., $a_{5,18} = 0$, $a_{5,19} = 1$, $a_{5,20} = 0$, $a_{5,21} = 1$, $a_{5,22} = 1$, $a_{5,23} = 0$, $a_{5,24} = 0$, $a_{5,25} = 1$, ..., $a_{5,27} = 1$, $a_{5,28} = 0$, $a_{5,29} = 1$, ..., $a_{5,31} = 1$, $a_{5,32} = 0$
a_6	27	$a_{6,1} = 0$, $a_{6,3} = 0$, $a_{6,4} = 1$, $a_{6,5} = 0$, $a_{6,6} = a_{5,6}$, $a_{6,7} = 0$, $a_{6,8} = 0$, $a_{6,10} = 1$, $a_{6,11} = 0$, ..., $a_{6,17} = 0$, $a_{6,19} = 1$, ..., $a_{6,21} = 1$, $a_{6,22} = 0$, $a_{6,23} = 1$, ..., $a_{6,27} = 1$, $a_{6,28} = 0$, $a_{6,29} = 0$, $a_{6,32} = 0$
a_7	19	$a_{7,1} = 1$, $a_{7,9} = 0$, $a_{7,10} = 1$, $a_{7,13} = 1$, ..., $a_{7,17} = 1$, $a_{7,18} = 0$, $a_{7,19} = 0$, $a_{7,20} = 1$, $a_{7,22} = 0$, $a_{7,26} = a_{6,26}$, $a_{7,27} = 0$, $a_{7,28} = 1$, $a_{7,29} = 0$, $a_{7,30} = 1$, $a_{7,31} = 0$, $a_{7,32} = 1$
a_8	25	$a_{8,1} = 1$, $a_{8,2} = 1$, $a_{8,3} = a_{7,3}$, $a_{8,4} = 0$, $a_{8,5} = a_{7,5}$, $a_{8,6} = a_{7,6}$, $a_{8,7} = 0$, $a_{8,8} = a_{7,8}$, $a_{8,9} = 1$, $a_{8,10} = 1$, $a_{8,11} = 0$, $a_{8,13} = 1$, ..., $a_{8,15} = 1$, $a_{8,20} = 0$, $a_{8,21} = a_{7,21}$, $a_{8,22} = 1$, $a_{8,24} = 1$, $a_{8,25} = 0$, ..., $a_{8,28} = 0$, $a_{8,30} = 0$, $a_{8,31} = 0$, $a_{8,32} = 1$
a_9	22	$a_{9,1} = 0$, $a_{9,2} = 0$, $a_{9,4} = 0$, $a_{9,6} = 1$, $a_{9,7} = 1$, $a_{9,13} = 0$, $a_{9,14} = 1$, $a_{9,15} = 1$, $a_{9,16} = 0$, $a_{9,18} = 0$, $a_{9,19} = 1$, $a_{9,20} = 0$, $a_{9,21} = 0$, $a_{9,23} = 0$, $a_{9,24} = 1$, $a_{9,25} = 1$, $a_{9,26} = 0$, $a_{9,27} = 1$, $a_{9,28} = 0$, $a_{9,30} = 0$, $a_{9,31} = 0$, $a_{9,32} = 1$
a_{10}	23	$a_{10,3} = a_{9,3}$, $a_{10,4} = 1$, $a_{10,5} = a_{9,5}$, $a_{10,6} = 0$, ..., $a_{10,9} = 0$, $a_{10,10} = 1$, $a_{10,11} = 1$, $a_{10,12} = 0$, $a_{10,14} = 1$, $a_{10,16} = 1$, $a_{10,17} = 1$, $a_{10,18} = 0$, $a_{10,20} = 1$, $a_{10,21} = 0$, $a_{10,22} = a_{9,22}$, $a_{10,23} = 0$, $a_{10,24} = 1$, $a_{10,25} = 0$, $a_{10,26} = 1$, $a_{10,28} = 0$, ..., $a_{10,30} = 0$

Table 5. (*continued*)

Var.	No. of conds.	Sufficient conditions
a_{11}	27	$a_{11,1}=1$, $a_{11,2}=0$, $a_{11,3}=0$, $a_{11,7}=1$, $a_{11,8}=0$, $a_{11,9}=1$, $a_{11,10}=0$, …, $a_{11,12}=0$, $a_{11,13}=1$, …, $a_{11,15}=1$, $a_{11,16}=0$, $a_{11,17}=0$, $a_{11,18}=1$, $a_{11,20}=1$, $a_{11,21}=0$, $a_{11,22}=0$, $a_{11,23}=1$, $a_{11,25}=0$, $a_{11,26}=1$, $a_{11,27}=0$, …, $a_{11,29}=0$, $a_{11,30}=1$, $a_{11,31}=a_{10,31}$, $a_{11,32}=1$
a_{12}	23	$a_{12,1}=0$, $a_{12,2}=1$, $a_{12,3}=1$, $a_{12,6}=0$, $a_{12,8}=1$, $a_{12,9}=0$, …, $a_{12,12}=0$, $a_{12,16}=1$, …, $a_{12,21}=1$, $a_{12,22}=0$, $a_{12,23}=0$, $a_{12,24}=1$, $a_{12,25}=0$, $a_{12,26}=0$, $a_{12,27}=1$, $a_{12,30}=a_{11,30}$, $a_{12,32}=1$
a_{13}	25	$a_{13,1}=1$, $a_{13,2}=1$, $a_{13,4}=0$, $a_{13,5}=1$, $a_{13,6}=1$, $a_{13,7}=a_{12,7}$, $a_{13,8}=0$, $a_{13,9}=1$, $a_{13,10}=0$, $a_{13,11}=0$, $a_{13,12}=1$, $a_{13,16}=1$, $a_{13,17}=0$, …, $a_{13,21}=0$, $a_{13,23}=1$, …, $a_{13,25}=1$, $a_{13,26}=0$, $a_{13,27}=0$, $a_{13,28}=1$, $a_{13,31}=0$, $a_{13,32}=0$
a_{14}	24	$a_{14,1}=0$, $a_{14,2}=0$, $a_{14,3}=a_{13,3}$, $a_{14,4}=1$, …, $a_{14,6}=1$, $a_{14,10}=1$, $a_{14,11}=0$, $a_{14,12}=1$, $a_{14,13}=a_{13,13}$, $a_{14,16}=0$, …, $a_{14,18}=0$, $a_{14,19}=1$, $a_{14,20}=1$, $a_{14,21}=0$, $a_{14,22}=1$, $a_{14,25}=0$, $a_{14,27}=1$, $a_{14,28}=0$, …, $a_{14,30}=0$, $a_{14,31}=1$, $a_{14,32}=0$
a_{15}	27	$a_{15,1}=0$, $a_{15,2}=a_{14,2}$, $a_{15,3}=a_{14,3}$, $a_{15,4}=0$, $a_{15,5}=0$, $a_{15,6}=1$, $a_{15,7}=1$, $a_{15,8}=0$, $a_{15,9}=1$, $a_{15,10}=0$, …, $a_{15,14}=0$, $a_{15,15}=1$, $a_{15,16}=1$, $a_{15,17}=0$, $a_{15,18}=1$, $a_{15,20}=0$, …, $a_{15,22}=0$, $a_{15,23}=1$, $a_{15,25}=1$, $a_{15,26}=0$, $a_{15,27}=1$, …, $a_{15,29}=1$
a_{16}	27	$a_{16,1}=1$, $a_{16,3}=0$, $a_{16,4}=1$, $a_{16,5}=0$, $a_{16,6}=0$, $a_{16,7}=1$, $a_{16,8}=0$, $a_{16,9}=0$, $a_{16,10}=1$, …, $a_{16,12}=1$, $a_{16,13}=0$, $a_{16,16}=1$, $a_{16,17}=a_{15,17}$, $a_{16,20}=1$, $a_{16,21}=0$, $a_{16,22}=1$, $a_{16,23}=0$, $a_{16,24}=0$, $a_{16,25}=1$, …, $a_{16,27}=1$, $a_{16,28}=0$, $a_{16,29}=1$, $a_{16,30}=0$, $a_{16,31}=1$, $a_{16,32}=0$
a_{17}	24	$a_{17,1}=1$, $a_{17,4}=0$, $a_{17,5}=1$, $a_{17,8}=a_{16,8}$, $a_{17,9}=0$, $a_{17,10}=1$, $a_{17,11}=1$, $a_{17,12}=0$, $a_{17,13}=1$, $a_{17,15}=0$, …, $a_{17,18}=0$, $a_{17,20}=1$, $a_{17,21}=1$, $a_{17,22}=0$, $a_{17,23}=1$, $a_{17,24}=0$, $a_{17,25}=1$, $a_{17,26}=1$, $a_{17,27}=0$, $a_{17,30}=1$, …, $a_{17,32}=1$
a_{18}	18	$a_{18,1}=0$, $a_{18,3}=a_{17,3}\oplus 1$, $a_{18,4}=0$, $a_{18,5}=0$, $a_{18,9}=1$, $a_{18,10}=1$, $a_{18,13}=1$, $a_{18,15}=1$, $a_{18,16}=1$, $a_{18,17}=0$, $a_{18,18}=0$, $a_{18,20}=0$, $a_{18,22}=0$, $a_{18,23}=0$, $a_{18,27}=0$, $a_{18,28}=0$, $a_{18,31}=1$, $a_{18,32}=0$
a_{19}	7	$a_{19,4}=0$, $a_{19,11}=1$, $a_{19,13}=0$, $a_{19,15}=0$, $a_{19,19}=a_{18,26}\oplus 1$, $a_{19,21}=a_{18,11}$, $a_{19,30}=0$
a_{20}	6	$a_{20,4}=1$, $a_{20,5}=a_{19,20}\oplus 1$, $a_{20,11}=0$, $a_{20,21}=a_{18,11}\oplus 1$, $a_{20,26}=a_{19,26}\oplus 1$, $a_{20,28}=0$
a_{21}	5	$a_{21,11}=0$, $a_{21,13}=a_{20,13}$, $a_{21,15}=a_{20,30}\oplus 1$, $a_{21,28}=0$, $a_{21,30}=a_{20,13}\oplus 1$
a_{22}	2	$a_{22,21}=a_{21,4}$, $a_{22,30}=1$
a_{23}	2	$a_{23,15}=a_{21,30}$, $a_{23,28}=a_{22,11}\oplus 1$
a_{24}	2	$a_{24,15}=0$, $a_{24,30}=a_{23,30}\oplus 1$
a_{25}	1	$a_{25,15}=1$

Therefore, the amount of computations for satisfying all the conditions up to Step 16 is

$$(2^{27}\cdot(\text{Steps 1 to 11})+2^{24}\cdot(\text{Steps 12 to 16}))\cdot 2^{30}\approx 2^{60.54}\text{ steps.}$$

From Step 20 to Step 25, all the message words are dependent words; after the redefinition, x_{20} is a dependent word, and x_{21}, …, x_{25} are words in the Round 2, which are already fixed after the message modifications for the Round 1. And the number of conditions for these steps is 18 in total. Therefore after satisfying all the conditions up to Step 16, we may try 2^{18} choices of (x_{17}, x_{18}, x_{19}) to

satisfy all the remaining conditions. But the complexity involved in these steps are negligible compared to the complexity for the steps 1 to 16.

Therefore, the amount of total computation is

$$(2^{27} \cdot (\text{Steps 1 to 11}) + 2^{24} \cdot (\text{Steps 12 to 16})) \cdot 2^{30} + 2^{18} \cdot (\text{Steps 17 to 25}) \approx 2^{60.54} \text{ steps}$$

which corresponds to about $2^{54.81} \approx 2^{55}$ hash function operations of 53-step HAS-160.

4 Conclusion

In this paper, we presented a collision search attack for 53-step HAS-160 with time complexity 2^{55}. This attack is to find an inner collision with long trailing zero message word differences, and this cannot be extended to the whole of HAS-160. We believe that new techniques should be used in order to find an attack for the complete HAS-160.

References

1. Telecommunications Technology Association, *Hash Function Standard Part 2: Hash Function Algorithm Standard (HAS-160)*, TTAS.KO-12.0011/R1, Dec. 2000.
2. FIPS 180-2, *Secure Hash Standard*, `http://csrc.nist.gov/publications/`, 2002.
3. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, Xiuyuan Yu, *Cryptanalysis of the Hash Functions MD4 and RIPEMD*, Advances in Cryptology - EUROCRYPT 2005, R. Cramer (Ed.), LNCS 3494, Springer-Verlag, pp. 1–18, 2005.
4. Xiaoyun Wang, Hongbo Yu, *How to Break MD5 and Other Hash Functions*, Advances in Cryptology - EUROCRYPT 2005, R. Cramer (Ed.), LNCS 3494, pp. 19–35, Springer-Verlag, 2005.
5. Xiaoyun Wang, Hongbo Yu, Yiqun Lisa Yin, *Efficient Collision Search Attacks on SHA-0*, Advances in Cryptology - CRYPTO '05, LNCS 3621, pp. 1–16, Springer-Verlag, 2005.
6. Xiaoyun Wang, Yiqun Lisa Yin, Hongbo Yu, *Finding Collisions in the Full SHA-1*, Advances in Cryptology - CRYPTO '05, LNCS 3621, pp. 17–36, Springer-Verlag, 2005.
7. Aaram Yun, Soo Hak Sung, Sangwoo Park, Donghoon Chang, Seokhie Hong, Hong-Su Cho, *Finding Collision on 45-Step HAS-160*, Information Security and Cryptology - ICISC 2005, LNCS 3935, pp. 146–155, Springer-Verlag, 2006.
8. Hongbo Yu, Xiaoyun Wang, Aaram Yun, Sangwoo Park, *Cryptanalysis of the Full HAVAL with 4 and 5 Passes*, to be published in the Proceedings of FSE 2006.

Klein Bottle Routing: An Alternative to Onion Routing and Mix Network

Kun Peng[1], Juan Manuel Nieto[1], Yvo Desmedt[2], and Ed Dawson[1]

[1] Information Security Institute, Queensland University of Technology
[2] University College London

Abstract. Traditionally, there are two methods to implement anonymous channels: free-route networks like onion routing and cascade networks like mix network. Each of them has its merits and is suitable for some certain applications of anonymous communication. Both of them have their own drawbacks, so neither of them can satisfy some applications. A third solution to anonymous channels, Klein bottle routing, is proposed in this paper. It fills the gap between onion routing and mix network and can be widely employed in anonymous communication.

Keywords: Anonymous channel, onion routing, mix network, Klein bottle routing.

1 Introduction

Anonymous (communication) channel is a very useful tool in e-business, e-government and other cryptographic applications, which often require anonymity and privacy. In an anonymous channel, the messages are untraceable, so can be transmitted anonymously. Traditionally, there are two methods to implement anonymous channels: onion routing [5,6,1] and mix network [4,8,13,12,11].

Onion routing is an anonymous routing mechanism. It employs free-route mechanism and decryption chain. A node in an onion routing communication network can send a message to any node in the network with a connection to it. The sender can flexibly choose any route from all the connections between him and the receiver. A decryption chain is employed to mask each message packet when it is transmitted. In a packet, a message is encrypted layer by layer using the public keys of all the routers on its route and the receiver. Each layer of encryption is just like a layer of onion skin, which encrypts the encrypted packet for a router and the identity of the next router. Given a message packet, each router unwraps a layer of encryption by decrypting the message packet using its private key, finds out the identity of the next router and forwards the unwrapped message packet to the next router. When a packet is routed by each router together with a large number of other packets, the onion structure and decryption chain prevent it from being traced. Onion routing is flexible and can be employed in various anonymous communication applications like anonymous email and anonymous browsing when verifiability of correct routing is not required. However, onion routing has two drawbacks. Firstly, it is not

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 296–309, 2006.
© Springer-Verlag Berlin Heidelberg 2006

verifiable and there is no guarantee that no message packet is lost or tampered with as all the existing public verifiability mechanism in anonymous channels are re-encryption oriented. Secondly, untraceability depends on communication traffic in the network and traffic analysis can always get some hint about a message packet's route unless each node sends a packet to every other node during transmission of this packet.

Strictly speaking, mix network is not a routing mechanism as there is only one unique route and one unique receiver in a mix network. This routing mechanism is also called cascade network. Multiple fixed nodes are stationed in a fixed order and form a fixed route called a mix network. Each sender encrypts and submits his message to the first node in the mix network. After all the senders have submitted their messages, each node in the mix network take turns to re-encrypt and shuffle them. The last node outputs the repeatedly re-encrypted and shuffled messages to the receiver. The receiver can read the messages after they are decrypted but cannot link them to their senders. Mix network is usually employed in special anonymous communication applications transmitting a batch of messages to a unique receiver like e-voting [10] and anonymous e-auction [14]. As every sender sends a message through the unique route to the unique receiver in a mix network, there is no concern for traffic analysis. As re-encryption instead of decryption-chain is employed to mask the packets, a mix network can be publicly verified to have transmitted every message without tampering with it. However, mix network is not flexible as a sender cannot freely choose receiver or route. Therefore, mix network is not a general solution to anonymous channels and can only be employed in special applications like e-voting and e-auction.

When general and flexible anonymous channel with public verifiability is required (e.g. for insured anonymous email), neither onion routing nor mix network can be applied. So a third type of anonymous channel is needed in this circumstance, which combines merits of the two types of traditional anonymous channel and overcomes their drawbacks. In this paper flexible routing in onion routing and re-encryption masking in mix network are combined to design a flexible and publicly verifiable general implementation of anonymous routing. It is called Klein bottle routing. The idea of applying re-encryption masking to anonymous routing was independently proposed by Gomulkiewicz *et al* [9]. Their proposal is a modified onion routing scheme to prevent repetitive attack. Danezis pointed out in [2] that the routing scheme in [9] is vulnerable to an interception-insertion attack. Our proposal in this paper is a new type of anonymous channel with its own security model and properties. Our new solution fills the gap between onion routing and mix network. Moreover, our scheme can be easily modified to prevent the attack in [2].

Contribution of this paper is as follows. Firstly, security of anonymous routing is modeled. A basic requirement for anonymous routing, isolated anonymity, is defined. Secondly, an encryption algorithm, ElGamal encryption with universal re-encryption and n-out-of-n distributed decryption, is designed as a variant of the ElGamal encryption algorithm with universal re-encryption [7]. Then Klein bottle routing is designed. In Klein bottle routing, to enable re-encryption of a

packet without knowledge of the receiver's public key, the ElGamal encryption algorithm with universal re-encryption and n-out-of-n distributed decryption are employed. As Klein bottle routing is a flexible routing scheme, there is no practical method to completely prevent traffic analysis. However, measures can be taken to reduce the harm of traffic analysis. For example, to make traffic analysis more difficult, in Klein bottle routing the size of a message packet is not changed when it is transmitted. Finally, security of Klein bottle routing is analysed. It is firstly formally proved to be isolatedly anonymous, namely anonymous when traffic analysis attack is not taken into consideration. It is then illustrated to be flexible, publicly verifiable and to resist traffic analysis based on packet size statistics.

2 Security of Anonymous Routing

In an anonymous routing system, each message is transmitted in the form of a packet, which contains two parts: encrypted data and encrypted routing information. The encrypted data is the encryption of the transmitted message, while the encrypted routing information is the encryption of the routers' identities. When a message s is sent by sender S through routers $N_1, N_2, \ldots, N_m$ to receiver R, it is transmitted in a packet chain $p_0, p_1, \ldots, p_m$ where p_i is the packet sent by N_i to N_{i+1}. (S is regarded as N_0 and R is regarded as N_{m+1}) A basic requirement for anonymity in an anonymous routing system is raised in the new definition as follows..

Definition 1. *An adversary randomly chooses two input packets p_1 and p_2 for a router to route, then the router selects i from $\{1, 2\}$ and routes p_i to an output packet p'. The communication channel is isolatedly anonymous if given p', without the router's help the adversary can output i with a probability no more than $0.5 + \epsilon$ and ϵ is negligible.*

Isolated anonymity is only a basic requirement for anonymity without considering traffic analysis. In practice, anonymity requirement in a communication network is more sophisticated. When an attacker can monitor all the traffic in a communication network, he can perform a traffic analysis, which takes into account the time each packet is transmitted between two routers. The attacker can try to analyse which of all the packets transmitted in the network are more likely to contain the same message. Obviously, isolated anonymity is necessary but not sufficient to guarantee anonymity, since it cannot guarantee that no message can be traced when traffic analysis is used. For example, when only one message is transmitted in the network in a long enough time period, traffic analysis definitely reveals its route. In another example, in most onion routing schemes an observer monitoring the traffic can tell how far each packet is from its destination according to its length. This advantage enables the observer to perform a simple but effective traffic analysis attack. For example, if a router receives a message packet with n routers to pass and after a long enough period only sends out one message packet with $n - 1$ routers to pass, the observer

can link these two packets and deduce that they contain the same message. As the traffic in a network is variable and complicated, traffic analysis is complex and difficult to formally define. Various analysis methods on the traffic might be effective depending on distribution of the routers, their connections and the traffic. So anonymity in a general and practical anonymous channel is difficult to precisely define when traffic analysis is involved. In this paper it is only required that the changing packet size based traffic analysis attack against anonymity is prevented, so that isolated anonymity can be achieved. More sophisticated traffic analysis attack is not considered in the relatively simple secuirty model in this paper.

Validity and public verifiability of routing are also important. A dishonest router may discard or tamper with a message packet. To publicly guarantee validity of routing of every packet, each router must be publicly verified to strictly follow the routing protocol when routing all the packets passing him. More precisely, each router must prove that each of its input packets is routed to one of its output packets containing the same message, which must be publicly verifiable.

3 Encryption Algorithm

With Klein bottle routing, re-encryption instead of decryption chain is employed to mask the packets in a communication channel. There are two advantages of re-encryption based masking. Firstly, validity of re-encryption of a packet can be publicly proved and verified without revealing its route. Secondly, re-encryption inherently keeps the size of the packets constant. However, there are some challenges to application of re-encryption to anonymous routing. Firstly, anonymity of the communication channel requires that the identities of the receiver and the routers must remain secret. So re-encryption of a packet must be performed without knowing who will decrypt it. That means re-encryption must take place without knowledge of the public key. Secondly, any part of a packet cannot be encrypted or re-encrypted with a single public key, otherwise the owner of the corresponding private key can trace the packet.

To implement re-encryption based masking in Klein bottle routing and respond to the two challenges, distributed decryption [3] is applied to the ElGamal encryption with universal re-encryption in [7]. Combination of these two techniques guarantees that re-encryption based masking can be implemented without compromising anonymity as re-encryption is performed without public key and recovering the private key to trace a packet needs collusion of all the private key share holders.

3.1 An Employed Cryptographic Primitive: Universal Re-encryption

Universal re-encryption [7] is a cryptographic primitive to be modified and adopted in this paper. The original ElGamal encryption algorithm is modified in [7], such that re-encryption can be performed without knowledge of public key. It is as follows.

- Key generation
  Large primes p and q are chosen, such that q is a factor of $p-1$. G is the subgroup of Z_p with order q. g is a generator of G. The private key x is randomly chosen from Z_q, while the public key is y where $y = g^x$. Unless specified, all multiplication computations are with modulus p.
- Encryption
  A message m is encrypted into $c = E(m) = ((a_0, b_0),\ (a_1, b_1)) = ((my^{r_0}, g^{r_0}),\ (y^{r_1}, g^{r_1}))$ where r_0 and r_1 are randomly chosen from Z_q.
- Decryption
  Given a ciphertext $c = ((a_0, b_0),\ (a_1, b_1))$, the decryption party calculates $m_0 = a_0/b_0^x$ and $m_1 = a_1/b_1^x$. If $m_1 = 1$, m_0 is output as the message. If $m_1 \neq 1$, decryption fails and c is declared as an invalid ciphertext.
- Re-encryption
  Given a ciphertext $c = ((a_0, b_0),\ (a_1, b_1))$, a party without knowledge of y can calculate $c' = RE(c) = ((a_0', b_0'),\ (a_1', b_1')) = ((a_0 a_1^{r_0'}, b_0 b_1^{r_0'}),\ (a_1^{r_1'}, b_1^{r_1'}))$ where r_0' and r_1' are randomly chosen from Z_q. c' is a re-encryption of c where $RE()$ denotes the re-encryption function.

3.2 A New Encryption Algorithm: ElGamal Encryption with Universal Re-encryption and n-out-of-n Distributed Decryption

Note that the universal re-encryption mechanism in [7] does not change the keys of ElGamal encryption algorithm (private key x and public key $y = g^x$), so the distributed decryption mechanism of ElGamal encryption in [3] can be applied. The only difference is that n-out-of-n decryption is employed in this new encryption algorithm while k-out-of-n threshold decryption ($k < m$) is employed in [3]. ElGamal encryption with universal re-encryption and n-out-of-n distributed decryption is designed as follows.

- Key generation
  Key sharing parties $A_1, A_2, \ldots, A_n$ are chosen. Each of them, A_i, selects an integer x_i from Z_q as his private key and publishes his public key $y_i = g^{x_i}$. The encryption key is y, which is the product of the public keys of all the parties: $\prod_{i=1}^{n} y_i$. The corresponding decryption key is $x = \sum_{i=1}^{n} x_i$, which is shared among the sharing parties with an n-out-of-n threshold.
- Distributed decryption
  A ciphertext $c = ((a_0, b_0),\ (a_1, b_1))$ encrypted with y is decrypted in the following way.
  1. Each A_i calculates $d_{0,i} = b_0^{x_i}$ and $d_{1,i} = b_1^{x_i}$ for $i = 1, 2, \ldots, n$.
  2. $m_0 = a_0 / \prod_{i=1}^{n} d_{0,i}$ and $m_1 = a_1 / \prod_{i=1}^{n} d_{1,i}$ are calculated. If $m_1 = 1$, m_0 is output as the message. If $m_1 \neq 1$, decryption fails and c is declared as an invalid ciphertext.

Later in this paper, the following notations are used to describe ElGamal encryption with universal re-encryption and n-out-of-n distributed decryption when it is employed in our masking and routing mechanism.

- Routers $N_1, N_2, \ldots, N_m$ and receiver R are the decryption key shares.
- Function $E_{N_1,N_2,\ldots,N_m,R}(s)$ stands for encryption of message s with a public key, which is the product of the public keys of $N_1, N_2, \ldots, N_m$ and R [1].
- Function $RE(c)$ stands for universal re-encryption of ciphertext c.
- Function $D_i(c)$ denotes partial decryption of ciphertext $c = ((a_0, b_0), (a_1, b_1))$ using N_i's private key: $D_i(c) = ((a_0/b_0^{x_i}, b_0), (a_1/b_1^{x_i}, b_1)$, which is N_i's operation in distributed decryption of c.
- Function $D_R(c)$ denotes partial decryption of ciphertext $c = ((a_0, b_0), (a_1, b_1))$ using R's private key x_R: $D_R(c) = ((a_0/b_0^{x_R}, b_0), (a_1/b_1^{x_R}, b_1))$.

Obviously, $D_R(D_m(D_{m-1}(\ldots(E_{N_1,N_2,\ldots,N_m,R}(s)))\ldots) = s$.

Anonymity of Klein bottle routing depends on a property of this new encryption system: semantic security under re-encryption. The original universal re-encryption function in [7] has been proved to be semantically secure under re-encryption. However, the definition and proof of semantic security under re-encryption in [7] are not suitable in this paper because of the following two reasons.

- Klein bottle routing applies distributed decryption to the original encryption scheme in [7]. So knowledge of some of the partial private keys must be considered when analysing semantic security.
- In the definition of semantic security in [7], it is assumed that the adversary generates the input ciphertexts and knows the encryption details. We believe that it is unnecessary to make this complex assumption in Klein bottle routing as participants like routers may know nothing about generation of the input ciphertexts. So we do not adopt this assumption and our definition is more general.

Semantic security under re-encryption of the ElGamal encryption system with universal re-encryption and n-out-of-n distributed decryption is defined and proved in this section in regard of routing application. To introduce semantic security under re-encryption, a game is introduced in Fig 1.

Definition 2. *The ElGamal encryption with universal re-encryption and n-out-of-n distributed decryption with re-encryption function $RE()$ is semantically secure under re-encryption if the adversary can win the game in Fig 1 in polynomial time with a probability no more than $0.5 + \epsilon$ and ϵ is negligible.*

Theorem 1. *The ElGamal encryption with universal re-encryption and n-out-of-n distributed decryption is semantically secure under re-encryption.*

Proof: Suppose $c_i = ((a_{i,0}, b_{i,0}), (a_{i,1}, b_{i,1}))$ and $c' = ((a'_0, b'_0), (a'_1, b'_1)) = ((a_{i,0}a_{i,1}^{r'_0}, b_{i,0}b_{i,1}^{r'_0}), (a_{i,1}^{r'_1}, b_{i,1}^{r'_1}))$ where r'_0 and r'_1 are randomly chosen from Z_q. If the adversary can win the game in Fig 1 in polynomial time with a probability

[1] Note that the ElGamal encryption system with universal re-encryption and n-out-of-n distributed decryption is employed. So $N_1, N_2, \ldots, N_m$ and R can cooperate to decrypt any ciphertext encrypted with the product of their public keys.

1. An adversary is given two public keys y_1 and y_2; all the partial public keys $y_{1,1}, y_{1,2}, \ldots, y_{1,n_1}$ and $y_{2,1}, y_{2,2}, \ldots, y_{2,n_2}$ such that $y_1 = \prod_{i=1}^{n_1} y_{1,i}$ and $y_2 = \prod_{i=1}^{n_2} y_{2,i}$. The adversary may know at most $n_1 - 1$ partial private keys among $\{x_{1,1}, x_{1,2}, \ldots, x_{1,n_1}\}$ and at most $n_2 - 1$ partial private keys among $\{x_{2,1}, x_{2,2}, \ldots, x_{2,n_2}\}$ where $y_{j,i} = g^{x_{j,i}}$. Thus the adversary has not any knowledge of the two private key x_1 and x_2 where $x_1 = \sum_{i=1}^{n_1} x_{1,i}$ and $x_2 = \sum_{i=1}^{n_2} x_{2,i}$.
2. The adversary chooses two ciphertexts c_1 and c_2 encrypted with y_1 and y_2 respectively.
3. A challenger randomly selects i from $\{1, 2\}$.
4. The challenger calculates $c' = RE(c_i)$ and sends it to the adversary while keeping i secret.
5. The adversary is asked to output i.

Fig. 1. The game for semantic security of re-encryption

$0.5 + \epsilon$ and ϵ is not negligible, then the adversary can tell that $\log_{a_{i,1}} a'_0/a_{i,0} = \log_{b_{i,1}} b'_0/b_{i,0}$ and $\log_{a_{i,1}} a'_1 = \log_{b_{i,1}} b'_1$ in polynomial time with a probability $0.5 + \epsilon$ and ϵ is not negligible. Note that the adversary does not know private keys x_1 and x_2 and his only method to tell whether $\log_{a_{i,1}} a'_0/a_{i,0} = \log_{b_{i,1}} b'_0/b_{i,0}$ and $\log_{a_{i,1}} a'_1 = \log_{b_{i,1}} b'_1$ is to solve the decisional Diffie-Hellman problem. This is contradictory to the widely accepted assumption that DDH problem is difficult to solve in polynomial time. Therefore the adversary can win the game in Fig 1 in polynomial time with a probability no more than $0.5+\epsilon$ where ϵ is negligible. □

4 Klein Bottle Routing

Klein bottle routing is implemented in this section, which employs repeated re-encryption and distributed decryption instead of a decryption chain. A Klein bottle is a single sided bottle with no boundary. Its inside is its outside, such that it contains itself. It is closed and non-orientable, so a symbol on its surface can be slid around on it and reappear backwards at the same place. We use this name to emphasize the difference between the new routing scheme and onion routing. In onion routing, the layers of encryption are removed one by one from a packet just like the layers of skin are removed one by one from an onion. The new routing mechanism always maintain the same size and nothing is removed from it. From its appearance it is impossible to tell how far it has traveled and how far it is still required to travel. It seems that the packet is traveling on a tour without start or end as if it is traveling on the surface of a Klein bottle.

A message packet in a Klein bottle routing network consists of a message and a route list containing the identities of a few routers. Re-encryption and distributed decryption are employed in Klein bottle routing to implement masking. Namely, a message is encrypted only once by its sender using a special public key, which is the combination of the public keys of all the parties on the message's route. In the route list, identity of each router on the message's route is encrypted by its sender using a special public key, which is the combination of the public

keys of the routers before that router. In the end of the route list, identity of the receiver is encrypted by the sender using a special public key, which is the combination of the public keys of all the routers. When a router transmits and masks a packet, it re-encrypts and partially decrypts (as its part of distributed decryption) the message and the route list in the packet before forwarding it to the next router. Anonymity of the communication channel requires that identity of the receiver and identities of other routers on the route (except the next router) are confidential to each router. So the routers must re-encrypt the messages passing them and their routing information without knowing the receiver's public key or other routers' public keys. Moreover, the decryption key must be shared such that no single party can decrypt the packets and trace them. So ElGamal encryption with universal re-encryption and n-out-of-n distributed decryption described in Section 3 is employed for the routers to re-encrypt and distributedly decrypt the packets.

Navigation is a new question in Klein bottle routing. Given an encrypted route list how can a router tell who is the next router to forward a packet to? How is it possible to prevent a router from knowing the length of the route and its position on the route? A navigation mechanism called cycling encrypted route list is designed to navigate the messages when re-encryption masking is employed. When a message packet is initially generated by the sender, it includes and encrypts the identities of all the routers to pass and the receiver in the order of being visited: $N_1, N_2, \ldots, N_m, R$, which is called the encrypted route list in the packet. Each router decrypts the first ciphertext in the encrypted route list to find the next router's identity and then cycles the encrypted route list one step forward so that the identity of the router after the next router becomes the first identity in the encrypted route list. For example, the first router decrypts the first ciphertext in the initial encrypted route list and finds N_2 as the next router, then cycles the encrypted route list so that it becomes the encryption of $N_2, N_3, \ldots, N_m, R, N_1$. The i^{th} router decrypts the first ciphertext in the current encrypted route list and finds N_{i+1} as the next router, then cycles the encrypted route list so that it becomes the encryption of $N_{i+2}, N_{i+3}, \ldots, N_m, R, N_1, \ldots, N_{i+1}$. This technique guarantees efficient navigation without revealing how close a router is to the final destination of the message. If necessary, dummy routers can be added into the end of the route, so that the absolute length of the route is not revealed.

Suppose the sender is S, the receiver is R and routers $N_1, N_2, \ldots, N_m$ are employed to transmit a message s. ElGamal encryption algorithm with universal re-encryption and m-out-of-m distributed decryption is set up for each entity as described in Section 3 where common p, q, G and g are used. The public key of R is y_R and the public key of N_i is y_i. The private key of R is x_R and the private key of N_i is x_i. Klein bottle routing protocol is demonstrated in Figure 2 in Figure 2 and described as follows.

1. S chooses the route $S - N_1 - N_2 - \ldots - N_m - R$ for message s and sends packet (c_0, d_0) to N_1 where $N_1, N_2, \ldots, N_m$ are randomly chosen routers, c_0 is the encrypted message and d_0 is the encrypted route list, which is an $m+1$ dimension vector $(d_{0,1}, d_{0,2}, \ldots, d_{0,m+1})$. More precisely, the encrypted

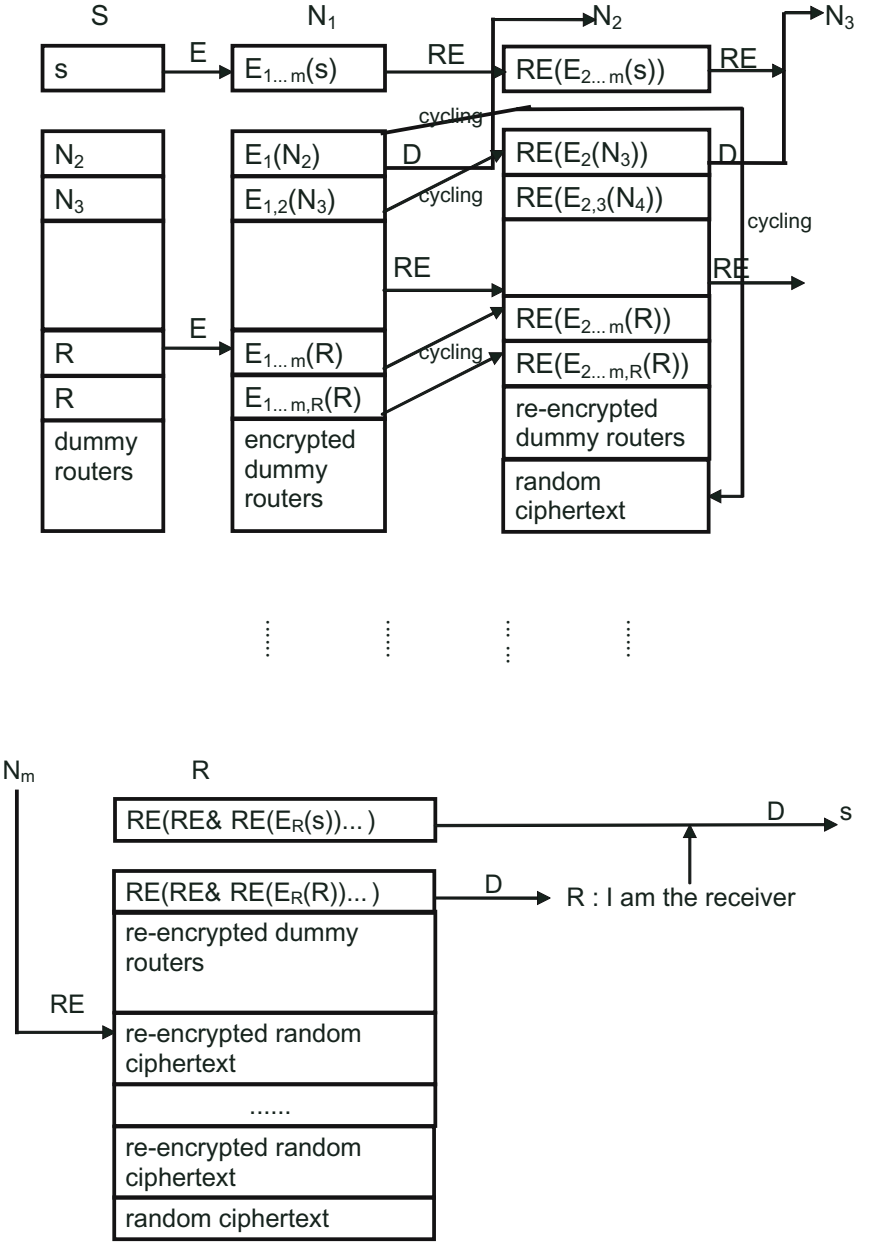


Fig. 2. Klein bottle routing

message is $c_0 = E_{N_1,N_2,\ldots,N_m,R}(s)$ and the encrypted route list is $d_0 = (d_{0,1}, d_{0,2}, \ldots, d_{0,m+1}) = (E_{N_1}(N_2),\ E_{N_1,N_2}(N_3),\ \ldots, E_{N_1,N_2\ldots,N_{m-1}}(N_m),\ E_{N_1,N_2\ldots,N_m}(R),\ E_{N_1,N_2\ldots,N_m,R}(R))$.

2. N_1 decrypts $d_{0,1}$ to find the next router's identity, N_2. Then N_1 sends (c_1, d_1) to N_2 where $c_1 = RE(D_1(c_0))$, $d_1 = (RE(D_1(d_{0,2})), \ldots, RE(D_1(d_{0,m+1})), C_1)$ and C_1 is a random ciphertext.
3. Each N_i receives (c_{i-1}, d_{i-1}), decrypts $d_{i-1,1}$ to find the next router's identity, N_{i+1}. Then N_i sends (c_i, d_i) to N_{i+1} where $c_i = RE(D_i(c_{i-1}))$, $d_i = (RE(D_i(d_{i-1,2})), \ldots, RE(D_i(d_{i-1,m+1})), C_i)$ and C_i is a random ciphertext. Namely, each router decrypts the first ciphertext in the encrypted route list to find the next router; cycles the encrypted route list one step forward; replaces the former first ciphertext in the encrypted route list with a random ciphertext and puts it in the end of the current encrypted route list; partially decrypts, re-encrypts the rest of the packet and sends the packet to the next node.
4. R decrypts the first ciphertext in the encrypted route list and finds its own name. He thus knows that he is the receiver. He then decrypts c_m and gets $s = D_R(c_m)$.

If the absolute length of each message's route must be confidential, some dummy routers can be added to the end of the initial route list when the sender generates the packet. If public verifiability is desired, the following routing operations with public proof and verification can be employed.

1. When a router receives a message packet, he usually does not route it immediately. He waits until the number of message packets he holds is over a threshold.
2. After the number of message packets he holds is over a threshold, the router routes them in a random order. We call this routing mechanism batch routing[2].
3. The router has to prove that in the last batch routing the messages encrypted in his output packets is a permutation of the messages encrypted in his input packets. This is very similar to the proof of correct shuffling in a mix network as both techniques employ batch operation, a random permutation of the packets and re-encryption masking. So the proof and verification protocol to guarantee correct re-encryption and shuffling in existing mix network schemes like [8] or [11] can be employed. To be used in Klein bottle routing, the proof and verification protocol is slightly modified to suit ElGamal encryption with universal re-encryption and n-out-of-n distributed decryption.

5 Analysis

Security of Klein bottle routing is analysed in this section.

Theorem 2. *Klein bottle routing is isolatedly anonymous.*

[2] Even if public verifiability is not required, batch routing can be employed to prevent a simple traffic analysis attack linking an input packet to a router to an immediately following output packet from the same router.

Lemma 1. *An adversary chooses two ciphertexts c_1 and c_2 in the encryption system employed in Klein bottle routing where he does not know the private keys to decrypt these two messages. A challenger randomly selects i from $\{1, 2\}$ and calculates $c' = RE(D_j(c_i))$ where $D_j()$ denotes partial decryption using N_j's private key*[3]*. Given c', the adversary can output i with a probability no more than $0.5 + \epsilon$ and ϵ is negligible.*

Proof: If Lemma 1 is incorrect, the adversary can output i with a probability $0.5 + \epsilon$ and ϵ is not negligible in the game described in Lemma 1. Then the adversary can use the following algorithm to break semantic security under re-encryption in the ElGamal encryption algorithm with universal re-encryption and n-out-of-n distributed decryption.

1. The adversary chooses ciphertexts e_1 and e_2 and calculates $c_1 = E_{N_j}(e_1)$ and $c_2 = E_{N_j}(e_2)$.
2. The adversary sends c_1 and c_2 to the challenger.
3. The challenger randomly selects i from $\{1, 2\}$ and calculate $c' = RE(D_j(c_i))$
4. c' is given to the adversary while i is kept secret.
5. The adversary can output i with a probability $0.5+\epsilon$ where ϵ is not negligible as Lemma 1 is assumed to be incorrect.

Note that $c' = RE(D_j(c_i)) = RE(D_j(E_{N_j}(e_i))) = RE(e_i)$. So using this algorithm the adversary can output i given c' without knowledge of the private keys where c' is the re-encryption of one of the two ciphertexts e_1 and e_2. This is contradictory to Theorem 1. Therefore, Lemma 1 is correct. □

Proof of Theorem 2:
If Theorem 2 is incorrect, an adversary without help of the router can output i with a probability $0.5 + \epsilon$ and ϵ is not negligible in the following game: the adversary chooses two input packets p_1 and p_2 for a router N_k, then the router randomly selects i from $\{1, 2\}$ and routes p_i to an output packet p'.

Suppose p_i contains (c_i, d_i) and p' contains (c', d') where c_i and c' are the encrypted messages in the packets; d_i and d' are the encrypted route lists; $d_i = (d_{i,1}, d_{i,2}, \ldots, d_{i,m+1})$ and $d' = (d'_1, d'_2, \ldots, d'_{m+1})$. So at least one of the two following events happen.

- Given c', a re-encryption of either c_1 or c_2, the adversary outputs i such that c' is a re-encryption of c_i with a probability $0.5 + \epsilon$ and ϵ is not negligible.
- Given $d'_{j-1 \bmod m+2}$, a re-encryption of either $d_{1,j}$ or $d_{2,j}$ where $1 \leq j \leq m + 1$, the adversary outputs i such that $d'_{j-1 \bmod m+2}$ is a re-encryption of $d_{i,j}$ with a probability $0.5 + \epsilon$ and ϵ is not negligible.

Note that $c' = RE(D_k(c_i))$ and $d'_{j-1 \bmod m+2} = RE(D_k(d_{i,j}))$ and the adversary does not know the private key to decrypt c_1, c_2, $d_{1,j}$ or $d_{2,j}$ as at least N_k does not collude with him. So both these two events are contradictory to Lemma 1. Therefore, Theorem 2 is correct. □

[3] For example, N_j acts as the challenger.

Theorem 2 illustrates that Klein bottle routing is anonymous if the decryption key is unknown and traffic analysis is not performed. As n-out-of-n key sharing is employed, no information about the decryption key is revealed unless all the routers and the receiver collude. Although traffic analysis cannot be completely prevented, it is not easy in Klein bottle routing. As dummy routers are added to the end of each route list, each encrypted route list seems to have the largest possible length. In Klein bottle routing, each router has three operations on a message packet: partial decryption, re-encryption and route list cycling, none of which changes the size of any message packet. So the size of any message packet is constant when it travels in the Klein bottle routing network. Constant packet size achieved in Klein bottle routing makes traffic analysis more difficult. The traffic analysis attack based on monitoring packet size mentioned in Section 2 can be prevented in a Klein bottle routing based communication network. The public proof and verification techniques in mix network [8,11] can be slightly modified and employed to provide public verification of correct routing when batch routing is employed. So Klein bottle routing is publicly verifiable. It is the only known publicly verifiable anonymous routing scheme so far. Klein bottle routing is simple and efficient as it avoids a difficult question in onion routing: keeping packet size unchangeable when decryption chain masking is used. The cycling-route-list-based navigation mechanism guarantees efficient navigation (only one decryption and a small number of re-encryptions are needed to find the next router) without revealing to the router its position on the route.

To prevent the interception-insertion attack in [2], we only need a small change to the packet format: the encrypted message is $c_0 = E_{N_1,N_2,\ldots,N_{m-2},N_m,R}(s)$ and the encrypted route list is $d_0 = (d_{0,1}, d_{0,2}, \ldots, d_{0,m+1}) = (E_{N_1}(N_2),\ E_{N_2}(N_3),$ $\ldots, E_{N_1,N_2\ldots,N_{m-3},N_{m-1}}(N_m),\ E_{N_1,N_2\ldots,N_{m-2},N_m}(R),$
$E_{N_1,N_2\ldots,N_{m-2},N_m,R}(R))$. Namely, one key is deleted from each encryption. Thus there is a gap in the address list and the inserted address in the attack will be trapped in the gap and cannot be decrypted. Therefore, the attack can be prevented.

In Table 1, a comparison between Klein bottle routing and the existing anonymous channel implementations is provided. When computation cost is compared, the number of full length exponentiations are counted. Although the concrete number of exponentiations cannot be given in Table 1 due to lack of details in some existing schemes, the cost of the schmemes in the table is similar. It is clearly illustrated that Klein bottle routing overcomes the drawback of onion routing and mix network. It is the only solution for general and flexible anonymous communication channels.

An interesting property of the new scheme is that it can be easily extended to support public verifiability. Any exsting re-encryption based public verifiability mechanism [12,11] can be easily employed to achieve public verifiability. So the new scheme is the first technique to support public verifiability in free-route anonymous networks. In a publicly verifiable free-route network, each router has to give a public proof of validity of his routing periodicly, while an auditing

authority checks the proof. As the proof is publicly verifiable, any dispute can be publicly solved.

Table 1. Comparison

scheme	general & flexible	anonymity	packet size based traffic analysis	publicly verifiable	cost
onion routing [5,6]	yes	informal	vulnerable	no	$O(m)$
onion routing [1]	yes	formal	invulnerable	no	$O(m)$
onion routing [9]	yes	informal	invulnerable	no	$O(m)$
mix network [11]	no	formal	invulnerable	yes	$O(m)$
Klein bottle routing	yes	formal	invulnerable	supported	$O(m)$

6 Conclusion

A new type of anonymous channel, Klein bottle routing, is designed in this paper. It overcomes the drawbacks of onion routing and mix network and fills the gap between them. Klein bottle routing is the only flexible general anonymous channel implementation with public verifiability. It employs batch routing and keeps the packet size unchangeable to avoid simple traffic analysis attacks.

Acknowledgements

We acknowledge the inspiring suggestion of Colin Boyd. This research work is Partially Sponsored by NICT, Japan.

References

1. J Camenisch and A Lysyanskaya. A formal treatment of onion routing. In *CRYPTO '05*, Lecture Notes in Computer Science, pages 169–187, Berlin, 2005. Springer-Verlag.
2. G Danezis. Breaking Four Mix-related Schemes Based on Universal Re-encryption. To appear *ISC06*.
3. Y Desmedt and Y Frankel. Threshold cryptosystems. In *Crypto89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315, Berlin, 1989. Springer-Verlag.
4. J Furukawa and K Sako. An efficient scheme for proving a shuffle. In *CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387, Berlin, 2001. Springer.
5. O Goldreich, S Micali, and A Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC 1987*, pages 218–229, 1987.
6. D Goldschlag, M Reed, and P Syverson. Onion routing for anonymous and private internet connections. *Comm. of the ACM, 42(2)*, page 8488, 1999.
7. P Golle, M Jakobsson, A Juels, and P Syverson. Universal re-encryption for mixnets. In *CT-RSA 2004*, pages 163–178, 2004.

8. J Groth. A verifiable secret shuffle of homomorphic encryptions. In *Public Key Cryptography 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 145–160, Berlin, 2003. Springer-Verlag.
9. M Kutylowski M Gomulkiewicz, M Klonowski. Onions based on universal re-encryption - anonymous communication immune against repetitive attack. In *WISA2004*, volume 3325 of *Lecture Notes in Computer Science*, pages 400–410, Berlin, 2004. Springer-Verlag.
10. C Neff. A verifiable secret shuffle and its application to e-voting. In *ACM Conference on Computer and Communications Security 2001*, pages 116–125, 2001.
11. K Peng, C Boyd, and E Dawson. Simple and efficient shuffling with provable correctness and ZK privacy. In *CRYPTO '05*, volume 3089 of *Lecture Notes in Computer Science*, pages 188–204, Berlin, 2005. Springer-Verlag.
12. K Peng, C Boyd, E Dawson, and K Viswanathan. A correct, private and efficient mix network. In *2004 International Workshop on Practice and Theory in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 439–454, Berlin, 2004. Springer-Verlag.
13. K Peng, C Boyd, E Dawson, and K Viswanathan. Efficient implementation of relative bid privacy in sealed-bid auction. In *WISA 2003*, volume 2908 of *Lecture Notes in Computer Science*, pages 244-256, Berlin, 2003. Springer-Verlag.
14. K Viswanathan, C Boyd, and E Dawson. A three phased schema for sealed bid auction system design. In *Information Security and Privacy, 5th Australasian Conference, ACISP'2000*, volume 1841 of *Lecture Notes in Computer Science*, pages 412–426, Berlin, 2000. Springer-Verlag.

New Constructions of Constant Size Ciphertext HIBE Without Random Oracle

Sanjit Chatterjee and Palash Sarkar

Applied Statistics Unit
Indian Statistical Institute
203, B.T. Road, Kolkata
India 700108
{sanjit_t, palash}@isical.ac.in

Abstract. At Eurocrypt 2005, Boneh-Boyen-Goh presented an interesting and important construction of a constant size ciphertext HIBE. The HIBE was proven to be secure in the selective-ID model. In this paper, we present two variants of the BBG-HIBE secure in more general security models. The first variant is proved to be secure in a generalization of the selective-ID model while the second variant is proved to be secure in the full security model. Our constructions are not straightforward modifications of the BBG-HIBE. Several techniques have to be suitably combined to obtain the required proofs.

1 Introduction

Identity based encryption (IBE) is a kind of public key encryption where the public key can be any binary string. The corresponding private key is generated by a private key generator (PKG) and securely transmitted to the proper entity. IBE was introduced by Shamir [17] and an appropriate security model and construction was given in [4]. The notion of IBE was extended to the notion of a hierarchical IBE (HIBE) in [15,14]. A HIBE reduces the work load of the PKG by allowing key generation to be delegated to lower levels.

Since the publication of [4], there have been several papers proposing constructions of IBE and HIBE which are secure in different security models and under different hardness assumptions. In a recent work, Boneh-Boyen-Goh [3] present a HIBE where the ciphertext expansion consists of only two elliptic curve points irrespective of the number of components in the corresponding identity. In other HIBEs, the ciphertext expansion is proportional to the length of the identity. The BBG-HIBE offers new and important applications for constructing other cryptographic primitives. The security of the BBG-HIBE is based on a variant of the DBDH-assumption called the wDBDHI-assumption. Further, the proof is in the selective-ID (sID) model which was introduced in [8,9]. The sID-model is significantly weaker than the full security model for HIBEs.

In this paper, we present two variants of the BBG-HIBE. The aim of these variants is to be able to prove security in models which are stronger than the sID-model. The first variant (called ccHIBE) is secure in a generalization of the

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 310–327, 2006.
© Springer-Verlag Berlin Heidelberg 2006

sID-model (called model $\boldsymbol{\mathcal{M}}_2$) introduced in [11]. The second variant (called FullccHIBE) is secure in the full model. The reduction for the full model security is not tight. The security degradation is along the lines of the suggestion in [18,3].

Security in the stronger models is attained at the cost of increasing the size of the public parameters and an increase in the computation time of the key generation and encryption. On the other hand, the ciphertext expansion as well as the decryption efficiency remains the same as that in the BBG-HIBE. In a HIBE, the entire private key for an identity is not necessarily required for decryption. The portion of the private key required for decryption is called the decryption subkey. The entire key is required for key delegation. In the BBG-HIBE, the decryption subkey consists of only two group elements. This feature is retained in the new constructions, even though the entire private key is longer than that of the BBG-HIBE. The smallness of the decryption subkey is important, since it might be required to load it onto a smart card. The growth of the entire private key is less significant since key delegation is a relatively infrequent activity.

Constant size ciphertext HIBE is an important cryptographic primitive. To some extent, the importance of the primitive justifies the variants that we present. On the other hand, from a technical point of view, our variants are not straightforward extensions of the BBG-HIBE. There are certain technical subtleties which need to be taken care of for the proofs to go through. Below we provide a description of some of these aspects.

In model $\boldsymbol{\mathcal{M}}_2$, as in the sID-model, the adversary specifies a positive integer $\tau \leq h$, which is the length of the challenge identity. For the ith ($1 \leq i \leq \tau$) level of the HIBE, the adversary commits to a set of values $\mathcal{I}_i^*$. It is not allowed to query the key extraction oracle on an identity $(\mathsf{v}_1, \ldots, \mathsf{v}_j)$, where $1 \leq j \leq \tau$ and $\mathsf{v}_i \in \mathcal{I}_i^*$ for each $1 \leq i \leq j$. On the other hand, as the challenge identity, the adversary submits $(\mathsf{v}_1^*, \ldots, \mathsf{v}_\tau^*)$ where $\mathsf{v}_i^* \in \mathcal{I}_i^*$ for $1 \leq i \leq \tau$. The sID-model is a special case of model $\boldsymbol{\mathcal{M}}_2$. This can be seen by fixing each of the sets $\mathcal{I}^*$ to be singleton sets. In general terms, in model $\boldsymbol{\mathcal{M}}_2$, the adversary gains more flexibility in choosing the challenge identity.

The BBG-HIBE has been proved to be secure in the sID-model. We augment this HIBE to attain security in model $\boldsymbol{\mathcal{M}}_2$. A similar construction has been done for the BB-HIBE in [11]. The main technical novelty in the proof is the use of a polynomial which in [3] is of degree one. The other problem is that the security of the BBG-HIBE is based on the wDBDHI* problem. An instance of this problem consists of a tuple $(P, Q, aP, a^2P, \ldots, a^hP, Z)$, where P, Q are points from an appropriate elliptic curve group and Z is a suitable finite field element. This instance is more complicated than an instance (P, aP, bP, cP, Z) of DBDH. Properly combining the polynomial-based security proof from [11] with the wDBDHI* problem is the main technical difficulty in the proof of security in model $\boldsymbol{\mathcal{M}}_2$.

The public parameters in the BBG-HIBE are $(P, P_1, P_2, P_3, Q_1, \ldots, Q_h)$. In extending the BBG-HIBE to attain security in model $\boldsymbol{\mathcal{M}}_2$, we have to replace each Q_i by a tuple $(Q_{i,1}, \ldots, Q_{i,n_i})$. The parameter P_3 does not change. The public parameters of the new HIBE are $(P, P_1, P_2, P_3, \overrightarrow{Q}_1, \ldots, \overrightarrow{Q}_h)$, where

$\overrightarrow{Q}_i = (Q_{i,1}, \ldots, Q_{i,n_i})$. The Q-parameters capture the dependence on the level in both the BBG-HIBE and in its extension to $\boldsymbol{\mathcal{M}}_2$. The parameters P_1, P_2, P_3 do not depend on the number of levels of the HIBE.

On the other hand, when we modify the BBG-HIBE to attain security in the full model, we need to change the parameter P_3 to a tuple $\overrightarrow{P}_3 = (P_{3,1}, \ldots, P_{3,h})$. The new $\overrightarrow{P}_3$ depends on the number of levels of the HIBE. The public parameters in this case are of the form $(P, P_1, P_2, \overrightarrow{P}_3, \overrightarrow{Q}_1, \ldots, \overrightarrow{Q}_h)$, where $\overrightarrow{Q}_i = (Q_{i,1}, \ldots, Q_{i,l})$.

It has been mentioned in [3] that the BBG-HIBE protocol can be modified as in [18] to attain security in the full model. The change to P_3 forms a part of this modification, which was perhaps not anticipated in [3]. Adapting the techniques of [18,10,16] to the more complicated wDBDHI* assumption is the main technical difficulty in the proof of security in the full model.

2 Preliminaries

2.1 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 $a, b \in \mathbb{Z}_p$.
- 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. Weil pairing [4] and Tate pairing [1,13] are examples of cryptographic bilinear maps.

The known examples of $e()$ have G_1 to be a group of Elliptic Curve (EC) points and G_2 to be a subgroup of a multiplicative group of a finite field. Hence, in papers on pairing implementations [1,13], it is customary to write G_1 additively and G_2 multiplicatively. On the other hand, some "pure" protocol papers [2,18] write both G_1 and G_2 multiplicatively, though this is not true of the early protocol papers [4,14]. Here we follow the first convention as it is closer to the known examples.

2.2 Hardness Assumption

Weak Decisional Bilinear Diffie-Hellman Inversion (wDBDHI*) Problem: This problem was introduced by Boneh-Boyen-Goh in [3]. An instance of the h-wDBDHI* problem over $\langle G_1, G_2, e() \rangle$ consists of a tuple

$$\langle P, Q, aP, a^2P, \ldots, a^hP, Z \rangle$$

for some $a \in \mathbb{Z}_p$ and the task is to decide whether $Z = e(P, Q)^{a^{h+1}}$ or Z is random.

Let $\mathcal{B}$ be a probabilistic algorithm that takes as input an instance of the h-wDBDHI* problem and outputs a bit. The advantage of $\mathcal{B}$ is defined to be

$$\mathsf{Adv}_{\mathcal{B}}^{h\text{-wDBDHI}^*} = \left| Pr[\mathcal{B}(P, Q, \overrightarrow{Y}, e(P,Q)^{a^{h+1}}) = 1] - Pr[\mathcal{B}(P, Q, \overrightarrow{Y}, R) = 1] \right|$$

where $\overrightarrow{Y} = (aP, a^2P, \ldots a^hP)$ and R is a random element of G_2. The probabilities are calculated over the random choices of $a \in \mathbb{Z}_p$ and $R \in G_2$ and also over the random bits used by $\mathcal{B}$. The quantity $\mathsf{Adv}^{h\text{-wDBDHI}^*}(t)$ denotes the maximum of $\mathsf{Adv}_{\mathcal{B}}^{h\text{-wDBDHI}^*}$ where the maximum is taken over all adversaries running in time at most t. We will say that the h-wDBDHI* problem is (ϵ, t)-hard if $\mathsf{Adv}^{h\text{-wDBDHI}^*}(t) \leq \epsilon$.

2.3 HIBE Protocol

A hierarchical identity based encryption (HIBE) scheme is specified by four probabilistic algorithms: Setup, Key Generation, Encryption and Decryption. An identity of depth j is a tuple $(\mathsf{v}_1, \ldots, \mathsf{v}_j)$, where each v_i is an element of a set $\mathcal{I}$.

Setup: The input is a security parameter and the output consists of the public parameters of the system along with the master key. The master key is known only to the private key generator (PKG). The message space, ciphertext space and the identity space are also defined during setup.

Key Generation: The task of this algorithm is to assign a private key d_{v} for an identity v of depth τ. It takes as input an identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_\tau)$ of depth τ and the private key $d_{|\tau-1}$ corresponding to the identity $\mathsf{v}_{|\tau-1} = (\mathsf{v}_1, \ldots, \mathsf{v}_{\tau-1})$ and returns d_{v}. In the case $\tau = 1$, the private key $d_{|\tau-1}$ is the master key of the PKG and the key generation is done by the PKG. In the case $\tau > 1$, the private key corresponding to $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_\tau)$ is done by the entity whose identity is $\mathsf{v}_{|\tau-1} = (\mathsf{v}_1, \ldots, \mathsf{v}_{\tau-1})$ and who has already obtained his/her private key $d_{|\tau-1}$.

Encryption: The encryption algorithm takes as input the identity v, the public parameters of the PKG and a message from the message space and produces a ciphertext in the cipher space.

Decryption: The decryption algorithm takes as input the ciphertext, the identity v under which encryption has been performed, the private key d_{v} of the corresponding identity v and the public parameters. It returns the message or bad if the ciphertext is not valid.

2.4 Security Models

The security of a HIBE protocol is defined in terms of a game between an adversary and a simulator. The full security model for IBE was introduced in [4]

and the extension to HIBE was given in [14]. The weaker selective-ID model was introduced in [8,9]. The security models $\boldsymbol{\mathcal{M}}_1$ and $\boldsymbol{\mathcal{M}}_2$ were introduced in [11]. We first describe the full security model and then point out the modifications required for the sID-model and $\boldsymbol{\mathcal{M}}_2$.

The adversary is given access to two oracles – the key extraction oracle and the decryption oracle. The key extraction oracle takes as input an identity and returns as output a random private key corresponding to the identity. The decryption oracle takes as input an identity and a ciphertext and returns as output either bad or the corresponding message. We assume that the adversary does not make any pointless query, i.e., any query for which it can obtain the output by itself. For example, the adversary does not query the decryption oracle with an identity for which it has already obtained a private key. The game proceeds in several phases.

Phase 1: In this phase, the adversary can query the key extraction oracle and the decryption oracle in an adaptive manner.

Challenge: At the end of Phase 1, the adversary outputs an identity v^* and two equal length messages M_0 and M_1. A random bit b is chosen and the adversary is given an encryption C^* of M_b under the identity v^*. There is the natural restriction that the adversary has not obtained a private key for any prefix of v^* in Phase 1 and will also not do so in Phase 2.

Phase 2: In this phase, the adversary continues making key extraction and decryption queries with the following additional restriction. It cannot query the decryption oracle with the pair (v^*, C^*).

Guess: At the end of Phase 2, the adversary outputs a guess b' of b.

The adversary wins the game if $b = b'$. The advantage of an adversary $\mathcal{A}$ in attacking the HIBE scheme is defined as:

$$\mathsf{Adv}^{\mathsf{HIBE}}_{\mathcal{A}} = |\Pr[(b = b')] - 1/2| \,.$$

The quantity $\mathsf{Adv}^{\mathsf{HIBE}}(t, q_{\mathsf{ID}}, q_{\mathsf{C}})$ denotes the maximum of $\mathsf{Adv}^{\mathsf{HIBE}}_{\mathcal{A}}$ where the maximum is taken over all adversaries running in time at most t and making q_{C} queries to the decryption oracle and q_{ID} queries to the key-extraction oracle. Any HIBE scheme secure against such an adversary is said to be secure against chosen ciphertext attack (CCA-secure).

We may restrict the adversary from making any query to the decryption oracle. A HIBE protocol secure against such an adversary is said to be secure against chosen plaintext attacks (CPA-secure). $\mathsf{Adv}^{\mathsf{HIBE}}(t, q)$ 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.

There are generic [8,9,5] as well as non-generic [6] techniques for converting a CPA-secure HIBE to a CCA-secure HIBE. In view of this, it is more convenient to initially construct a CPA-secure HIBE and then convert it into a CCA-secure one.

2.5 Selective-ID Model

The selective-ID model restricts the adversary's behaviour. The main difference is that in the adversarial game for the sID model, the adversary has to commit to the challenge identity v^* even before the protocol is setup. Thus, the actual protocol setup in the game can depend on v^*, though undetectable by the adversary. As before, in Phases 1 and 2, the adversary is not allowed to query the key extraction oracle for a private key of any prefix of v^*. The other parts of the game and the other restrictions are as in the case of full model security.

Requiring the adversary to commit to the challenge identity before setup is a significant restriction. On the other hand, it is easier to obtain efficient protocols with tight security reductions in the sID model.

2.6 Generalised Selective-ID Models

As mentioned earlier, two new security models, $\boldsymbol{\mathcal{M}}_1$ and $\boldsymbol{\mathcal{M}}_2$ have recently been introduced in [11]. Here we describe only $\boldsymbol{\mathcal{M}}_2$ since this is the model that we require.

$\boldsymbol{\mathcal{M}}_2$ generalises sID model in the following manner. Before the set-up of the protocol, the adversary commits to sets of identities $\mathcal{I}_1^*, \ldots, \mathcal{I}_\tau^*$, where $1 \leq \tau \leq h$ and h is the maximum number of levels of the HIBE. Let $|\mathcal{I}_i^*| = n_i$. The adversary's commitment fixes the length of the challenge identity to be τ. Also, the set $\mathcal{I}_i^*$ corresponds to the set of committed identities for the ith level of the HIBE.

In Phases 1 and 2, the adversary is not allowed to query the key extraction oracle on any identity $(\mathsf{v}_1, \ldots, \mathsf{v}_j)$ such that $j \leq \tau$ and $\mathsf{v}_i \in \mathcal{I}_i^*$ for all $1 \leq i \leq j$. The challenge identity is a tuple $(\mathsf{v}_1^*, \ldots, \mathsf{v}_\tau^*)$ where $\mathsf{v}_i^* \in \mathcal{I}_i^*$ for all $1 \leq i \leq \tau$.

The model $\boldsymbol{\mathcal{M}}_2$ is parametrized by h and a tuple $(n_1, \ldots, n_h)$ of positive integers. This is explictly written as $(h, n_1, \ldots, n_h)$-$\boldsymbol{\mathcal{M}}_2$ model. This model is a generalization of the sID-model which can be seen by fixing all the $\mathcal{I}_i^*$s to be singleton sets. More specifically, $(h, 1, \ldots, 1)$-$\boldsymbol{\mathcal{M}}_2$ is the sID-model.

2.7 BBG-HIBE [3]

Let G_1, G_2 and $e()$ be as defined in Section 2.1. The maximum depth of the HIBE is a positive integer h. Identities at depth τ, with $1 \leq \tau \leq h$ are of the form $(\mathsf{v}_1, \ldots, \mathsf{v}_\tau)$ where each $\mathsf{v}_i \in \mathbb{Z}_p^*$. Messages are elements of G_2.

Setup: Choose a random $\alpha \in \mathbb{Z}_p$ and set $P_1 = \alpha P$. Choose random elements $P_2, P_3, Q_1, \ldots, Q_h \in G_1$. Set the public parameters to be

$$(P, P_1, P_2, P_3, Q_1, \ldots, Q_h)$$

while the master key is αP_2.

Key Generation: Given an identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_k)$ of depth $k \leq h$, pick a random $r \in \mathbb{Z}_p$ and output

$$d_{\mathsf{v}} = (\alpha P_2 + r(\mathsf{v}_1 Q_1 + \ldots + \mathsf{v}_k Q_k + P_3), rP, rQ_{k+1}, \ldots, rQ_h).$$

The private key for v can also be generated given the private key for $\mathsf{v}_{|k-1}$ as is the requirement of a HIBE (see [3] for the details).

Encrypt: To encrypt $M \in G_2$ under the identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_k) \in (\mathbb{Z}_p)^k$, pick a random $s \in \mathbb{Z}_p$ and output

$$(e(P_1, P_2)^s \times M, sP, s(\mathsf{v}_1 Q_1 + \ldots + \mathsf{v}_k Q_k + P_3)).$$

Decrypt: To decrypt (A, B, C) using the private key $d_{\mathsf{v}} = (a_0, a_1, b_{k+1}, \ldots, b_h)$, compute

$$A \times \frac{e(a_1, C)}{e(B, a_0)} = M.$$

3 ccHIBE

Setup: Let $\langle G_1, G_2, e \rangle$ with $G_1 = \langle P \rangle$ be as in Section 2.1. The maximum depth of the HIBE is a prespecified value denoted by h. Identities are of the form $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_\tau)$ with $1 \leq \tau \leq h$ and each $\mathsf{v}_i \in \mathbb{Z}_p^*$. Messages are elements of G_2.

Let $(n_1, \ldots, n_h)$ be a tuple of integers. Choose a random $\alpha \in \mathbb{Z}_p$ and set $P_1 = \alpha P$. Choose random points P_2, P_3 from G_1. The public parameters are $(P, P_1, P_2, P_3, \overrightarrow{Q}_1, \ldots, \overrightarrow{Q}_h)$ where $\overrightarrow{Q}_i = (Q_{i,1}, \ldots, Q_{i,n_i})$. Each $Q_{i,j}$ is randomly chosen from G_1. The master secret is αP_2.

Notation: For ease of description, we define a notation.

$$V_i(y) = y^{n_i} Q_{i,n_i} + \cdots + yQ_{i,1}.$$

Let $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$ be an identity. By V_i we will denote $V_i(\mathsf{v}_i)$.

Key Generation: Given an identity $(\mathsf{v}_1, \ldots, \mathsf{v}_\tau)$, pick a random $r \in \mathbb{Z}_p$ and output

$$d_{\mathsf{v}} = \left(\alpha P_2 + r \left(P_3 + \sum_{j=1}^{\tau} V_j \right),\ rP,\ r\overrightarrow{Q}_{\tau+1}, \ldots, r\overrightarrow{Q}_h \right)$$

where $r\overrightarrow{Q}_i = (rQ_{i,1}, \ldots, rQ_{i,n_i})$. A private key at level τ consists of $(2 + \sum_{i=\tau+1}^{h} n_i)$ elements of G_1. Among these, only the first two are required in decryption, the rest are used to generate a private key for the next level as follows:

Let a private key for $(\mathsf{v}_1, \ldots, \mathsf{v}_{\tau-1})$ be $(A'_0, A'_1, \overrightarrow{B}'_\tau, \ldots, \overrightarrow{B}'_h)$, where

$$A'_0 = \alpha P_2 + r' \left(\sum_{j=1}^{\tau-1} V_j + P_3 \right),$$

$A'_1 = r'P$, and for $\tau \leq j \leq h$, $\overrightarrow{B}'_j = (r'Q_{j,1}, \ldots, r'Q_{j,n_j})$. Let $B'_{j,k} = r'Q_{j,k}$. Pick a random $r^* \in \mathbb{Z}_p$ and compute $d_{\mathsf{v}} = (A_0, A_1, \overrightarrow{B}_{\tau+1}, \ldots, \overrightarrow{B}_h)$ where

$$\begin{aligned} A_0 &= A'_0 + \textstyle\sum_{i=1}^{n_\tau} \mathsf{v}_\tau^i B'_{\tau,i} + r^* \left(\sum_{j=1}^{\tau} V_j + P_3 \right), \\ A_1 &= A'_1 + r^* P, \\ B_{\tau+1} &= \overrightarrow{B}'_{\tau+1} + r^* \overrightarrow{Q}_{\tau+1}, \\ &\ldots, \\ B_h &= \overrightarrow{B}'_h + r^* \overrightarrow{Q}_h. \end{aligned}$$

If we put $r = r' + r^*$, then d_{v} is a proper private key for $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_\tau)$.

Encrypt: To encrypt $M \in G_2$ under the identity $(\mathsf{v}_1, \ldots, \mathsf{v}_\tau)$, pick a random $s \in \mathbb{Z}_p$ and output

$$\left(e(P_1, P_2)^s \times M, sP, s \left(P_3 + \sum_{j=1}^{\tau} V_j \right) \right).$$

Decrypt: To decrypt (A, B, C) for $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_\tau)$ using the private key $d_{\mathsf{v}} = (d_0, d_1, \ldots)$, compute

$$A \times \frac{e(d_1, C)}{e(B, d_0)} = e(P_1, P_2)^s \times M \times \frac{e\left(rP, s\left(P_3 + \sum_{j=1}^{\tau} V_j\right)\right)}{e\left(sP, \alpha P_2 + r\left(P_3 + \sum_{j=1}^{\tau} V_j\right)\right)} = M.$$

Note: Here ccHIBE is parametrized by $(n_1, \ldots, n_h)$ and $(h, n_1, \ldots, n_h)$-ccHIBE is used to explicitly denote this parametrization.

3.1 Security Reduction for ccHIBE

We wish to show that ccHIBE is secure in model $\boldsymbol{\mathcal{M}}_2$. Recall that Adv is used to denote the advantage of an adversary in attacking a HIBE. By the notation $\mathsf{Adv}^{(h,n_1,\ldots,n_h)\text{-ccHIBE}}_{(h,n'_1,\ldots,n'_h)\text{-}\boldsymbol{\mathcal{M}}_2}(t,q)$ we will denote the maximum advantage of an adversary which runs in time t and makes q key-extraction queries in attacking $(h, n_1, \ldots, n_h)$-ccHIBE in the model $(h, n'_1, \ldots, n'_h)$-$\boldsymbol{\mathcal{M}}_2$.

Theorem 1. *Let $h, n_1, \ldots, n_h, q$ be positive integers and $n'_1, \ldots, n'_h$ be another set of positive integers with $n'_i \leq n_i$ for $1 \leq i \leq h$. Then*

$$\mathsf{Adv}^{(h,n_1,\ldots,n_h)\text{-ccHIBE}}_{(h,n'_1,\ldots,n'_h)\text{-}\boldsymbol{\mathcal{M}}_2}(t,q) \leq \mathsf{Adv}^{h\text{-wDBDHI}^*}(t + O(\sigma n q))$$

where $n = \sum_{i=1}^{h} n_i$ and σ is the time for a scalar multiplication in G_1.

The proof is provided in Appendix A.

4 FullccHIBE

In this section, we consider the problem of constructing a constant size ciphertext HIBE which is secure in the full model. Our construction is based on the IBE scheme given by Waters [18] and its generalization given in [10,16]. We note that the possibility of obtaining such a constant size ciphertext HIBE based on the work in [18] was mentioned as a passing remark in [3] though no details were provided.

Let the maximum height of the HIBE be h. Any identity v at height $k \leq h$ is represented by a k-tuple, $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_k)$ where each $\mathsf{v}_i = (\mathsf{v}_{i,1}, \ldots, \mathsf{v}_{i,l})$ and $\mathsf{v}_{i,j}$ is an (n/l)-bit string. In other words, an n-bit identity at each level is represented as l blocks each of length n/l bits. This manner of representing identities is used in [10,16] which generalizes the construction in [18], where $l = n$.

In our construction, the public parameter size depends on both the size parameter l and the height h of the HIBE. If we decrease the value of l, the public parameter size also decreases. However, the security of the HIBE degrades as l decreases. Hence, the decrease in the size of the public parameters comes at an increase in the security degradation. This trade-off can be converted into a trade-off between the memory required to store the public parameters and the time required for the different operations. This space/time trade-off has been studied in details in [10]. A similar space/time trade-off also holds for the present case.

4.1 Construction

Setup: Choose a random secret $\alpha \in \mathbb{Z}_p$ and set $P_1 = \alpha P$. Randomly choose P_2; an h-length vector $\overrightarrow{P_3} = (P_{3,1}, \ldots, P_{3,h})$; and h many l-length vectors $\overrightarrow{U_1}, \ldots, \overrightarrow{U_h}$ from G_1, where each $\overrightarrow{U_j} = (U_{j,1}, \ldots, U_{j,l})$. The public parameters consist of the elements

$$\langle P, P_1, P_2, \overrightarrow{P_3}, \overrightarrow{U_1}, \ldots, \overrightarrow{U_h} \rangle$$

while the master secret is αP_2. Note that, for each level i of the HIBE we have $l + 1$ elements i.e., $P_{3,i}$ and $\overrightarrow{U_i}$.

Notation: Let v_j be an n-bit string written as $\mathsf{v}_j = (\mathsf{v}_{j,1}, \ldots, \mathsf{v}_{j,l})$, where each $\mathsf{v}_{j,i}$ is an (n/l)-bit string. Define

$$V_j = P_{3,j} + \sum_{i=1}^{l} \mathsf{v}_{j,i} U_{j,i}.$$

The modularity introduced by this notation is useful in describing the protocol.

Key Generation: Given an identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_k)$ for $k \leq h$, this algorithm generates the private key d_{v} of v as follows. Choose a random element $r \in \mathbb{Z}_p$ and output

$$d_{\mathsf{v}} = \left(xP_2 + r \left(\sum_{j=1}^{k} V_j \right),\ rP,\ rP_{3,k+1}, \ldots, rP_{3,h},\ r\overrightarrow{U}_{k+1}, \ldots, r\overrightarrow{U}_h \right)$$

where $r\overrightarrow{U_j} = (rU_{j,1}, \ldots, rU_{j,l})$.

The private key for v can also be generated given the private key for any of its ancestors as is the general requirement for a HIBE scheme. Let the private key for $(\mathsf{v}_1, \ldots, \mathsf{v}_{k-1})$ be $(d'_0, d'_1, a'_k, \ldots, a'_h, \overrightarrow{b'}_k, \ldots, \overrightarrow{b'}_h)$. Pick a random $r' \in \mathbb{Z}_p$ and then $d_{\mathsf{v}} = (d_0, d_1, a_{k+1}, \ldots, a_h, \overrightarrow{b}_{k+1}, \ldots, \overrightarrow{b}_h)$, where

$$d_0 = d'_0 + a'_k + \sum_{i=1}^{l} \mathsf{v}_{k,i} b_{k,i} + r' \sum_{j=1}^{k} V_j;$$
$$d_1 = d'_1 + r'P;$$

and for $k+1 \leq j \leq h$

$$a_j = a'_j + r' P_{3,j}; \qquad \overrightarrow{b_j} = \overrightarrow{b'_j} + r' \overrightarrow{U_j}.$$

Encrypt: To encrypt a message $M \in G_2$ under the public key $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_k)$ choose a random $s \in \mathbb{Z}_p$ and then the cipher text is

$$C = \left(e(P_1, P_2)^s \times M, sP, s \sum_{j=1}^{k} V_j \right)$$

where V_j is as defined in Key Generation part.

Decrypt: Let (A, B, C) be a ciphertext and $\mathsf{v} = \mathsf{v}_1, \ldots, \mathsf{v}_k$ be the corresponding identity. Then we decrypt using $d_{\mathsf{v}} = (d_0, d_1, \ldots)$ as

$$A \times \frac{e(d_1, C)}{e(B, d_0)} = M.$$

Note that, only the first two components of the private key are required for the decryption.

4.2 Security

Security of the FullccHIBE scheme described above can be reduced from the hardness of the h-wDBDHI* problem. The reduction combines ideas from the proof in Section 3.1 with ideas from the proofs in [18] and [10,16]. In particular, the general idea of tackling adaptive adversaries including an "artificial abort" stage is from [18], the modification for the case of $1 < l \leq n$ is from [10,16], whereas the idea of the simulation of the key-extraction queries is from the proof in Section 3.1 and is based on algebraic techniques originally used by Boneh and Boyen [2]. To explain this idea further, the simulator in the proof will abort on certain queries made by the adversary and also on certain challenge identities. The idea of controlling this abort strategy is based on the technique from [18]. On the other hand, if on a certain query, the simulator does not abort, then the technique for the actual simulation of the key-extraction oracle is similar to the technique in Section 3.1.

The challenge generation is a bit different due to the fact that in FullccHIBE level j of the HIBE has a parameter $P_{3,j}$, whereas in ccHIBE, there is one parameter P_3 for all levels of the HIBE. In case of BBG-HIBE or its augmented version

ccHIBE, the height of the target identity is fixed in the commitment stage itself. Based on this information the simulator sets up the HIBE and the effect of the committed identity tuple for BBG-HIBE or the sets of committed identities in ccHIBE is assimilated in P_3. In case of FullccHIBE there is no prior commitment stage in the reduction and the number of levels in the target identity may vary between 1 and h. This is the intuitive reason of why we need different $P_{3,i}$ for each level of the HIBE.

Theorem 2. *The* FullccHIBE *protocol is* (ϵ, t, q) *CPA-secure assuming that the* (t', ϵ', h)*-wDBDHI** *assumption holds, where*

$\epsilon \leq 2\epsilon'/\lambda$;
$t' = t + O(\sigma q) + O(\epsilon^{-2} \ln(\epsilon^{-1})\lambda^{-1} \ln(\lambda^{-1}))$; *and*
$\lambda = 1/(2(4lq2^{n/l})^h)$.

We assume $2q > 2^{n/l}$.

The proof is given in Appendix B.

5 Conclusion

In this paper, we have presented two variants of the BBG-HIBE, both of which have constant size ciphertext expansion. The first variant is proved to be secure in a generalization of the selective-ID model while the second variant is secure in the full model. We combine techniques from several papers along with the BBG-HIBE to obtain the new constructions and their proofs.

References

1. Paulo S. L. M. Barreto, Hae Yong Kim, Ben Lynn, and Michael Scott. Efficient Algorithms for Pairing-Based Cryptosystems. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer, 2002.
2. Dan Boneh and Xavier Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In Cachin and Camenisch [7], pages 223–238.
3. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In Cramer [12], pages 440–456. Full version available at Cryptology ePrint Archive; Report 2005/015.
4. Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. Earlier version appeared in the proceedings of CRYPTO 2001.
5. Dan Boneh and Jonathan Katz. Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 87–103. Springer, 2005.
6. Xavier Boyen, Qixiang Mei, and Brent Waters. Direct Chosen Ciphertext Security from Identity-Based Techniques. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 320–329. ACM, 2005.

7. Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004.
8. Ran Canetti, Shai Halevi, and Jonathan Katz. A Forward-Secure Public-Key Encryption Scheme. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer, 2003.
9. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-Ciphertext Security from Identity-Based Encryption. In Cachin and Camenisch [7], pages 207–222.
10. Sanjit Chatterjee and Palash Sarkar. Trading Time for Space: Towards an Efficient IBE Scheme with Short(er) Public Parameters in the Standard Model. In Dong Ho Won and Seungjoo Kim, editors, *ICISC*, volume 3935 of *Lecture Notes in Computer Science*, pages 424–440. Springer, 2005.
11. Sanjit Chatterjee and Palash Sarkar. Generalization of the Selective-ID Security Model for HIBE Protocols. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 241–256. Springer, 2006. Revised version available at Cryptology ePrint Archive, Report 2006/203.
12. Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005.
13. Steven D. Galbraith, Keith Harrison, and David Soldera. Implementing the Tate Pairing. In Claus Fieker and David R. Kohel, editors, *ANTS*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer, 2002.
14. Craig Gentry and Alice Silverberg. Hierarchical ID-Based Cryptography. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002.
15. Jeremy Horwitz and Ben Lynn. Toward Hierarchical Identity-Based Encryption. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer, 2002.
16. David Naccache. Secure and Practical Identity-Based Encryption. Cryptology ePrint Archive, Report 2005/369, 2005. http://eprint.iacr.org/.
17. Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.
18. Brent Waters. Efficient Identity-Based Encryption Without Random Oracles. In Cramer [12], pages 114–127.

A Proof of Theorem 1

Proof: We show that an adversary playing the $(h, n'_1, \ldots, n'_h)$-$\boldsymbol{\mathcal{M}}_2$ game against a simulator for the HIBE $(h, n_1, \ldots, n_h)$-ccHIBE can be converted into an algorithm for the h-wDBDHI* problem.

An instance of the h-wDBDHI* problem is a tuple $\langle P, Q, Y_1, \ldots, Y_h, T\rangle$ where $Y_i = \alpha^i P$ for some random $\alpha \in \mathbb{Z}_p^*$ and T is either equal to $e(P,Q)^{\alpha^{h+1}}$ or a random element of G_2.

Adversary's commitment: The adversary outputs $\mathcal{I}_1^*, \ldots, \mathcal{I}_\tau^*$ where $1 \leq \tau \leq h$ and each set $\mathcal{I}_i^*$ is a set of cardinality n_i'.

Setup: Define polynomials $F_1(x), \ldots, F_h(x)$ as follows. For $1 \leq i \leq \tau$, define

$$\begin{aligned} F_i(x) &= \prod_{\mathsf{v} \in \mathcal{I}_i^*} (x - \mathsf{v}) \\ &= x^{n_i'} + a_{i,n_i'-1} x^{n_i'-1} + \ldots + a_{i,1} x + a_{i,0}. \end{aligned}$$

For $\tau + 1 \leq i \leq h$, define $F_i(x) = x$ (and so $F_i(x) \not\equiv 0 \bmod p$ for any $x \in \mathbb{Z}_p^*$). For $1 \leq i \leq \tau$, define $a_{i,n_i'} = 1$ and $a_{i,n_i} = \cdots = a_{i,n_i'+1} = 0$; for $\tau + 1 \leq i \leq h$, set $n_i' = 1$, $a_{i,0} = 0$, $a_{i,1} = 1$ and $a_{i,2} = \cdots = a_{i,n_i} = 0$.

For $1 \leq i \leq h$, define $J_1(x), \ldots, J_h(x)$ in the following manner.

$$J_i(x) = b_{i,n_i} x^{n_i} + b_{i,n_i-1} x^{n_i-1} + \ldots + b_{i,1} x + b_{i,0}$$

where $b_{i,j}$ are random elements of $\mathbb{Z}_p$. Note that $F_i(x)$ is of degree n_i' while $J_i(x)$ is of degree n_i.

The public parameters are defined as follows. Choose a random $\beta \in \mathbb{Z}_p$.

1. $P_1 = Y_1 = \alpha P$;
2. $P_2 = Y_h + \beta P = (\alpha^h + \beta) P$;
3. $P_3 = \sum_{i=1}^{h} (b_{i,0} P + a_{i,0} Y_{h-i+1})$; and
4. for $1 \leq i \leq h$, $1 \leq j \leq n_i$,
   $Q_{i,j} = b_{i,j} P + a_{i,j} Y_{h-i+1}$;

The public parameters are $(P, P_1, P_2, P_3, \overrightarrow{Q}_1, \ldots, \overrightarrow{Q}_h)$; $\overrightarrow{Q}_i = (Q_{i,1}, \ldots, Q_{i,n_i})$. The distribution of the public parameters is as expected by the adversary. The corresponding master key $\alpha P_2 = Y_{h+1} + \beta Y_1$ is unknown to $\mathcal{B}$.

Phase 1: Suppose a key extraction query is made on $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$ for $j \leq h$. (Note that j may be less than, equal to or greater than τ.)

If $j \leq \tau$, then there must be a $k \leq j$ such that $F_k(\mathsf{v}_k) \not\equiv 0 \bmod p$, as otherwise $\mathsf{v}_i \in \mathcal{I}_i^*$ for each $i \in \{1, \ldots, j\}$ – which is not allowed by the rules of $\boldsymbol{\mathcal{M}}_2$. In case $j > \tau$, it is possible that $F_1(\mathsf{v}_1) = \cdots = F_\tau(\mathsf{v}_\tau) = 0$. Then, since $\mathsf{v}_{\tau+1} \in \mathbb{Z}_P^*$ and $F_{\tau+1}(x) = x$, we have $F_{\tau+1}(\mathsf{v}_{\tau+1}) \not\equiv 0 \bmod p$.

Thus, in all cases, there is a k such that $F_k(\mathsf{v}_k) \not\equiv 0 \bmod p$. We choose k to be the first such value in the range $\{1, \ldots, j\}$ and so for $i < k$, we have $F_i(\mathsf{v}_i) \equiv 0 \bmod p$. We next show that it is possible to construct a valid private key for v from what is known to the adversary.

Recall that $Y_i = \alpha^i P$ and hence $Y_{i_1+i_2} = \alpha^{i_1} Y_{i_2}$. Choose a random r in $\mathbb{Z}_p$ and define

$$A_1 = \beta Y_1 - \frac{1}{F_k(\mathsf{v}_k)} \left(\sum_{i=1}^{j} J_i(\mathsf{v}_i) Y_k \right) + r \left(\sum_{i=1}^{j} (F_i(\mathsf{v}_i) Y_{h-i+1} + J_i(\mathsf{v}_i) P) \right);$$

$$A_2 = -\frac{1}{F_k(\mathsf{v}_k)} \sum_{i \in \{1,\ldots,j\} \setminus \{k\}} F_i(\mathsf{v}_i) Y_{h+k-i+1};$$

$$A_3 = \sum_{i=j+1}^{h} \left(r(b_{i,0}P + a_{i,0}Y_{h-i+1}) - \frac{1}{F_k(\mathsf{v}_k)}(b_{i,0}Y_k + a_{i,0}Y_{h+k-i+1}) \right).$$

It is possible to compute A_1, A_2 and A_3 from what is known to the simulator. First note that $F_k(\mathsf{v}_k) \not\equiv 0 \bmod p$ and hence $1/F_k(\mathsf{v}_k)$ is well defined. The values $F_i(\mathsf{v}_i)$, $J_i(\mathsf{v}_i)$ and $P, Y_1, \ldots, Y_h$ are known to the simulator. Hence, A_1 and A_3 can be computed directly. In A_2, the values $Y_{h+2}, \ldots, Y_{h+k}$ are involved. However, the corresponding coefficients are $F_{k-1}(\mathsf{v}_{k-1}), \ldots, F_1(\mathsf{v}_1)$. By definition, k is the first integer in the set $\{1, \ldots, j\}$ such that $F_k(\mathsf{v}_k) \not\equiv 0 \bmod p$. Hence, $F_{k-1}(\mathsf{v}_{k-1}) \equiv \cdots \equiv F_1(\mathsf{v}_1) \equiv 0 \bmod p$ and consequently, the values $Y_{h+2}, \ldots, Y_{h+k}$ are not required by the simulator in computing A_2.

The first component d_0 of the private key d_{V} for v is obtained as $d_0 = A_1 + A_2 + A_3$. We have

$$\begin{aligned} d_0 &= A_1 + A_2 + A_3 \\ &= \pm Y_{h+1} + A_1 + A_2 + A_3 \\ &= Y_{h+1} + \beta Y_1 - \alpha^k \frac{F_k(\mathsf{v}_k)}{F_k(\mathsf{v}_k)} Y_{h-k+1} + (A_1 - \beta Y_1) + A_2 + A_3 \\ &= \alpha P_2 + \left(r - \frac{\alpha^k}{F_k(\mathsf{v}_k)} \right) A \end{aligned}$$

where

$$A = \sum_{i=1}^{j} (J_i(\mathsf{v}_i)P + F_i(\mathsf{v}_i)Y_{h-i+1}) + \sum_{i=j+1}^{h} (b_{i,0}P + a_{i,0}Y_{h-i+1}).$$

The last equality is obtained by substituting the values of A_1, A_2 and A_3 and performing a lengthy algebraic simplification. The details are a bit tedious though not too difficult and hence we omit them. Now

$$\begin{aligned} J_i(\mathsf{v}_i)P + F_i(\mathsf{v}_i)Y_{h-i+1} &= \sum_{l=1}^{n_i} b_{i,l}\mathsf{v}_i^l P + \sum_{l=1}^{n_i} a_{i,l}\mathsf{v}_i^l Y_{h-i+1} + b_{i,0}P + a_{i,0}Y_{h-i+1} \\ &= \sum_{l=1}^{n_i} \mathsf{v}_i^l Q_{i,l} + b_{i,0}P + a_{i,0}Y_{h-i+1} \\ &= V_i + b_{i,0}P + a_{i,0}Y_{h-i+1}. \end{aligned}$$

Hence,

$$\begin{aligned} A &= \sum_{i=1}^{j} V_i + \sum_{i=1}^{h} (b_{i,0}P + a_{i,0}Y_{h-i+1}) \\ &= P_3 + \sum_{i=1}^{j} V_i. \end{aligned}$$

This shows

$$d_0 = \alpha P_2 + r' \left(P_3 + \sum_{i=1}^{j} V_i \right)$$

where $\tilde{r} = r - (\alpha^k / F_k(\mathsf{v}_k))$. Since r is random, so is $\tilde{r}$ and hence d_0 is properly formed. Also,

$$d_1 = -\frac{1}{F_k(\mathsf{v}_k)} Y_k + rP = -\frac{\alpha^k}{F_k(\mathsf{v}_k)} P + rP = \tilde{r}P$$

which is as required. To form a valid private key $\tilde{r}\overrightarrow{Q}_i$ has to be computed for $j < i \leq h$. This is done as follows.

$$\begin{aligned} \tilde{r}Q_{i,l} &= \left(r - \frac{\alpha^k}{F_k(\mathsf{v}_k)} \right) (b_{i,l} P + a_{i,l} Y_{h-i+1}) \\ &= r(b_{i,l} P + a_{i,l} Y_{h-i+1}) - \frac{1}{F_k(\mathsf{v}_k)} \left(b_{i,l} Y_k + a_{i,l} Y_{h+k-i+1} \right). \end{aligned}$$

Thus, we get

$$d_{\mathsf{v}} = \left(d_0, d_1, \tilde{r}\overrightarrow{Q}_{j+1}, \ldots, \tilde{r}\overrightarrow{Q}_h \right).$$

Challenge: After completion of Phase 1, the adversary outputs two messages $M_0, M_1 \in G_2$ together with a target identity $\mathsf{v}^* = (\mathsf{v}_1^*, \ldots, \mathsf{v}_\tau^*)$ on which it wishes to be challenged. The constraint is that each $\mathsf{v}_i^* \in \mathcal{I}_i^*$ and hence $F_i(\mathsf{v}_i^*) \equiv 0 \bmod p$ for $1 \leq i \leq \tau$. If $\tau < h$, then $a_{j,0} = 0$ for $\tau < j \leq h$. The simulator picks a random $b \in \{0, 1\}$ and constructs the challenge ciphertext

$$\left(M_b \times T \times e(Y_1, \beta Q),\ Q,\ \left(\sum_{i=1}^{\tau} J_i(\mathsf{v}_i^*) + \sum_{i=\tau+1}^{h} b_{i,0} \right) Q \right).$$

Suppose, $Q = \gamma P$ for some unknown $\gamma \in \mathbb{Z}_p$. Using the fact $F_i(\mathsf{v}_i^*) \equiv 0 \bmod p$ for $1 \leq i \leq \tau$ and $a_{i,0} = 0$ for $\tau + 1 \leq i \leq h$, we have

$$\begin{aligned} \left(\sum_{i=1}^{\tau} J_i(\mathsf{v}_i^*) + \sum_{i=\tau+1}^{h} b_{i,0} \right) Q &= \gamma \left(\sum_{i=1}^{\tau} J_i(\mathsf{v}_i^*) P + F_i(\mathsf{v}_i^*) Y_{h-i+1} \right. \\ &\qquad \left. + \sum_{i=\tau+1}^{h} (a_{i,0} Y_{h-i+1} + b_{i,0} P) \right) \\ &= \gamma \left(P_3 + \sum_{i=1}^{\tau} V_i \right). \end{aligned}$$

If the input provided to the simulator is a true h-wDBDHI* tuple, i.e., $T = e(P, Q)^{(\alpha^{h+1})}$, then

$$\begin{aligned} T \times e(Y_1, \beta Q) &= e(P,Q)^{(\alpha^{h+1})} \times e(Y_1, \beta Q) \\ &= e(Y_h, Q)^{\alpha} \times e(\beta P, Q)^{\alpha} \\ &= e(Y_h + \beta P, Q)^{\alpha} \\ &= e(P_2, \gamma P)^{\alpha} \\ &= e(P_1, P_2)^{\gamma}. \end{aligned}$$

So, the challenge ciphertext

$$\left(M_b \times e(P_1, P_2)^{\gamma}, \gamma P, \gamma \left(\sum_{j=1}^{\tau} V_j + P_3 \right) \right)$$

is a valid encryption of M_b under $\mathsf{v}^* = (\mathsf{v}_1^*, \ldots, \mathsf{v}_\tau^*)$. On the other hand, when T is random, the first component of the challenge ciphertext is a random element of G_2 and provides no information to the adversary.

Phase 2: This is similar to Phase 1.

Guess: Finally, the adversary outputs its guess $b' \in \{0,1\}$. The simulator outputs $1 \oplus b \oplus b'$.

This gives us the required bound on the advantage of the adversary in breaking the HIBE protocol. □

B Proof of Theorem 2

Proof: Suppose $\mathcal{A}$ is a (t,q)-CPA adversary for the h-HIBE, then we construct an algorithm $\mathcal{B}$ that solves the h-wDBDHI* problem. $\mathcal{B}$ takes as input a tuple $\langle P, Q, Y_1, \ldots, Y_h, T \rangle$ where $Y_i = \alpha^i P$ for some random $\alpha \in \mathbb{Z}_p^*$ and T is either equal to $e(P,Q)^{\alpha^{h+1}}$ or is a random element of G_2. We define the following game between $\mathcal{B}$ and $\mathcal{A}$.

Setup: $\mathcal{B}$ chooses random $u_1, \ldots, u_h \in \mathbb{Z}_m$ and l-length vectors $\overrightarrow{x_1}, \ldots, \overrightarrow{x_h}$ with entries from $\mathbb{Z}_m$. Here $m = 2\max(2q, 2^{n/l}) = 4q$. Similarly, it chooses random $v_1, \ldots, v_h \in \mathbb{Z}_p$ and l-length vectors $\overrightarrow{y_1}, \ldots, \overrightarrow{y_h}$ from $\mathbb{Z}_p$. It further chooses k_j for $1 \le j \le h$ randomly from $\{0, \ldots, \mu_l\}$, where $\mu_l = l(N^{1/l} - 1)\}$. Let, $\mathsf{v}_j = (\mathsf{v}_{j,1}, \ldots, \mathsf{v}_{j,l})$. For $1 \le j \le h$, it then defines the functions:

$$F_j(\mathsf{v}_j) = p + mk_j - u_j - \sum_{i=1}^{l} x_{j,i} \mathsf{v}_{j,i}$$

$$J_j(\mathsf{v}_j) = v_j + \sum_{i=1}^{l} y_{j,i} \mathsf{v}_{j,i}$$

$$K_j(\mathsf{v}_j) = \begin{cases} 0 \text{ if } u_j + \sum_{i=1}^{l} x_{j,i} \mathsf{v}_{j,i} \equiv 0 \bmod m \\ 1 \text{ otherwise} \end{cases}$$

These functions originate in the work of Waters [18] and have later been used in [10]. The current definitions extend the definitions given in [10] for IBE to the case of HIBE. These functions are used to control the abort strategy by the simulator.

Next, $\mathcal{B}$ assigns $P_1 = Y_1,\ P_2 = Y_h + yP$, $P_{3,j} = (p + mk_j - u_j)Y_{h-j+1} + v_jP$ for $1 \leq j \leq h$ and $U_{j,i} = -x_{j,i}Y_{h-j+1} + y_{j,i}P$ for $1 \leq j \leq h$ and $1 \leq i \leq l$. It provides $\mathcal{A}$ the public parameters $\langle P, P_1, P_2, \overrightarrow{P}_3, \overrightarrow{U}_1, \ldots, \overrightarrow{U}_h \rangle$. Everything else is internal to $\mathcal{B}$. 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. The master secret αP_2 is unknown to $\mathcal{B}$.

Using the definition of the public parameters it is possible to show that

$$V_j = P_{3,j} + \sum_{i=1}^{l} \mathsf{v}_{j,i} U_{j,i} = F_j(\mathsf{v}_j) Y_{h-j+1} + J_j(\mathsf{v}_j) P.$$

As in the proof of Theorem 1, this fact is crucial to the answering key-extraction queries and challenge generation.

Phase 1: Suppose $\mathcal{A}$ asks for the private key corresponding to an identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_u)$, for $u \leq h$. $\mathcal{B}$ first checks whether there exists a $j \in \{1, \ldots, u\}$ such that $K_j(\mathsf{v}_j) \neq 0$. It aborts and outputs a random bit if there is no such j. Otherwise, it answers the query in a manner similar to that in the proof of Theorem 1.

$\mathcal{B}$ chooses r randomly from $\mathbb{Z}_p$ and computes

$$d_{0|j} = -\frac{J_j(\mathsf{v}_j)}{F_j(\mathsf{v}_j)} Y_j + yY_1 + r(F_j(\mathsf{v}_j) Y_{h-j+1} + J_j(\mathsf{v}_j) P);$$
$$d_1 = \frac{-1}{F_j(\mathsf{v}_j)} Y_j + rP.$$

It is standard to show that $d_{0|j} = \alpha P_2 + \tilde{r} V_j$ and $d_1 = \tilde{r} P$, where $\tilde{r} = r - \frac{\alpha^j}{F(I_j, k_j)}$. As in the proof of Theorem 1, it is possible to show that $\mathcal{B}$ can compute $\tilde{r} V_i$ for any $i \in \{1, \ldots, u\} \setminus \{j\}$; and $\tilde{r} P_{3,k}$, $\tilde{r} \overrightarrow{U_k}$ for $u < k \leq h$. The simulator computes $d_0 = d_{0|j} + \sum_{i \in \{1,\ldots,u\} \setminus \{j\}} \tilde{r} V_i$. $\mathcal{A}$ is provided the private key corresponding to v as $d_{\mathsf{v}} = \left(d_0, d_1, \tilde{r} P_{3,u+1}, \ldots, \tilde{r} P_{3,h}, \tilde{r} \overrightarrow{U}_{u+1}, \ldots, \tilde{r} \overrightarrow{U}_h \right)$. Note that d_{v} is a valid private key for v following the proper distribution. $\mathcal{B}$ will be able to generate this d_{v} as long as there is a $j \in \{1, \ldots, u\}$ such that $F_j(\mathsf{v}_j) \not\equiv 0$ for which it suffices to have $K_j(\mathsf{v}_j) \neq 0$.

Challenge: $\mathcal{A}$ submits two messages $M_0, M_1 \in G_2$ together with a challenge identity $\mathsf{v}^* = (\mathsf{v}_1^*, \ldots, \mathsf{v}_\tau^*)$, $\tau \leq h$ on which it wants to be challenged. $\mathcal{B}$ aborts and outputs a random bit, if $F_j(\mathsf{v}_j^*) \not\equiv 0$ for any $j \in \{1, \ldots, \tau\}$. Otherwise, $\mathcal{B}$ chooses a random bit $\gamma \in \{0, 1\}$ and gives $\mathcal{A}$ the tuple

$$\mathsf{CT} = \left(T \times e(Y_1, yQ) \times M_\gamma, Q, \sum_{j=1}^{\tau} J_j(\mathsf{v}_j^*) Q \right).$$

If $\langle P, Q, Y_1, \ldots, Y_h, T\rangle$ given to $\mathcal{B}$ is a valid h-wDBDHI* tuple, i.e., $T = e(P,Q)^{\alpha^{h+1}}$ then CT is a valid encryption for M_γ. Suppose $Q = cP$ for some unknown $c \in \mathbb{Z}_p$. Then the first component of CT can be seen to be $e(P_1, P_2)^c$. Further, using $F_j(\mathsf{v}_j^*) \equiv 0 \bmod p$ it can be shown that $J_j(\mathsf{v}_j^*)Q = cV_j$. The correctness of the third component of CT follows from this fact. If T is a random element of G_2, CT gives no information about $\mathcal{B}$'s choice of γ.

Phase 2: Similar to Phase 1, with the restriction that $\mathcal{A}$ cannot ask for the private key of v^* or any of its ancestors.

Guess: $\mathcal{A}$ outputs a guess γ' of γ.

A lower bound λ on the probability of aborting upto this stage is the following.

$$\lambda = \frac{1}{2(4lq2^{n/l})^h}.$$

Waters [18] obtains a similar bound (for the case $l = n$) in the context of an IBE secure in the full model under the DBDH assumption. In the same paper, Waters had suggested a construction for HIBE where new public parameters are generated for each level of the HIBE. Generating new public parameters for each level of the HIBE simplifies the probability analysis for the lower bound on the probability of abort.

The security of FullccHIBE is based on the h-wDBDHI* problem which is obtained by modifying the BBG-HIBE. For each level of the HIBE we have separate public parameters $P_{3,i}$ and $(U_{i,1}, \ldots, U_{i,l})$. This makes it possible to easily apply the reasoning of Waters leading to the above mentioned lower bound.

At this point, we have to also use the technique of "artificial abort" employed by Waters [18]. The idea is that the probability of aborting upto this is not independent of the adversarial queries. The idea of the artificial abort technique is to allow the simulator to sample the transcript of queries it obtained from the adversary and on certain conditions abort and output a random bit. This increases the total probability of abort and makes it almost equal for all adversarial inputs. This helps in the probability analysis. The effect of sampling the transcript is to increase the runtime of the simulator. See [18] for the details. Incorporating the probability analysis of Waters into the present situation in a straightforward manner we obtain the required result. □

A New Proxy Signature Scheme Providing Self-delegation

Younho Lee[1], Heeyoul Kim[1], Yongsu Park[2], and Hyunsoo Yoon[1]

[1] Network and Security Laboratory
Division of Computer Science
Korea Advanced Institute of Science and Technology(KAIST)
{yhlee, hykim, hyoon}@nslab.kaist.ac.kr
[2] College of Information and Communication, Hanyang University
yspark@hanyang.ac.kr

Abstract. We improve Malkin et al's construction (Eurocrypt'04) of the proxy signature scheme in the random oracle model[1]. Unlike Malkin et al's scheme, the proposed scheme does not assume the existence of the trusted secure device, which has a global secret key that all users' private keys can be recovered with. This makes the proposed scheme more scalable and efficient because users need not access and cooperate with the secure device to generate their public/private key pairs.

We show that the proposed scheme is provably secure based on the modified strong RSA assumption that was made by the Itkis et al (CRYPTO'01).

Keywords: cryptographic protocol, digital signature, proxy signature, self-delegation.

1 Introduction

Nowadays, numerous internet services, such as internet banking, home trading, on-line payments, electronic commercial services, and others, rely on PKI to authenticate each user.

Due to the ease of key management, it is frequent that an individual user uses only one certified private/public key pair and its corresponding certificate; they tend to be used to all on-line services. However, this causes a potential security problem: first, the probability of the private key exposure is increased. For instance, if the private key is used on the insecure computing environment such as a public PC or a friend's PDA, this gives a malicious program a chance to plunder the private key by searching the memory where the private key is stored, or by intercepting the password for decrypting the enciphered private

[1] 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 MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment).

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 328–342, 2006.
© Springer-Verlag Berlin Heidelberg 2006

key. Moreover, many other on-line services, which rely on this plundered private key for authentication, can be compromised.

It can be a solution for the above problem to use a temporal secondary private key on behalf of the primary private key. Since the right of the temporal secondary key can be limited only to a specific purpose and its validity period can be limited, the damage of the key exposure is reduced.

To accomplish the right delegation from the primary private key to the secondary private key, it can be considered to employ the delegation-by-certificate method [5,13]. In general, the delegation-by-certificate is useful to delegate an entity's signing right to another entity. In delegation-by-certificate, a user, called as a right grantor, delegates a restricted amount of authority of his/her private key to that of another user, called as a right grantee, by signing a delegation certificate. The delegation certificate is attached to the right grantee's signature when the right grantee signs a document on behalf of the right grantor.

Unfortunately, the delegation-by-certificate is not appropriate to authorize the secondary private key. As has been stated in [7], because the delegated right from the primary private key to the secondary is a personalized right, the propagation of the delegated right should be limited. For example, suppose that a user who has a right to access the on-line music service and this right is only bound to the user. If the user generates many secondary private keys and their appropriate certificates, and sends them to other users, e. g. his/her friends or families; they can access the on-line music service because their identities cannot be revealed from the secondary private keys.

Thus, the self-delegation concept has been proposed to provide a way to limit the propagation of the delegated right as well as to reduce the damage caused by the key exposure [7]. In self-delegation, a secondary private key which has a special attribute is used to prevent the unlimited propagation of the delegated right.

In 2003, Boldyreva et al. have brought the self-delegation concept into the proxy signature and have proposed a proxy signature scheme providing the self-delegation facility (PSsd) [1]. Later, Malkin et al. [10] found the security flaws of the Boldyreva et al's scheme and have proposed a new PSsd scheme based on the key insulated signature scheme (KI) [3,4][2].

However, because Malkin et al. designed the PSsd scheme to show a theoretical equivalence between PSsd and KI, it has several drawbacks to be used practically. The first problem is that each user can't generate his/her private/public key pair without the help of the central trusted secure device, which acts as a key generation center. In Malkin et al's scheme, there exists a global secret key[3].

[2] This is a part of their contributions. The main contribution of their work is to define a new secure hierarchical model of the proxy signature and to show the relations between the proxy signature and the key-evolving signatures.

[3] In Theorem 2 in [11], the master key and the first phase signing key of KI are used for all users' private key generation. Only with these two information, all users' private key can be recovered. Thus, users should not access these information explicitly. This is the reason of our opinion that the construction in [11] should use the trusted device.

Since the trusted secure device should participate in the key generation of each user with this global secret key, the scalability of the scheme is severely degraded. Besides, the existence of the trusted secure device causes the second problem that this trusted secure device can be a critical target of adversary: unlike the certificate authority, this trusted secure device has a global secret key with which all users' private keys can be recovered.

We address these problems and construct a new proxy signature scheme providing self-delegation. In the proposed scheme, there does not exist a global secret key to generate the private/public key of each user. This removes a single critical target of the adversary. Besides, all users can generate their key pair independently. Thus, the proposed scheme is more scalable than the previous one.

The algorithms in the proposed scheme are based on those of the forward secure signature scheme by G. Itkis and L. Reyzin [9]. We define the security of the proposed scheme and show the security of the proposed scheme based on the modified version of the strong RSA assumption [9] using the random oracle model [2].

The rest of this paper has the following organization. In Section 2, the preliminaries are provided. In Section 3, we provide the proposed PSsd scheme. We provide the security and performance analysis of the proposed scheme in Section 4 and Section 5 respectively. In Section 6, we discuss the relation between Malkin et al's work [10,11] and the proposed scheme.

2 Preliminary

In this section, the preliminaries are provided. The first subsection gives an overview of the proposed PSsd scheme. The second subsection provides the security requirements of the PSsd scheme.

2.1 Overview of the Proposed PSsd Scheme

An overview of the proposed PSsd scheme is informally described. We first explain the participants of the PSsd scheme. There are 4 types of participants:

- **Users:** They own devices, many private keys, and corresponding only one public key respectively. They store their private keys in their main device initially. The private keys can be classified as a signing key and delegation key, whose descriptions are shown below.
- **Right grantor devices** (OD)**:** They delegate their signing right to ED. They sends ED a proxy for ED to exercise the delegated signing right using them. In self-delegation case, the proxy contains the signed certificate and a number of private keys for ED. On the other hand, the proxy only includes the signed certificate in non-self-delegation case.

– **Right grantee devices** (ED)**:** They receive a proxy from OD and proxy-sign a message on behalf of OD.
– **Verifiers:** The ones who verify the proxy signature. It is assumed that they have the authentic public key which corresponds to the proxy signature to be verified.

Next, we classify the private keys according to their usage.

– **Signing key:** This is to be used for a device to sign a message. The proxy signing key is included in this type.
– **Delegation key:** OD keeps the delegation keys until OD sends it to ED during the self-delegation. ED uses one of the delegation keys as a proxy signing key.

The proposed PSsd scheme has five algorithms. The description of each algorithm is given below.

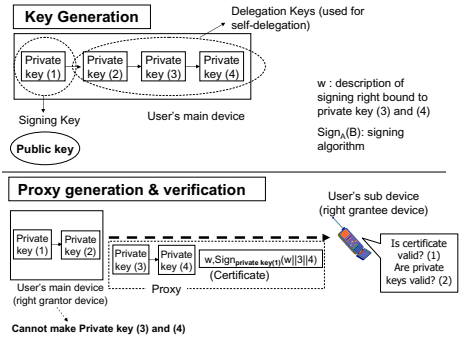


Fig. 1. Overview of key generation, proxy generation, and verification algorithms

• **Key generation:** This algorithm is used to generate an individual user's private keys and the corresponding public key. All private keys should be distinguishable. The upside of Figure 1 shows an example of the result of key generation algorithm. After the key generation, the public key is published.

• **Proxy generation:** To delegate a signing right, OD generates a proxy to send it to ED. This algorithm includes both self-delegation and non-self-delegation case. For non-self-delegation, OD simply generates a signed certificate which certifies the delegated signing right, and sends it, which is the proxy in this case, to ED[4]. On the other hand, to perform the self-delegation, OD sends not only the certificate but also one or some of delegation keys for ED to exercise the delegated signing right with them. In this case, the proxy contains the delegation

[4] It is exactly the same as the delegation-by-certificate case [1,12]. In non-self-delegation case, the delegation propagation problem is not happened since the identity of ED's owner is revealed by the public key.

keys as well as the certificate. After a self-delegation, OD removes the delegation keys that are sent so that they cannot be recovered even if OD is compromised. The downside of Figure 1 shows the self-delegation case.

• **Proxy verification:** After receiving the proxy, ED verifies the certificate in the proxy with the public key of the OD. If the certificate is shown to be invalid, ED cannot exercise the delegated signing right. In self-delegation case, the delegation keys are included in the proxy. These delegation keys are also verified. After verification, one of them is used as a proxy signing key and the others are remained to the delegation keys for ED's later delegation.[5] In non-self-delegation case, the proxy signing key is the signing key of ED. The delegated signing right is controlled by the description of signing right in the certificate.

• **Proxy signature generation:** This algorithm is used to generate a proxy signature with ED's proxy signing key. The proxy signature includes the certificates that are received from OD as a part of the proxy.

• **Proxy signature verification:**The verifier runs this algorithm to verify the proxy signature is valid. The verifier should have the public keys of all users which own all the devices that participate in generating the proxy signature, for example, after a device D_1, which is owned by a user U_1, delegates a signing right to D_2 which also belongs to U_1, D_2 can also delegate its delegated signing right to another device D_3 which belongs to another user U_2. In this case, the verifier have both U_1's and U_2's public key to verify the D_3's proxy signature.

2.2 Security Requirements of the Proposed PSsd Scheme

The security requirements of PSsd are as follows. These requirements include that of the ordinary proxy signature by Mambo et al [12]. This is informal description but it may help understanding why the formal security model at Section 4 is suggested.

- *Verifiability*: A verifier can be convinced of the OD's agreement by verifying the proxy signature.
- *Unforgeability*: Only the designated ED can generate a valid proxy signature.
- *Non-repudiation*: ED cannot repudiate a valid proxy signature generated by itself.
- *Limited Propagation*: It should be provided how to limit the propagation of the self-delegation.
- *Forward Security*: After the self-delegation, the proxy signing key of the ED should be difficult to derive from the private key of OD.
- *Backward Security*: It should be difficult to derive the private key of OD from the delegated proxy signing key of ED.

[5] At later self-delegation, the current ED will act as OD.

3 Proposed Scheme

In this section, the proposed PSsd scheme is provided. We extends the key generation technique of the Itkis et al's forward secure signature scheme [9] to provide the self-delegation facility. The description of each algorithm is given below. In the description, $a \xleftarrow{R} A$ means that a is randomly selected element from a set A.

Key generation:

Input: $id \in N$(user identification information(ID)), $T \in N$(number of delegation keys), $k \in N$(bit length of n_{id}), l (bit length of $e_{id,i}(i = 0, .., T)$).

Output: $SK_{id,0}$ (signing key), $SK_{id,1}, ..., SK_{id,T}$ (delegation keys), PK_{id} (public key).

Procedure:

1. Generate random $(\lceil k/2 \rceil - 1)$ bit primes q_1, q_2 s.t. $p_i = 2q_i + 1 (i = 1, 2)$ are both prime.
2. $n_{id} \leftarrow p_1 p_2$, $t_{id,0} \xleftarrow{R} Z^*_{n_{id}}$.
3. Generate primes $e_{id,i}$ s.t. $2^l(1+i/(T+1)) \leq e_{id,i} \leq 2^l(1+(i+1)/(T+1))(i = 0, 1, 2, .., T)$[6].
4. $f_0 \leftarrow e_{id,1} \cdot ... \cdot e_{id,T} \bmod \phi(n_{id})$ *where* $\phi(n_{id}) = 4q_1q_2$.
5. $s_{id,0} \leftarrow t_{id,0}^{f_0} \bmod n_{id}$, $V_{id} \leftarrow 1/s_{id,0}^{e_{id,0}} \bmod n_{id}$, $t_{id,1} \leftarrow t_{id,0}^{e_{id,0}} \bmod n_{id}$.
6. $PK_{id} \leftarrow (n_{id}, V_{id}, T)$, $SK_{id,0} \leftarrow (0, T, n_{id}, s_{id,0}, e_{id,0})$.
7. For $i = 1, ..., T$ do
   (a) $f_i \leftarrow f_{i-1}/e_{id,i} \bmod \phi(n_{id})$, $s_{id,i} \leftarrow t_{id,i}^{f_i} \bmod n_{id}$.
   (b) $SK_{id,i} \leftarrow (i, T, n_{id}, s_{id,i}, e_{id,i})$.
   (c) If $(i < T)$ do $t_{id,i+1} \leftarrow t_{id,i}^{e_{id,i}} \bmod n_{id}$.
8. Erase $t_{id,i}$ $(i = 1, ..., T)$, p_1, p_2 and Return $(PK_{id}, (SK_{id,1}, ..., SK_{id,T}))$.
9. Exit.

Proxy generation: The description of this algorithm assumes that current OD, which runs this algorithm, delegates all or a part of the signing right which has already been transferred from another device. Thus, the current OD has a chain of certificates which represent the multi-level delegation. In the description, W represents the set of these certificates in the chain. Moreover, for the general description of this algorithm, it is assumed that the current OD has a set of private keys $SK_{i,l_x}, ..., SK_{i,l_y} (0 \leq l_x \leq l_y \leq T)$. Among them, SK_{i,l_x} is used as a signing key and the others are used as delegation keys.

This algorithm gets a key limitation range l_1, l_2 $(l_x < l_1 \leq l_2 \leq l_y)$ as an input only when it executes the self-delegation. This means that $SK_{i,l_1}, ..., SK_{i,l_2}$ are

[6] Refer to Itkis et al's work [9] for detail algorithm.

included in the proxy. ED will use SK_{i,l_1} as the proxy signing key and the other keys as the delegation keys.

The notation $H(\cdot,\cdot)$ is regarded as a secure hash function where $H : \{0,1\}^* \times Z^*_{n_{id}} \rightarrow \{0,1\}^l$ $(2^{l+1} < n_{id})$. The detailed description of this algorithm is as follows.

Input: i (ID of the OD's owner), j (ID of the ED's owner), $SK_{i,l_x}, ..., SK_{i,l_y} (0 \leq l_x \leq l_y \leq T)$ (private keys), $m_{w_{new}} \in \{0,1\}^*$ (description of the delegated signing right), $l_1, l_2 (l_x < l_1 \leq l_2 < l_y)$ (key limitation range, only gets as an input in the self-delegation), W (set of certificates).

Output: $Proxy = (W'$ (new set of certificates), $(SK_{i,l_1}, ..., SK_{i,l_2})$ (only when self-delegation)).

Procedure:

1. If $(W \neq \phi)$ do /* get the last certificate on the certificate chain in W */
   - If $(id_{last} \neq i)$ do $(m_{w_{last}}, l_{last}, id_{last}, \sigma_{last}, c_{last}, e_{id_{last},l_{last}}) \leftarrow W$.
   else $((m_{w_{last}}, l_x, l_y), l_{last}, id_{last}, \sigma_{last}, c_{last}, e_{i,l_{last}}) \leftarrow W$.
   else $\sigma_{last} \leftarrow 1$.
2. $(l_x, T, n_i, s_{i,l_x}, t_{i,l_x}, e_{i,l_x}) \leftarrow SK_{i,l_x}$, $r_{new} \overset{R}{\leftarrow} Z^*_n$, $d_{new} \leftarrow r_{new}^{\sigma_{last}} \bmod n_i$, $\sigma_{new} \leftarrow H(m_{w_{new}}, d_{new}^{e_{i,l_x}})$, $c_{new} \leftarrow r_{new} \cdot s_{i,l_x}^{\sigma_{new}} \bmod n_i$.
3. If $(i \neq j)$ do /* non-self-delegation */
   $W' \leftarrow (W, (m_{w_{new}}, l_x, i, \sigma_{new}, c_{new}, e_{i,l_x}))$, Return W', Exit.
4. If $(i = j)$ do /* self-delegation */
   $W' \leftarrow (W, ((m_{w_{new}}, l_1, l_2), l_x, i, \sigma_{new}, c_{new}, e_{i,l_x}))$, Return $(W', (SK_{i,l_1}, ..., SK_{i,l_2}))$, Erase $SK_{i,l_1}, ..., SK_{i,l_2}$.
5. Exit.

Proxy verification: ED which receives the proxy from OD verifies the proxy with this algorithm. For a general description, the description of this algorithm assumes a multi-level delegation case. This algorithm contains two sub-algorithms. The description of each sub-algorithm is independently given after that of the proxy verification algorithm.

Input: $Proxy = (W)$ (non-self-delegation case) or $(W, (SK_{i,l_1}, ..., SK_{i,l_2}))$ (self-delegation case), $PKS = \{PK_{id} | (\cdot, \cdot, id, \cdot, \cdot) \in W\}$ (a set of public keys to verify the certificates in W), i (ID of OD's owner).

Output: true (proxy verification success) or false(proxy verification failure).

Procedure:

1. $W \leftarrow Proxy$.
2. If $(\mathsf{CertVerify}(W, i, PKS) = \text{false})$ do Return false.
3. If $((SK_{i,l_1}, ..., SK_{i,l_2}) \in W)$ do /* Current delegation is self-delegation */
   If $(\mathsf{DelegationKeyVerify}((SK_{i,l_1}, ..., SK_{i,l_2}), PK_i) = \text{false})$ do Return false.
4. Return true.

The CertVerify and DelegationKeyVerify algorithms are given below respectively. The CertVerify is also used in the proxy signature verification algorithm.

CertVerify(W, i, PKS) (W: a set of certificates which form a certificate chain, i: ID of the OD's owner, PKS: a set of public keys to verify the certificates in W)

1. If $(W = \phi)$ do Return true, Exit.
2. $(W, (M_{last}, l_{last}, id_{last}, \sigma_{last}, c_{last}, e_{id_{last},l_{last}})) \leftarrow W$.
3. If $id_{last} \neq i$ do Return false, Exit.
4. $W \leftarrow (W, (M_{last}, l_{last}, id_{last}, \sigma_{last}, c_{last}, e_{id_{last},l_{last}}))$.
5. While (W has two or more certificates) do
   (a) $(W, (M', l', id', \sigma', c', e_{id',l'}), (M, l, id, \sigma, c, e_{id,l})) \leftarrow W$.
   /* get last two certificates in the certificate chain from W */
   (b) If $(id' = id)$ do /* self-delegation case */
   $(m'_w, l'_1, l'_2) \leftarrow M'$.
   If $(M = m_w)$ do if $l'_1 \neq l$ do Return false, Exit.
   else $(m_w, l_1, l_2) \leftarrow M$,
   If not $(l'_1 < l_1 \leq l_2 \leq l'_2)$ do Return false, Exit.
   else $m'_w \leftarrow M'$, $m_w \leftarrow M$, /* non self-delegation case */
   If (m_w violates m'_w[7]) do Return false, Exit.
   (c) $PK_{id} \leftarrow PKS$, $(\cdot, V_{id}, \cdot) \leftarrow PK_{id}$.
   (d) if $(\sigma = H(m_w, V_{id}^{\sigma\sigma'} c^{e_{id,l}\sigma'}))$ do $W \leftarrow (W, (M', l', id', \sigma', c', e_{id',l'}))$.
   else Return false, Exit.
6. If (W has only one certificate) do
   (a) $((M, l', id, \sigma, c, e_{id,l'})) \leftarrow W$.
   (b) $m_w \leftarrow M$.
   (c) If $(\sigma = H(m_w, V_{id}^{\sigma} c^{e_{id,l'}}))$ do $W \leftarrow null$ else Return false, Exit.
7. Return true, Exit.

DelegationKeyVerify$((SK_{i,l_1}, ..., SK_{i,l_2}), PK_i)$ (where $(SK_{i,l_1}, ..., SK_{i,l_2})$ are the delegation keys to be verified and PK_i is their corresponding public key.)

1. $(\cdot, V_i, \cdot) \leftarrow PK_i$.
2. For $j = l_1$ to l_2 do
   (a) $(j, T, n_i, s_{i,j}, t_{i,j}, e_{i,j}) \leftarrow SK_{i,j}$.
   (b) If $(V_i \cdot s_{i,j}^{e_{i,j}} \not\equiv 1 \pmod{n_i})$ do Return false, Exit.
3. Return true, Exit.

Proxy signature generation: ED runs this algorithm to generate a proxy signature. This algorithm is a modified version of the GQ signature scheme [8].

Input: i (ID of ED's owner), $m \in \{0,1\}^*$ (message to be signed), SK_{i,l_x} (proxy signing key), $W(= (..., (\cdot, l_{last}, id_{last}, \sigma_{last}, c_{last}))$ (included in the proxy).

[7] Its decision is dependent on that of the application where this right delegation ability is used. For more information, please refer to the discussion in [10].

Output: $\mathsf{ProxySig}(m, SK_{l_x}) = (W, (m, l_x, i, \sigma, c))$ (Proxy signature).

Procedure:

1. If $(W \neq \phi)$ do $(\cdot, l_{last}, id_{last}, \sigma_{last}, c_{last}) \leftarrow W$.
   /* get last warrant information from W */
   else $\sigma_{last} \leftarrow 1$.
2. $(l_x, T, n_i, s_{i,l_x}, t_{i,l_x}, e_{i,l_x}) \leftarrow SK_{i,l_x}$, $r \stackrel{R}{\leftarrow} Z_n^*$, $d \leftarrow r^{\sigma_{last}} \bmod n_i$.
   $\sigma \leftarrow H(m, d^{e_{i,l_x}})$, $c \leftarrow r s_{i,l_x}^{\sigma} \bmod n_i$.
3. Return $(W, (m, l_x, i, \sigma, c))$, Exit.

Proxy signature verification:

Input: $(W, (m, l_x, i, \sigma, c, e_{i,l_x}))$ (proxy signature), PK_i (public key of ED's owner), $PKS = \{PK_{id} | (\cdot, \cdot, id, \cdot, \cdot) \in W\}$ (Public keys to verify the certificates in W).

Output: true (proxy signature is valid) or false (proxy signature is invalid)

Procedure:

1. $A \leftarrow \mathsf{CertVerify}(W, i, PKS)$.
2. If $A =$ false do Return false, Exit.
3. If $W \neq \phi$ do $(W, (\cdot, l_{last}, id_{last}, \sigma_{last}, c_{last}) \leftarrow W)$ else $\sigma_{last} \leftarrow 1$.
4. $(\cdot, V_i, \cdot) \leftarrow PK_i$.
5. If $\sigma = H(m, V_i^{\sigma_{last}\sigma} c^{e_{i,l_x}\sigma_{last}})$ do Return true else Return false.
6. Exit.

4 Security Analysis

Based on the security requirements described in Section 2, we build the security model for the proposed proxy signature scheme and analyze whether the proposed scheme meets them respectively with the proposed definition. We only deal with the self-delegation case because the proposed scheme exactly follows the delegation-by-certificate for non-self-delegation case: the delegation-by-certificate has been known to be secure with the appropriate setting of the description of the signing right [1,10]. This is a reasonable assumption because the users in the proposed scheme generates their own security parameters independently[8].

4.1 Definition of Security in the Proposed PSsd Scheme

We define the experiment **F-PSsd** first. In addition, the definition of the security in the PSsd scheme is provided with the experiment **F-PSsd**.

[8] In Malkin et al's work [10,11] all users' keys are NOT independently generated. In the description of $\mathsf{Gen_{PS}}(1^k)$ at the proof of theorem 2 in [11] shows that all users generate their private keys with one instance of $\mathsf{Gen_{KI}}(1^k, c \cdot n)$.

• **Experiment F-PSsd**

1. Generate the public key $PK_{id,0} = \{n_{id}, V_{id}, T_{id}\}$ and the private keys $SK_{id,0}, ..., SK_{id,T}$. In addition, the number of devices $b \in N$ is selected including the main device that has all private keys at initial step. Let these selected devices be named as $0, 1, ..., b$.
2. The self-delegation is executed: the main device (0) self-delegates the signing right to a device 1 by sending the delegation keys and their appropriate certificates, 1 self-delegates the delegated signing right to 2, and repeat it until $b-1$ self-delegates its signing right to b. Finally, the devices $0, ..., b$ make a delegation chain.
3. Among the participant devices, the adversary tries to select a specified device j ($\in \{0, ..., b\}$) as a target device to attack. After the selection of the target device, the adversary can acquire all information of the other devices including their signing key and delegation keys.
4. With the gathered information of the previous step, the adversary runs an algorithm B. B can query the proxy signature for any message $m \in \{0,1\}^*$ to device j (as signature oracle) and request to delegate a part of j's signing right to another devices (as delegation oracle). This execution can be repeated until B wants to terminate this step.
5. If possible, B outputs a forgery result, which can be either the forgery of the proxy signature or that of the proxy.

If the adversary in **F-PSsd** fails, the proposed scheme meets the five security requirements of Section 2.2 owing to the following reason:

- *Verifiability*: If the adversary does not win, the adversary cannot make a valid proxy signature of the target device j. This includes that the certificates which are contained in the proxy signature cannot be forged. Thus, the proposed scheme meets this requirement.
- *Unforgeability*: Since the failure of B means that the proxy signature cannot be forged, this requirement is preserved.
- *Non-repudiation*: Since the signing key is assigned before the start of the adversary's attack and the proxy signature and the proxy cannot be forged, the proposed scheme meets the requirement.
- *Forward Security and Backward Security*: As B fails even if the signing keys of all devices except j are revealed to B, these requirements are preserved.

Aside from the experiment model, the proposed scheme meets the *Limited Propagation* requirement because the number of times that a device can execute the self-delegation is limited to the number of delegation keys in the proposed scheme.

Let the probability that the experiment **F-PSsd** outputs a valid forgery result be $\mathbf{Succ}^{PSsd}(PSsd[k,l,T], \mathsf{B})$. We define that the insecurity function $\mathbf{InSec}^{PSsd}(PSsd[k,l,T], t, q_{sig})$ is the maximum value of $\mathbf{Succ}^{PSsd}(PSsd[k,l,T], \mathsf{B})$ when the number of the signature and delegation queries are limited to q_{sig}.

Based on the definition of $\mathbf{InSec}^{PSsd}(PSsd[k,l,T],t,q_{sig})$, the security of the PSsd scheme is defined as follow.

Definition 1 (Security of the Pssd scheme). *If* $\mathbf{InSec}^{PSsd}(PSsd[k,l,T],t,q_{sig})$ *for a* Pssd *scheme is negligible, then the* Pssd *scheme is secure.*

4.2 The Proofs

To show the proposed PSsd scheme is secure, we bring in the modified strong RSA assumption which has been suggested by Itkis et al's work [9]: Given $n = p_1p_2$ such that $p_1 = 2q_1+1$ and $p_2 = 2q_2+1$ where q_1 and q_2 are $(\lceil k/2 \rceil - 1)$ bit prime numbers, let $\mathbf{Succ}^{SRSA}([k,l,T],\mathsf{A})$ be the probability that A can find a (β, r) pair for a random $\alpha \in Z_n^*$ where $\beta^r \equiv \alpha \bmod n$ (where $1 < r \leq 2^{l+1}$). Then, the modified strong RSA assumption is that if $\mathbf{InSec}^{SRSA}(k,l,t)$ is the maximum probability of $\mathbf{Succ}^{SRSA}([k,l,T],\mathsf{A})$, $\mathbf{InSec}^{SRSA}(k,l,t)$ is negligible. (For more detail, see Section 2.2 in [9].) Our proofs are based on the random oracle model [2].

We first show that $\mathbf{Succ}^{SRSA}([k,l,T],\mathsf{A}) \geq \epsilon'$ if $\mathbf{Succ}^{PSsd}(PSsd[k,l,T],t,q_{sig}) \geq \epsilon$. Following this, by the antithesis, we show that if $\mathbf{Succ}^{SRSA}([k,l,T],\mathsf{A}) \leq \epsilon'$, which is the modified strong RSA assumption, then $\mathbf{Succ}^{PSsd}(PSsd[k,l,T],t,q_{sig}) \leq \epsilon$.

Theorem 1. *Given a forger* **F** *for the proposed* PSsd *scheme that runs in time at most* t*, asking* q_{hash} *queries and* q_{sig} *signing queries, such that* $\mathbf{Succ}^{PSsd}(PSsd[k,l,T],\mathsf{B}) \geq \epsilon$*, an algorithm* A *can be constructed such that* $\mathbf{Succ}^{SRSA}([k,l,T],\mathsf{A}) \geq \epsilon'$ *with running in time* t' *where* $t' = 2t + O(t)$ *(*$O(t)$ *is polynomial) and* $\epsilon' \geq \frac{(\epsilon - q_{sig}q_{hash}2^{2-k})^2}{(T+1)^2 \cdot q_{hash}} - \frac{\epsilon - q_{sig}q_{hash}2^{2-k}}{2^l(T+1)}$.

Proof. Please refer to the appendix A. □

Theorem 2. *The proposed scheme is secure based on the definition 1.*

Proof. Please refer to the appendix B. □

5 Performance Analysis

In this section, we analyze the computational cost to run each algorithm. The computational cost for each algorithm in the proposed scheme is given below. We referred to the Itkis et al's work [9] (in Section 3.3) for the computational complexity of the basic operation. (T, k, l are described at the key generation in Section 3.)

- **Key generation:** The computation cost to generate a common modulus n_{id} and $e_{id,0}, ..., e_{id,T}$ is $O(k^5)$ and $O(l^4T)$ respectively [9]. Since the computational complexity to execute one modular exponentiation in $Z_{n_{id}}^*$, one modular inversion, and one modular multiplication in $Z_{\phi(n_{id})}^*$ are $O(k^2l)$, $O(k^2)$, $O(kl)$ respectively [9], the overall computational complexity of the key generation algorithm is $O(k^5 + l^4 + Tk^2l + Tk^2 + Tkl)$ excluding the cost of running the secure hash function.

- **Proxy generation:** As three modular exponentiation in $Z^*_{n_{id}}$ are needed to run, the proxy generation algorithm requires a total time of $O(k^2 l)$.
- **Proxy verification:** To verify the certificates in W, it needs $O(|W|k^2 l)$ running times. In self-delegation case, the computational cost for verifying the received delegation keys is $O(bk^2 l)$ where b is the number of the received delegation keys. Thus, the overall running time is $O((|W| + b)k^2 l)$.
- **Proxy signature generation:** As three modular exponentiation in n_{id} are needed to run, $O(k^2 l)$ running time is required.
- **Proxy signature verification:** Since the certificates in W should be verified, $O(|W|k^2 l)$ running time is required.

6 Relation Between Malkin et al's Construction and the Proposed Scheme

In a sentence, our result does not severely affect Malkin et al's work because KI can be constructed with the proposed technique and vice versa.

What we have addressed is to design a proxy signature scheme providing self-delegation in real-world setting. In a practical aspect, a new user can participate in the system or leave the system. In Malkin et al's proxy signature, the system starts after all users have their keys. We have thought that this construction is not appropriate for the real-world construction because joining or leaving users are always possible. As we have said, Malkin et al's construction leads scalability problem and, if the system starts before all users get their keys, a security problem rises due to the existence of the global secret key. Surely, it is guessed that this construction is to show the theoretical equivalence between the proxy signature and KI. In their security model, the adversary starts after the key generation procedure. Thus, the security problem we indicated does not happen.

In a theoretical sense, the proposed scheme can also be transferred to the generic construction. In the proposed construction, the proxy signature (with n users and the maximum number of self-delegation allowed for each user is c) can be constructed from n instances of $(c, c-1)$ KI. In this case, the adversary in the proxy signature may be more powerful than that of KI because the adversary in the proxy signature can success without breaking the security of KI.

References

1. A. Boldyreva, A. Palacio, and B. Warinsch. Secure Proxy Signature Scheme for Delegation of Signing Rights. IACR ePrint Archive, 2003, available at http://eprint.iacr.org/2003/096/.
2. M. Bellare and P. Rogaway. Random Oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conferences on Computer and Communication Security*, pp. 62-73, Nov. 1993.
3. Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-Insulated Public Key Cryptosystems. In *Advances in Cryptology-Eurocrypt'02*, LNCS 2332, pp. 65-82, 2002.

4. Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong Key-Insulated Signature Schemes. In *Proceedings of PKC 2003*, LNCS 2567, pp. 130-144, 2003.
5. M. Gasser, A. Goldstein, C. Kaufman, and B. Lampson. The Digital Distributed Security Architecture. In *Proceedings of National Computer Security Conference*, 1989.
6. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, Vol. 17, No. 2, pp. 281-308, 1988.
7. O. Goldreich, B. Pfitzmann, and R. L. Rivest. Self-Delegation with Controlled Propagation - or - What if You Lose Your Laptop. In *Advances in Cryptology-CRYPTO'98*, LNCS 1462, pp. 153-168, 1998.
8. L. Guillou and J. Quisquater. A paradoxical identity-based signature scheme resulting from zero-knowledge. In *Advances in Cryptology-CRYPTO'88*, LNCS 403, pp. 216-231, 1990.
9. G. Itkis and L. Reyzin. Forward-secure signatures with optimal signing and verifying. In *Advances in Cryptology-CRYPTO'01*, LNCS 2139, pp. 332-354, 2001.
10. T. Malkin, S. Obana, and M. Yung : The Hierarchy of Key Evolving Signatures and a Characterization of Proxy Signatures. In *Advances in Cryptology-Eurocrypt'04*, LNCS 2332, pp. 306-332, 2004.
11. T. Malkin, S. Obana, and M. Yung : The Hierarchy of Key Evolving Signatures and a Characterization of Proxy Signatures. IACR ePrint Archive, 2004, available at http://eprint.iacr.org/2004/052/.
12. M.Mambo, K.Usuda, and E.Okamoto : Proxy signatures: Delegation of the power to sign messages. *IEICE Trans. Fundam.*, E79-A(9), pp1338-1354, 1996.
13. B. Neuman. Proxy-based authorization and accounting for distributed systems. In *Proceedings of 13th International Conference of Distributed Computing Systems*, pp. 283-291, 1993.

Appendix A Proof of Theorem 1

A uses **F-PSsd** as a subroutine. A gets an instance of strong RSA problem $(n, \alpha \in Z_n^*)$ as an input and solves it with **F-PSsd**. Initially, A generates an device owner $id \in Z$, $(e_{id,0}, ..., e_{id,T})$ with the same procedure as the key generation algorithm. Before the execution, A manipulates the random tape of **F-PSsd** so that $n_{id} = n$, $t_{id,0}^{e_{id,j}} = \alpha$ during the execution of **F-PSsd** ($t_{id,0}$ is unknown to A). A computes the public key $V = 1/\alpha^{e_{id,0} \cdot e_{id,1} \cdot \cdots \cdot e_{id,j-1} \cdot e_{id,j+1} \cdot \cdots \cdot e_{id,T}}$ and stores it. (In the rest of the proof, we describe $a_{id,x}$ as a_x for a simple description.)

After these initial establishment, A repeatedly runs **F-PSsd** until **F-PSsd** attacks the device which has SK_j. Assume Dev_j has SK_j as a signing key. In this case, A can provide the answer of the oracle queries and request by **F-PSsd** as follows.

- **Private Keys:** A can provide all private keys except SK_j. For example, if **F-PSsd** requests SK_o $(o < j)$, A can compute the secret information of the private key $s_o = \alpha^{e_0 \cdot e_1 \cdot \cdots \cdot e_{o-1} \cdot e_{o+1} \cdot \cdots \cdot e_{j-1} \cdot e_{j+1} \cdot \cdots \cdot e_T}$. Surely, A can provide SK_o where $(o > j)$ with a similar computation as $(o < j)$ case.
- **Proxy Signature and Delegation Query:** The answer of Dev_j's proxy signature query for a message $m \in \{0,1\}^*$ (or a delegation query with a

description of the signing right $m_w \in \{0,1\}^*$) can be provided by the following procedure. As **F-PSsd** has the information that all devices have except Dev_j, **F-PSsd** can generate the proxy if A just gives **F-PSsd** the OD's signature part when the delegation query is requested. Thus, in the rest of the proof, we regard that the delegation query is the same as the proxy signature query. In the signature query, **F-PSsd** gives A (m, σ_{last}) as inputs (where σ_{last} is the last certificate in W which Dev_j has as a part of proxy.

1. Select $c_s \in Z_n^*, \sigma_s \in \{0,1\}^l$ randomly and compute $y_s = c_s^{e_j \cdot \sigma_{last}} V^{\sigma_s \cdot \sigma_{last}}$.
2. Return (W, m, j, id, c_s, e_j) and stores $(s, j, e_j, y_s, \sigma_s, c_s, m)$ in signature query table.

– **Hash Query:** For t-th hash query with an input (m_t, y_t), A first finds the tuple $(\cdot, \cdot, \cdot, y', \sigma', \cdot, m')$ such that $m_t = m'$ and $y_t = y'$. If it exists, return σ'. Otherwise, A generates a random $\sigma_t \in \{0,1\}^l$ and returns (t, m_t, y_t, σ_t). Finally A stores it in the hash query table.

If **F-PSsd** outputs a successful forgery, let the output be $(W, m_1, j, id, c_1, e_j)$. Then A finds the corresponding hash query result $(h', m_1, y_1 = c_1^{e_j \cdot \sigma_{last1}} V^{\sigma_1 \cdot \sigma_{last1}}, \sigma_1)$ in the hash table and memorizes it.

Now, A initializes **F-PSsd** such that the previous execution runs again. For the same input as the previous execution, A answers the same hash query result that is stored in the hash query table during the second execution. However, if the input of h'-th hash query is (m_1, y_1) that is the same as the first execution, A generates a new random value $\tau_1 \in \{0,1\}^l$ and returns it. If (m_1, y_1) is queried at different times or not queried, A stops the execution and restarts the experiment.

After the second execution, A gets $(W, m_1, j, ID, c_2, e_j)$ as **F-PSsd**'s output. As the same execution is done before h'-th hash query, both executions have the same input at h'-th hash query. So $y_1 \equiv c_1^{\sigma_1 \cdot \sigma_{last1}} V^{e_j \cdot \sigma_{last1}} \equiv c_2^{\tau_1 \cdot \sigma_{last1}} V^{e_j \cdot \sigma_{last1}} \pmod n$. Thus, $(c_1/c_2)^{e_j} \equiv V^{\tau - \sigma_1} \equiv \alpha^{(e_0 \cdot \ldots \cdot e_{j-1} \cdot e_{j+1} \cdot \ldots \cdot e_T) \cdot \tau - \sigma_1} \pmod n$.

Because e_j is guaranteed to be relatively prime with $e_0 \cdot \ldots \cdot e_{j-1} \cdot e_{j+1} \cdot \ldots \cdot e_T$ and $\tau - \sigma_1$ has at least one fewer bit than e_j, $gcd((e_0 \cdot \ldots \cdot e_{j-1} \cdot e_{j+1} \cdot \ldots \cdot e_T) \cdot (\tau - \sigma_1), e) = gcd(\tau - \sigma_1, e) < e$ (as long as $\sigma_1 \neq \tau$). Thus, $r = e/gcd((e_0 \cdot \ldots \cdot e_{j-1} \cdot e_{j+1} \cdot \ldots \cdot e_T) \cdot (\tau - \sigma_1), e) > 1$ and by Lemma 1 in [9], A will be able to efficiently compute the r-th root of α.

Probability Analysis: In the signature query step, since **F-PSsd** gets A's response instead of the true random oracle, it can be happened that for some signature query, the hash value that A needs to define has already been defined through a previous answer to a hash query. In this case, as A cannot reply an altered hash value during the second execution, **F-PSsd**'s success probability with the interaction of A is reduced at most to $\epsilon - q_{sig} q_{hash} 2^{2-k}$ $(\because |Z_n^*| = 4q_1 q_2 > 2^{k-2})$. Let this value be δ. In addition, A can continue the execution only if **F-PSsd** outputs a forgery against Dev_j. Let this probability be $\epsilon_j = \delta/(T+1)$.

Suppose that p_h is the probability that **F-PSsd** succeeds to the forgery with h-th hash query result. Then, $\epsilon_j = \sum_{h=1}^{q_{hash}} p_h$.

Now we consider A's success probability. For A's success, **F-PSsd** should make the forgery with h-th hash query result at both executions of **F-PSsd**.

Since both executions are independent, the probability is at least p_h^2. But the additional requirement that the hash query result in the first execution should be different from that in the second execution ($\sigma_1 \neq \tau_1$) reduces the probability to $p_h(p_h - 2^{-l})$.

Therefore, $\epsilon' \geq \sum_{h=1}^{q_{hash}} p_h(p_h - 2^{-l})$ and this leads the following inequalities by Lemma 1 in [9].

$$\begin{aligned}
\epsilon' &\geq \sum_{h=1}^{q_{hash}} p_h(p_h - 2^{-l}) \\
&\geq \epsilon_j^2/q_{hash} - 2^{-l}\epsilon_j \\
&\geq \frac{\delta^2}{(T+1)^2 \cdot q_{hash}} - \frac{\delta}{2^l(T+1)} \\
&= \frac{(\epsilon - q_{sig}q_{hash}2^{2-k})^2}{(T+1)^2 \cdot q_{hash}} - \frac{\epsilon - q_{sig}q_{hash}2^{2-k}}{2^l(T+1)}
\end{aligned}$$

By solving the above quadratic inequality for $(\epsilon - q_{sig}q_{hash}2^{2-k})/(T+1)$, the following inequality is derived.

$$\begin{aligned}
(\epsilon - q_{sig}q_{hash}2^{2-k})/(T+1) &\leq 2^{-(l+1)}q_{hash} + \sqrt{2^{-(2l+2)}q_{hash}^2 + q_{hash}\epsilon'} \\
&\leq 2^{-(l+1)}q_{hash} + 2^{-(l+1)}q_{hash} + \sqrt{q_{hash}\epsilon'} \\
&= 2^{-l}q_{hash} + \sqrt{q_{hash}\epsilon'}
\end{aligned}$$

Therefore, $\epsilon \leq (T+1)(2^{-l}q_{hash} + \sqrt{q_{hash}\epsilon'}) + q_{sig}q_{hash}2^{2-k}$.

Probability Analysis: A executes **F-PSsd** twice. As the other operations that A executes have polynomial-time complexity, it can be derived that $t' = 2t + O(t)$ where $O(t)$ is the polynomial of T, l, k. □

Appendix B Proof of Theorem 2

Because $\mathbf{Succ}^{SRSA}([k,l,T],\mathsf{A}) \geq \epsilon'$ if $\mathbf{Succ}^{PSsd}(PSsd[k,l,T],\mathsf{B}) \geq \epsilon$, by antithesis, if $\mathbf{Succ}^{SRSA}([k,l,T],\mathsf{A}) \leq \epsilon'$, then $\mathbf{Succ}^{PSsd}(PSsd[k,l,T],\mathsf{B}) \leq \epsilon$. Thus, $\mathbf{InSec}^{PSsd}(PSsd[k,l,T],t,q_{sig}) = \epsilon$ because the modified strong RSA assumption indicates that $\mathbf{Succ}^{SRSA}([k,l,T],\mathsf{A}) \leq \epsilon'$ where ϵ' is negligible.

If we solves the inequalities in Theorem 1, we can derive that $\epsilon \leq (T+1)(2^{-l}q_{hash} + \sqrt{q_{hash}\epsilon'}) + q_{sig}q_{hash}2^{2-k}$ (For more detail, see the appendix). As both q_{hash} and q_{sig} are much smaller than 2^k and 2^l, ϵ is negligible. Thus, based on the definition 1, the proposed scheme is secure. □

Extended Sanitizable Signatures*

Marek Klonowski** and Anna Lauks

Institute of Mathematics and Computer Science, Wrocław University of Technology,
ul. Wybrzeże Wyspiańskiego 27
50-370 Wrocław, Poland
Marek.Klonowski@im.pwr.wroc.pl, Anna.Lauks@im.pwr.wroc.pl

Abstract. Sanitizable signatures introduced by Ateniese et al. is a powerful and fairly practical tool that enables an authorised party called the censor to modify designated parts of a signed message in an arbitrary way without interacting with the signer. In our paper we present several extensions of this paradigm that make sanitizable signatures even more useful. First of all we show how to limit the censor's abilities to modify mutable parts of a signed message to a predetermined set of strings. In our next proposal we show how to construct a scheme wherein the censor can insert an arbitrary string into a document, but this must be the same string in all designated places. We also present a construction based on a sanitizable signature that allows the censor to present only a constant number of versions of the sanitized message. Another extension provides so-called strong transparency. In this case the verifier does not know which parts of the message could have been modified. Finally, we point out new applications of sanitizable signatures based on combining them with time released cryptography techniques.

1 Introduction and Previous Work

The concept of a particular kind of sanitizable signatures we refer to was introduced by Ateniese et al. in [1]. According to this notion, a sanitizable signature is a new kind of digital signature with the following property: a designated party called the censor can change (designated) parts of a signed message in such a way that the signature of modified message still remains valid. The point is that this change can be done without the collaboration of the signer. This property can be obtained thanks to the chameleon hashes that are extensively used in the construction of the signature. Chameleon hashes, also called trapdoor commitments, allow only the owner of a special private key to find a collision. From the censor's point of view, a single message signed using a sanitizable signature scheme is a set (possibly infinite) of "messages allowed by the signer". It is a fairly practical solution in many scenarios presented in [1] like injections with

* Partially supported by KBN grants 0 T00A 003 23 (years 2003–2005) and 3T11C 011 26 (year 2006).

** Contact author. Marek Klonowski is supported by The Foundation for Polish Science (FNP) in the year 2006.

M.S. Rhee and B. Lee (Eds.): ICISC 2006, LNCS 4296, pp. 343–355, 2006.
© Springer-Verlag Berlin Heidelberg 2006

advertisements in various files, anonymising data-bases, secure routing and many others, wherein semi-trusted provider has to modify a broadcasted message on behalf of the author or owner of particular data.

However, in some scenarios the fact that mutable parts of a message can be modified by the censor in an arbitrary way (the censor can insert any string in the place of the original one and the signature would be still accepted during the verification procedure) can be very problematic. Authors of sanitizable signatures scheme [1] proposed inserting in the immutable part of the message an auxiliary information like "write here the day of the week" expressed in natural language. In many scenarios this would be insufficient. We can imagine, for example, a situation in which we have to deal with a huge amount of different kinds of messages that have to be processed in an automatic manner. We should also take into account attacks based on injecting malicious code into a program. In other words we need an efficient mechanism that would limit the choice of the censor to predetermined set.

Another problem appears when we need to allow the censor only to change limited number of blocks in the message (chosen arbitrary by the censor from potentially mutable blocks). For example, we would like to give him the opportunity of changing k blocks out of n possible.

Moreover, in the original scheme the censor can insert material which is independent of other blocks. Let us consider a situation in which the censor would like to present an application form with several gaps that are to be filled with the same name of a particular candidate for a particular position. Existing scheme does not support this kind of scenario, wherein mutable blocks should be correlated.

All these needs are addressed by modified sanitizable signature schemes, described below. We also propose some other extensions – we describe a version of the scheme with so called strong transparency. This means that the verifier does not know which parts of a signed message are mutable. At the end, we briefly outline some minor extensions and applications based on combining sanitizable signatures with time released cryptography techniques.

1.1 Related Work

Steinfeld, Bull and Zheng presented *Content Extraction Signatures* [17], that allow everyone to hide some blocks of a signed message. Another significant paper is [8] by Johnson et al. These authors introduced many useful schemes. One of them is the *Redactable Signature*. Functions offered by this scheme is in principle very similar to the *Content Extraction Signature*, but both schemes are built on different cryptographic primitives. Other papers presenting various schemes that allow hiding parts of a signed message are papers by Miyazaki et al. [13,14] and Izu et al. [7]. In these papers one can find many interesting schemes tailored for different scenarios. Most of them take into account many parties that are interested in hiding different parts of a message. Comparison of most of these schemes can be found in [14].

Our paper refers mainly to Ateniese et al.'s *sanitizable signatures* mentioned above and described in more detail below. Roughly speaking, in this scheme the designated censor can change the content of designated (so-called *mutable*) parts of a signed message without interaction with the signer. In previous schemes the censor could only hide some parts of the message. For this reason the scheme of Ateniese et al. has different functions and different applications than previous proposals. Let us note that the first kind of signatures (allowing only hiding some blocks of a signed message) were also called *sanitizable signatures* in some papers ([13,14]).

We also extensively use the *accumulator technique* introduced in [3] and developed and used among others in [8]. Accumulators are hash functions that "contain" a set of messages S. Given all messages from S one can prove that a particular message $m \in S$. Moreover, one can prove this fact in a very efficient manner without revealing the rest of the set S. We also propose combining sanitizable signatures with various time-released techniques. The idea of time-released cryptography is attributed to Timothy May [16,11]. The goal is to encrypt a message or sign a document in such a way that it can be decrypted (verified) after a predetermined interval of time has passed [16] or a particular condition is fulfilled [9].

Sanitizable signatures and other schemes mentioned above can be regarded as a particular kind of the schemes that allow computing a signature without any secret key from another signature in a predetermined and limited fashion without cooperation with the original signer. This idea was introduced several years ago by Ronald Rivest [15]. Many impressive schemes based on this idea have been proposed. The most important schemes we are aware of are homomorphic signatures - in particular the *transitive signature* of Micali and Rivest [12].

1.2 Organisation of This Paper

In the second section we recall the sanitizable signatures of Ateniese et al. [1]. The third section is devoted to techniques limiting the range of possible modifications made by censor. This is based on the accumulator technique also recalled in this section. In the fourth section we show how to force the censor to make the same changes in logically linked blocks. In the fifth section we present techniques that strongly limit the censor - i.e. allow him to change only k out of n mutable blocks. The sixth section contains a modification of the scheme that provides strong transparency mentioned above. In the seventh section we consider combining time released cryptography with sanitizable signatures. The last section contains final remarks and open questions that we find important.

2 The Sanitizable Signature of Ateniese et al.

Ateniese et al. [1] introduced the notion of sanitizable signatures that allow modifying some blocks. They also proposed a kind of sanitizable signature based on chameleon hashes. The idea of this proposal is as follows. The message is

divided into blocks. Then the value of the chameleon hash function of some blocks, called *mutable*, is computed. The value of the"regular" hash function is computed for other *immutable*, blocks. Then the signer signs hash value of all concatenated blocks. The chameleon hashing used in the construction allows designated party, (censor) to find a collision. So, in other words he is able to change the mutable blocks of the message in such a way that the signature remains still valid. We recall briefly this scheme, since our contribution can be regarded as its extension of this scheme.

2.1 Chameleon Hashing

From the practical point of view the kind of chameleon hash function that is used plays very important role. Authors of the idea of sanitizable signatures recommended using chameleon hashes introduced in [2]. They chose this scheme because of its strongly unforgeable property, what means that pointing collision will not reveal the secret key (trapdoor value). For this reason, one public key can be used many times. According to this scheme, in order to compute chameleon hash of a message m, first the prime numbers q and p (of bitlength κ and such that $p = uq + 1$) are chosen. Then the private key $x \in [1, q-1]$ is selected uniformly at random and the public key is computed as $y = g^x$, where g is a generator of subgroup of squares of order q. After that random values $\alpha, \beta \in [1, q-1]$ are chosen and the parameter e is computed as $e = H(m, \alpha)$, where $H(\cdot, \cdot)$ is a standard collision-resistance hash function, mapping arbitrary-length bitstrings to strings of fixed length. Finally the chameleon hash of m is computed as:

$$CH_y(m, \alpha, \beta) = \alpha - (y^e g^\beta \mod p) \mod q.$$

The collision can be found only by the owner of the private key x. The idea of finding a collision is as follows. Let $C = CH_y(m, \alpha, \beta)$. First the random number $k' \in [1, q-1]$ is generated. Then the other values are computed as:

$$\begin{aligned} \alpha' &= C + (g^{k'} \mod p) \mod q, \\ e' &= H(m', \alpha'), \\ \beta' &= k' - e'x \mod q. \end{aligned}$$

Of course $CH_y(m', \alpha', \beta') = C$. Indeed,

$$\alpha' - (y^{e'} g^{\beta'} \mod p) \mod q = C + (g^{k'} \mod p) \mod q - (g^{xe'} g^{\beta'} \mod p) \mod q = C$$

2.2 Description of the Sanitizable Signature Scheme

Preliminaries: Let x_S, y_S denote the private and the public key of the signer respectively. Let x_C, y_C be the private and the public key of the censor (a person that is allowed to modify some parts of the signed message). A chameleon hash of the message m with a random parameter r under the public key y_C will be denoted by $CH_{y_C}(m, r)$.

Let $Sig_{x_S}(m)$ be a secure digital signature of a message m computed by the owner of the private key x_S.

Preparing a sanitizable signature of m: Let $m = m_1||\ldots||m_t$. In order to create a sanitizable signature of the message m, the signer has to choose a random number ID_m and assign blocks of the message m, say $m_{i_1}, \ldots, m_{i_k}$ that the censor will be able to modify. We call these blocks *mutable* or *sanitizable*. After that, using the public key of the censor y_C, the signer computes:

$$\bar{m}_i = CH_{y_C}(ID_m||i||m_i, r_i) \quad \text{for } i \in \{i_1, \ldots, i_k\}$$
$$\bar{m}_i = m_i||i \quad \text{for other } i.$$

The sanitizable signature is finally computed as:

$$\sigma = Sig_{x_S}(ID_m||t||y_C||\bar{m}_1||\ldots||\bar{m}_t).$$

Verifying a sanitizable signature of m: The procedure of signature verification depends on the kind of signing scheme. During the verification identifier ID_m, the public key of the censor and random values r_i are attached (necessary for computing the proper chameleon hashes). That is,

$$\text{VERIFY}(\sigma,\, m,\, y_S,\, y_C,\, ID_m,\, r_{i_1},\, \ldots,\, r_{i_k}) \longrightarrow \{\text{FALSE}, \text{TRUE}\}.$$

2.3 Remarks

Let us note that only the censor has the private key x_C corresponding to the public key y_C. For this reason only the censor is able to find "collision" - i.e. arbitrary (ID_m, i, m'_i) and particular r'_i such that $CH_{y_C}(ID_m||i||m_i, r_i) = CH_{y_C}(ID_m||i||m'_i, r'_i)$. This property of chameleon hashes allows the censor to modify parts of the message in such a way that the signature σ remains still valid.

3 Limiting the Set of Possible Modifications of a Single Mutable Block

One of the most important problems with regular sanitizable signatures presented in the previous section is that the censor can exchange particular block in arbitrary way. In practice the signer often needs to give only limited list of possible blocks that can be inserted in particular position in the text. As it is suggested in [1], the signer can insert in the immutable part of the text a kind of limitation like *write here the name of the day* just before a mutable block. Anyway such a solution implemented directly can not be automatically verified. We should also take into account various attacks based on "injecting" malicious code.

Let us note that we can overcame this weakens by "naive" solution wherein length of the value of hash function is linear in the cardinality of the set of acceptable block's contents in particular position.

It would be much more efficient to use hash and sign paradigm with a one-way hash function that has the same output only at elements from the set of *acceptable* values pointed by the censor.

I.e. we need *set-hash function* $H : \{0,1\}^* \to \{0,1\}^l$ that for elements of the finite set $M = \{m_1, \ldots, m_k\}$ has the value h i.e. for $m_i \in M$ holds $H(m_i) = h$. Moreover this function should behave like a regular hash function. Loosely speaking, for given h and M it is computationally infeasible in practice to find $m' \notin M$ such that $H(m') = h$.

Since we cannot find such a function we propose two simple solutions partially similar to the idea of set-hash function. One of them is based on so-called accumulator technique, another on Bloom filters. We remark advantages and disadvantages of both of them.

3.1 Sanitizable Signatures and Accumulator Technique

We discuse below techniques based on idea of *accumulators* (see [3]). Let us note that extended accumulator idea was previously used in [8] in another context to built redactable signature scheme. Below we recall accumulator technique.

Preliminaries: Let $N = p \cdot q$ and $p = 2p' + 1, q = 2q' + 1$ for large primes p, p', q, q'. Moreover we assume that $|p| = |q|$. Let $H^*(m)$ be the value of a "regular" hash function of the message m.

Hashing: Let $M = \{m_1, \ldots, m_k\}$ be the set of messages to be hashed. We compute:

$$H(M) = x^{\prod_{i \leq k} H^*(m_i)} \bmod N$$

Verifying: Let us define:

$$H_j(M) = x^{\prod_{i \neq j} H^*(m_i)} \bmod N.$$

Let $z = H(M)$ and $z_j = H_j(M)$. To check if the massage $m_j \in M$, we have to check if the following equality holds:

$$z = z_j^{H^*(m_j)} \bmod N.$$

Note that it is easy to compute z_j if one knows the set M.

Now we can ask if this construction meets security requirements. In particular we would like to know if it is possible in practice to find a preimage for an arbitrary z or collision for the set M i.e. $M' \neq M$ such that $H(M') = H(M)$. This function is indeed secure. Security analysis of this function is non trivial and provided in [8].

Sanitizable signatures with accumulator hashing. Now we can apply accumulators described above to sanitizable signatures. All sub-protocols described in previous section are exactly the same except that:

1. The signer signs the value of an accumulator for the set M allowed for particular block instead of using chameleon hashes.

2. The censor provides the appropriate values z and z_j to prove that $m_j \in M$ during the verification.

Note that everyone can check if M is a set of allowed messages by simple computing $H(M)$. Let us note that very often M can be correctly guessed for example when one can insert number of the day of the month in a particular place. To avoid that signer can add to the set M a random string and reveal it only to the censor. In this scenario only the censor and the signer can present appropriate value z_j.

Shortcomings of an approach based on accumulators. Solution proposed above works - censor has limited permission for modifying signed message. Moreover presented signature for the verifier has a constant size independently on cardinality of "permitted" set M. However, the censor needs to know whole set M to provide modified signature. It can be cumbersome in some application when set M contains large number of elements - for example, a list of employees of a bank. In such a situation one can consider using Bloom filters. This idea is outlined below.

3.2 Sanitizable Signatures and Bloom Filters

Bloom Filters. Bloom filter is a technique described for the first time in [4]. A Bloom filter for a set $M = \{m_1, \ldots, m_t\}$ is represented by an r-bit array B_M. All bits are initially set to 0. Let $h_1, \ldots, h_k$ be k independent hash functions with the common range $\{1, \ldots, r\}$. In this scheme $i - th$ bit is set to 1 if and only if there exists such a message m_j and a function h_l such that $h_l(m_j) = i$. Having m one can try to check if $m \in M$. At the beginning $h_i(m)$ for each i is computed. If $m \in M$ then $h_i(m) = j$ implies $B_M[j] = 1$. If this condition does not hold for any i we certainly know that $m \notin M$. Otherwise, if this condition holds for every i we can assume that $m \in M$. Obviously we should take into account "false positive" error. Probability of a false acceptance can be approximated very well by the following formula:

$$(1 - e^{-k \cdot t/r})^k.$$

Note that one can check if particular message belongs to the set M without knowledge of all elements of the set M. For that reason we could exchange accumulators from previous subsection with Bloom filters in order to overcome the shortcomings of accumulator functions.

Shortcomings of an approach based on Bloom filters. The price of using Bloom filters seems to be high. First of all, the censor can insert in mutable part the message that is unwanted by the original signer. Note that probability of false acceptance can be significantly reduced at the price of using much longer arrays. We suppose that sanitizable signatures with Bloom filters could be applicable in very particular situations - for example in real time system in which accuracy is not extremely important. Let us note that one can make finding $m \notin M$ such that m is accepted during the verification procedure infeasible at the price of

significant extending filter's size (parameter r). However, an array with very large number of bits can be useless.

4 Enforcing the Same Modifications of Different Mutable Blocks

In this section we consider a scenario in which a document contains some number of mutable blocks that should always have the same values. As an example we can imagine an application form with the first name and the family name in several places.

Let us suppose that the signer would like to allow the censor to change these blocks, but at the same time he wants to be sure that the document remains coherent after modifications (all designated blocks should contain the same value). Below we present a signing scheme that supports this requirement.

Preliminaries: Let us assume that g is a generator of Z_p^* and $\mathrm{Sig}(\cdot)$ is a secure digital signing scheme. In order to simplify the notation we skip "mod p" whenever it is obvious from the context. Moreover let us assume that the signer wants to sign a message $m = m_1||m_2||\ldots||m_t$ with k mutable blocks (say $m_{i_1}, \ldots, m_{i_k}$) that should always have the same values.

Signing: In order to sign m, the signer has to generate random values $x_1, \ldots, x_k, r$. Then he computes: $h_i = g^{x_i}$ for $i = 1, 2, \ldots, t$.
Signature of m is a triple: $(c,\ r,\ \mathrm{Sig}(c))$, where:

$$c = h_1^{m_1} \cdot h_2^{m_1} \cdot h_3^{m_3} \cdot \ldots \cdot h_t^{m_t} \cdot g^r.$$

For making blocks $m_{i_1}, \ldots, m_{i_k}$ mutable signer has to reveal (only to) the censor the sum $s = x_{i_1} + \ldots + x_{i_k}$.

Signature Verification: The verification process consists of two steps:

1. checking the validity of the signature $\mathrm{Sig}(c)$,
2. checking if $c = h_1^{m_1} \cdot h_2^{m_2} \cdot h_3^{m_3} \cdot \ldots \cdot h_t^{m_t} \cdot g^r$.

Changing Mutable Blocks: Censor having the sum $s = x_{i_1} + \ldots + x_{i_k}$ can exchange blocks $m_{i_1} = m_{i_2} = \ldots = m_{i_k}$ with any $m_{i_1}^* = m_{i_2}^* = \ldots = m_{i_k}^*$. The new signed message will have a form $m^* = m_1^*||m_2^*||\ldots||m_t^*$, where $m_i^* = m_i$ for $i \notin \{i_1, \ldots, i_k\}$. The signature of the modified message m^* is a triple $(c,\ r^*,\ \mathrm{Sig}(c))$, where $r^* = r - \delta \cdot s$ and $\delta = m_{i_1}^* - m_{i_1}$. Let us stress that using s these values can be easily computed.

Moreover, if $m_{i_1}^* = m_{i_2}^* = \ldots = m_{i_k}^*$ then

$$h_1^{m_1^*} \cdot \ldots \cdot h_t^{m_t^*} \cdot g^{r^*} = h_1^{m_1} \cdot \ldots \cdot h_t^{m_t} h_{i_1}^{\delta} \cdot \ldots \cdot h_{i_k-1}^{m_1} \cdot h_{i_k+1}^{m_1} \cdot \ldots \cdot h_{i_k}^{\delta} \cdot g^{r-\delta \cdot s} =$$
$$= h_1^{m_1} \cdot \ldots \cdot h_t^{m_t} \cdot g^{r-\delta \cdot s+\delta(x_{i_1}+\ldots+x_{i_k})} = c.$$

So the triple $(c, r^*, Sig(c)$ will be accepted as a signature of m^*. Note that it is not feasible to provide a valid signature without s for modified message $m^* \neq m$. Moreover, even with s, the censor cannot present a signature for a message with diffrent mutable blocks.

5 Limiting the Number of Modifications of Mutable Blocks

Let us consider a scenario in which the signer wants to limit the number of mutable blocks of a message m that can be changed by the censor. We want to allow him to choose l out of n blocks to be modified. This goal can be achieved by punishing the censor for too large number of changed blocks. The simplest method of such a punishing is based on revealing the censor's private key. This idea can be implemented by using hash function that causes the key exposure after using a trapdoor. For instance, the chameleon hash defined in [10] as:

$$CH_y(m, r) = y^m \cdot g^r.$$

where $y = g^x$ is the public key, g is the generator of a prime order cyclic group and x is the private key.

Indeed, if someone knows (m_1, r_1) and (m_2, r_2) such that $CH_y(m_1, r_1) = CH_y(m_1, r_2)$ and $m_1 \neq m_2$, he can extract the private key x as:

$$x = \frac{r_2 - r_1}{m_1 - m_2}$$

Of course in the sanitizable signatures scheme with this chameleon hash function, the signer is able to reveal the private key of the censor, if any block is changed. Below we present scheme based on similar trick.

5.1 Scenario Allowing Modification k Out of n Mutable Blocks

Let us suppose that the signer wants to allow the censor to change up to k out of n different mutable blocks. In case of misuse, i.e. if more than the k blocks are modified, the censor's private key will be revealed.

Preliminaries: Let us assume that g is the generator of Z_p^*, x_C is a private key of the censor and $y_C = g^{x_C}$ is his public key. As before, in order to simplify the notation we skip $\bmod p$ whenever it is obvious from the context. Moreover let us assume that the signer wants to sign the message $m = m_1||m_2||\ldots||m_t$ using $\mathrm{Sig}_{x_S}(\cdot)$, a secure digital signing scheme. He wants to allow the censor to change k out of n of blocks of the signed message, say any k blocks out of $m_{i_1}, \ldots, m_{i_n}$. To make it possible, the censor must first choose random values $f_1, f_2, \ldots, f_k$. Then he constructs a polynomial:

$$F(y) = x_C + f_1 y + f_2 y^2 + \cdots + f_k y^k$$

and sends values $g_i = g^{f_i}$ for $0 < i \leq k$ to the signer .

Signing: Signer chooses a random identifier ID_m and computes $z_i = y \cdot g_1^i \cdot \ldots \cdot g_k^{i^k} = g^{F(i)}$ for $i = 1 \ldots n$. After that, the signer chooses random values $r_1, \ldots, r_n$ and computes a chameleon hash of each of the n mutable blocks of the message according to the formula:

$$\widetilde{m}_i = ID_m||i||m_i$$
$$\bar{m}_i = CH(\widetilde{m}_i, r_i) = z_i^{\widetilde{m}_i} \cdot g^{r_i} = g^{F(i)\cdot\widetilde{m}_i + r_i}$$

For immutable blocks of the message m he computes: $\bar{m}_i = m_i||i$. Finally the signature of the message m is:

$$Sig_{x_C}(ID_m||t||y_C||\bar{m}_1||\ldots||\bar{m}_n).$$

Revealing the censor's private key: One can easy see that modification of the i-th block causes leaking of $F(i)$. Obviously, when $k+1$ different mutable blocks are changed, the signer gets $k+1$ distinct points of the polynomial and that allows him to reconstruct the polynomial thanks to the Lagrange interpolation. This suffices to get x_C - the private key of the censor.

6 Strong Transparency

In [1] an idea of transparency in sanitizable signatures was also introduced. Transparency in this context means that no one except the censor and the signer is able to guess whether the message has been sanitized. In the original paper authors distinguished among two kinds of transparency:

Weak transparency: verifier knows exactly which parts of the message are potentially mutable and which parts are immutable. However, he does not know if they were indeed modified.

Strong transparency: verifier cannot distinguish immutable from mutable parts of signed message.

In this section we would like to propose an extension of the basic scheme that supports *strong transparency* in sanitizable signature scheme. Authors of [1] noted that their scheme provides weak transparency. In one of many interesting extensions described in [1] authors tries to provide strong transparency by assigning independent public key to the each block. So the signature has a form:

$$Sig_{x_s}(ID_m||t||y_{C_1}||y_{C_2}||\ldots||y_{C_t}||\bar{m}_1||\ldots||\bar{m}_t)$$

where t is a number of blocks, $\bar{m}_i$ is a chameleon hash value of $i-th$ block of a message using public key y_{C_i} of a particular censor. Thanks to this trick each block could have been modified by a diffrent censor. However, authors of [1] remarked that this technique can be also used in order to provide strong transparency property. They suggest to use "dummies" - each block of the message is potentially mutable, but private keys assigned to some blocks are random "dummy" strings – no one knows corresponding private key. In effect no one can change a block in which hash with "dummy" key is used. This solution has a shortcoming recognised already by authors of this proposal. We should take into account that very often in practise there are be several well-known censors with established public keys. They could be recognised easily and distinguished from "dummies". In such a (probable in practice) situation this solution cannot satisfy requirement of strong transparency.

6.1 Modified Scheme Supporting Strong Transparency

Idea of Ateniese et al. can be slightly enhanced in order to obtain true strong transparency. The main trick is as follows: no one but the censor and the signer knows relation between public keys used during chameleon hashing and identity of the censor that posses corresponding private key that allows modifications. This can be simply achieved by reviling the private key only to the censor. We propose below a generic extension that can be combined with chameleon hash functions based on discrete logarithm problem.

Let $E_y(\cdot)$ denote a secure asymmetric encryption scheme providing *public key confidentiality*. Roughly speaking, this property means that having encrypted message and a public key y, no one can answer if the message was encrypted by using y with probability non-negligibly greater than 0.5 without corresponding private key. There are many schemes with this property. The simplest example is ElGamal encryption scheme. More details can be found for example in [6].

Preliminaries: Let us assume that y_{C_i} is a public key of the $i-th$ censor and x_{C_i} denote the corresponding private key. For the $i-th$ block the signer prepares another public key $\hat{y}_{C_i} = (y_{C_i})^{k_i} = g^{x_{C_i} \cdot k_i}$, where g is a generator of an appropriate group. Signer computes also $E_{y_{C_i}}(k_i)$ – a ciphertext of k_i encrypted using the key y_{C_i}. In practise the ElGamal encryption scheme can be used here. Note that private key corresponding to $\hat{y}_{C_i}$ is $x_{C_i} \cdot k_i$.

Signing: Modified signature has a form:

$$Sig_{x_S}(ID_m||t||\hat{y}_{C_1}||\hat{y}_{C_2}||\ldots||\hat{y}_{C_t}||\bar{m}_1||\ldots||\bar{m}_t)$$

where $\bar{m}_i$ is a chameleon hash of the block m_i created by using key $\hat{y}_{C_i}$ analogously like in section 2.2. The signature should be provided with all ciphertexts $E_{y_{C_i}}(k_i)$ for $1 \leq i \leq t$ to the censor.

Sanitizing: i-th designated censor can easily retrieve value k_i from $E_{y_{C_i}}(k_i)$. Then he can compute the private key $x_{C_i} \cdot k_i$ necessary for sanitizing the block of the message. At this stage finding a collision is simple.

Verifying: Verification process looks exactly like in original scheme described in section 2.2. Note that the verifier uses simply the public keys signed by the original signer.

Let us note that in the proposed scheme decryption of all $E_{y_{C_i}}(k_i)$ is necessary if one needs to modify the i-th block. Anyway, knowledge gained once could be used many times for preparing many version of original message with modified particular block.

6.2 Open Question: Super Strong Transparency

We suppose that it would be very useful to have a scheme that provides even more strong property. Namely, we would like to construct a scheme that hides structure of a document to be sanitized. All proposals introduced up to now are based on the idea of dividing whole messages into independent blocks. One can

easily note that a verifier is aware of this division. This knowledge can be useful and reveals some (possible) intentions of the signer. It seems to be very useful to have a scheme with property that we call *super strong transparency* - no party except the signer and the censor is able to establish what was sanitized and what could be potentially sanitized - even the division into mutable/immutable blocks remains hidden. We suspect that idea can be particularly interesting if we combine it with limiting the set of possible modifications that can be injected into message by the censor. The naive solution based on dividing messages into single-bit blocks is in principle useless because of many obvious reasons e.g. size of the signature as well as resources necessary for signature creation and its verification. We left this issue of constructing practical scheme with super strong transparency as an open problem.

7 Sanitizable Signatures and Time-Released Crypto

We think that it could be very fruitful to combine the idea of sanitizable signatures with time-released crypto techniques. They provide the possibility of revealing a secret or verifying a signature after a pre-determined time has passed ([16]) or other certain events have happened ([9]). In fact all these techniques reveal particular secret values when a certain condition is fulfilled. We can easily see that each legal value x can be given as $\hat{x} = hash(x, r)$ where r is a random value. So to prove the validity of x, the censor needs r. This value can be revealed at a later time, using techniques from [9,16]. Thanks to this simple improvement we can allow the censor to insert more and more blocks into the signed message in particular position depending on the situation. For example, we can allow inserting certain subtitles to a film if the age of a viewer is confirmed.

8 Conclusions, Open Problems

We have presented several extensions of sanitizable signatures and their applications. We hope that they can be useful and make the original scheme even more applicable in many business situations, especially in a distributed environment. We feel that many questions about sanitizable signatures, as well as other schemes that allow limited changes of signed messages, remain open. For example a more efficient way of indicating acceptable set of modifications could be very useful. We suppose that it would be very interesting to prepare a practical sanitizable signature scheme that is not based on dividing the document into blocks and provides strong transparency.

Acknowledgement

We would like to thank anonymous reviewers for their valuable comments.

References

1. Ateniese G., Chou D. H., de Medeiros B., Tsudik, G.: *Sanitizable Signatures*, European Symposium on Research in Computer Security ESORICS 2005, LNSC 3679, pp. 159-177, Springer-Verlag 2005.
2. Ateniese G., de Medeiros B.: *On the Key Exposure Problem in Chameleon Hashes*, Security in Communication Networks 2004, LNCS 3352, pp. 165-179, Springer-Verlag 2005.
3. Benaloh J., de Mare M.: *One-Way Accumulators: A Decentralised Alternative to Digital Signatures*, Advances in Cryptology EUROCRYPT 1993, LNCS 765, pp. 274-285, Springer-Verlag 1994.
4. Bloom B.H..: *Space/time trade-offs in hash coding with allowable errors*, Communication of ACM , vol. 13, no. 7, pp. 422-426, July 1970.
5. Chen X., Zhang F., Kim K.: *Chameleon Hashing Without Key Exposure*, Information Security ISC 2004, LNCS 3225, pp. 87-98, Springer-Verlag 2004.
6. Golle P., Jakobsson M., Juels A., Syverson P. F.: *Universal Re-encryption for Mixnets*, Topics in Cryptology CT-RSA 2004, LNCS 2964, pp. 163-178, Springer-Verlag 2004.
7. Izu T., Kanaya N., Takenaka M., Yoshioka T., *PIATS: A Partially Sanitizable Signature Scheme* ICICS 2005, LNCS 3783, pp.72-83, Springer-Verlag, 2005
8. Johnson R., Molnar D., Song D. X., Wagner D.: *Homomorphic Signature Schemes*, Topics in Cryptology CT-RSA 2002, LNCS 2271, pp. 244-262, Springer-Verlag 2002.
9. Klonowski M., Kutyłowski M., Lauks A., Zagórski F.: *Conditional Digital Signatures*, TrustBus 2005, LNCS 3650, pp. 490-497, Springer-Verlag 2005.
10. Krawczyk H., Rabin T.: *Chameleon Signatures*, Proceedings of the Network and Distributed System Security Symposium NDSS 2000, pp. 143-154.
11. May T. C.: *Time-release crypto* , February 1993, `http://www.hks.net/cpunks/cpunks0/1460.html`.
12. Micali S., Rivest R. L.: *Transitive Signatures Schemes*, Topics in Cryptology CT-RSA 2002, LNCS 2271, pp. 236-243, Springer-Verlag 2002.
13. K. Miyazaki, G. Hanaoka, H. Imai, Digitally Signed Document Sanitizing Scheme Based on Bilinear Maps, ACM Symposium on InformAtion, Computer and Communications Security (ASIACCS 2006), Taipei, Taiwan, March 21-24, 2006.
14. K. Miyazaki, M. Iwamura, T. Matsumoto, R. Sasaki, H. Yoshiura, S. Tezuka, H. Imai, *Digitally Signed Document Sanitizing Scheme with Disclosure Condition Control*, The Institute of Electronics, Information and Communication Engineers (IEICE) Trans. on Fundamentals, Vol, E88-A, pp. 239-246, No. 1, January 2005.
15. Rivest R. L.: *Two Signatures Schemes*, Slides from talk given at Cambridge University, Oct. 17, 2000, `http://theory.lcs.mit.edu/~rivest/publications.html`.
16. Rivest R. L., Shamir A., Wagner D.A.: *Time-lock puzzles and timed-released Crypto*, Revisited version, March 10, 1996, `http://theory.lcs.mit.edu/~rivest/publications.html`.
17. R. Steinfeld, L. Bull, Y. Zheng, *Content Extraction Signatures*, ICISC 2001, LNCS 2288, pp.285-304, Springer-Verlag, 2001.

Author Index