Robert H. Deng
Feng Bao
HweeHwa Pang
Jianying Zhou (Eds.)

# Information Security Practice and Experience

LNCS 3439

First International Conference, ISPEC 2005
Singapore, April 2005
Proceedings

Springer

Lecture Notes in Computer Science 3439

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

## Editorial Board

Robert H. Deng Feng Bao
HweeHwa Pang Jianying Zhou (Eds.)

# Information Security Practice and Experience

First International Conference, ISPEC 2005
Singapore, April 11-14, 2005
Proceedings

Springer

Volume Editors

Robert H. Deng
Singapore Management University
469 Bukit Timah Road, Singapore 259756
E-mail: robertdeng@smu.edu.sg

Feng Bao
HweeHwa Pang
Jianying Zhou
Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
E-mail: {baofeng, hhpang, jyzhou}@i2r.a-star.edu.sg

Library of Congress Control Number: 2005923658

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

ISSN 0302-9743
ISBN-10 3-540-25584-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-25584-0 Springer Berlin Heidelberg New York



Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11407959 06/3142 5 4 3 2 1 0

# Preface

The inaugural Information Security Practice and Experience Conference (ISPEC) was held on April 11–14, 2005, in Singapore.

As applications of information security technologies become pervasive, issues pertaining to their deployment and operation are becoming increasingly important. ISPEC is intended to be an annual conference that brings together researchers and practitioners to provide a confluence of new information security technologies, their applications and their integration with IT systems in various vertical sectors. The Program Committee consisted of leading experts in the areas of information security, information systems, and domain experts in applications of IT in vertical business segments.

The topics of the conference covered security applications and case studies, access control, network security, data security, secure architectures, and cryptographic techniques. Emphasis was placed on the application of security research to meet practical user requirements, both in the paper selection process and in the invited speeches.

Acceptance into the conference proceedings was very competitive. The Call for Papers attracted more than 120 submissions, out of which the Program Committee selected only 35 papers for inclusion in the proceedings.

This conference was made possible only through the contributions from many individuals and organizations. We would like to thank all the authors who submitted papers. We also gratefully acknowledge the members of the Program Committee and the external reviewers, for the time and effort they put into reviewing the submissions.

Special thanks are due to Ying Qiu for managing the website for paper submission, review and notification. Patricia Loh was kind enough to arrange for the conference venue, and took care of the administration in running the conference.

Last but not least, we are grateful to the Institute for Infocomm Research, and also the School of Information Systems, Singapore Management University for sponsoring the conference.

February 2005

Robert H. Deng,
Feng Bao, HweeHwa Pang,
Jianying Zhou

# ISPEC 2005

## First Information Security Practice and Experience Conference

## Singapore
## April 11–14, 2005

*Organized by*

Institute for Infocomm Research, Singapore

*Sponsored by*

Institute for Infocomm Research, Singapore
*and*
Singapore Management University, Singapore

David Naccache ..... Gemplus, France
Masahiro Mambo ..... Univ. of Tsukuba, Japan
Chris Mitchell ..... Univ. of London, UK
SangJae Moon ..... Kyungpook National Univ., Korea
Reihaneh Safavi-Naini ..... Univ. of Wollongong, Australia
Kouichi Sakurai ..... Kyushu Univ., Japan
Ravi Sandhu ..... GMU, USA
Shiuhpyng Shieh ..... NCTU, Taiwan
Dawn Song ..... CMU, USA
Dan Suciu ..... Univ. of Washington, USA
Rahul Telang ..... CMU, USA
Vijay Varadharajan ..... Macquarie Univ., Australia
Victor Wei ..... Chinese Univ. of Hong Kong, China
Moti Yung ..... Columbia Univ., USA
Jianying Zhou ..... I2R, Singapore

**External Reviewers**

Issac Agudo, Joonsang Baek, Lujo Bauer, Eric Brier, Julien Brouchier, Kisik Chang, C.I. Chen, Shiping Chen, Xi Chen, Benoit Chevallier-Mames, Eun Young Choi, Jean-Sebastien Coron, Guerric Meurice de Dormale, Y.J. Fu, Juan Gonzalez, Huiping Guo, Helena Handschuh, Yvonne Hitchcock, Yoshiaki Hori, Shih-I Huang, Changho Jung, Lea Kissner, Caroline Kudla, Anantharaman Lakshminarayanan, Fu-Yuan Lee, Kwangsoo Lee, Zhou-Yu Lee, Feiyu Lei, Shiqun Li, Tieyan Li, Xiangxue Li, Ya-Jeng Lin, Becky Liu, Changshe Ma, Jose A. Montenegro, James Newsome, Jose A. Onieva, Alina Opera, Pascal Paillier, Joseph Pamula, Young-Ho Park, Kun Peng, Angela Piper, Kyung-Hyune Rhee, Rodrigo Roman, W. Shin, Yuji Suga, Toshihiro Tabata, Yoshifumi Ueshige, Lionel Victor, Guilin Wang, Lingyu Wang, Shuhong Wang, Claire Whelan, Hongjun Wu, Hsiao-Chan Wu, Yongdong Wu, Yi Xu, G.M. Yang, Tzu-I Yang, Jungjae Yoo, Kee-Young Yoo, T.H. Yuen, Ruishan Zhang, Xuan Zhou, Bo Zhu, Huafei Zhu

# Table of Contents

## Network Security



## Cryptographic Techniques I

## Secure Architecture I



## Access Control



## Intrusion Detection

## Applications and Case Studies



## Secure Architecture II

## Data Security



## Cryptographic Techniques II

# Risk Assessment of Production Networks Using Honeynets – Some Practical Experience

Stephan Riebach, Erwin P. Rathgeb, and Birger Toedtmann

Computer Networking Technology Group
Institute for Experimental Mathematics and Institute for Computer Science and Business Information Systems, University Duisburg-Essen
{riebach, erwin.rathgeb, btoedtmann}@exp-math.uni-essen.de

**Abstract:** Threats for today's production networks range from fully automated worms and viruses to targeted, highly sophisticated multi-phase attacks carried out manually. In order to properly define and dimension appropriate security architectures and policies for a network, the possible threats have to be identified and assessed both in terms of their impact on the resources to be protected and with respect to the probability and frequency of related attacks. To support this assessment, honeynets, i.e. artificial networks set up specifically to monitor, log and evaluate attack activities, have been proposed. In this paper, experiences and results gained with setting up, deploying and operating such a honeynet are reported together with some comments on the effectiveness of this approach.

## 1 Introduction

It is well known that today's networks are subject to numerous threats ranging from blind, automated worm and virus attacks via prefabricated "standard" attacks by using readily available exploits to highly sophisticated, expert attacks. It is also obvious that securing a network involves an intelligent tradeoff between cost (in terms of equipment, expertise, usage restrictions and manpower) and the required level of security. To properly balance this tradeoff, current data on types, frequency and impact of attacks is required. Although some general information on worm and virus threats as well as on known system vulnerabilities is readily available, more specific information related to the customized mix of hardware, operating systems and software used in a network is difficult to obtain.

Larger networks are typically secured by using firewall systems filtering out "evil" packets and traffic on the network, transport and application layer. In addition, Intrusion Detection Systems (IDS) are typically deployed as additional safeguard to detect attacks and anomalies behind the firewalls. Reports and log files from these systems can provide some information on the frequency of attacks. However, since their purpose is to suppress any suspicious activity as early as possible to protect the production network and its data, this information is clearly biased as, e.g. multi phase attacks are blocked at an early stage. To some extent, IDS log information reveals

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 1 – 12, 2005.

also the type of attack. However, as a typically rule based system, e.g., the IDS can only recognize known patterns. Moreover, these logs provide only a limited basis to correlate different attack activities.

Therefore, it has been proposed to set up dedicated, artificial networks, called honeynets [1,1a], specifically for the purpose of monitoring, observing and analyzing malicious activities. Since honeynets are typically not hidden behind firewalls and are tightly controlled with all activity logged on packet level, they provide an unbiased view of the threat situation and at the same time allow performing in depth forensic analysis offline. Since honeynets don't have real users and, thus, don't hold information that has to be protected, malicious activity can be allowed in a controlled way to be able to observe the impact of successful intrusions.

In order to gain practical experience with the honeynet approach, we have implemented and deployed a honeynet. We have operated it over a period of four months and have used it for gathering detailed statistical data on attack activity on one hand and for in depth forensic analysis of specific attacks on the other. This paper summarizes our experiences with respect to the effectiveness of the honeynet approach and also highlights some of our findings.

## 2 Generic Honeynet Architecture

The term "honeynet" was coined by a group of security experts organized in the "Honeynet Project" (www.honeynet.org). This group promotes the development and application of honeynet concepts and is the main source of the definitions used in this section.

The basic idea that led to the development of honeynets was to detect, observe and document the activity of hackers inside a computer network. Therefore, honeynets are highly specialized networks which have to be kept strictly separate from the actual production networks, have no real users – and thus no real traffic activity – and don't contain any real information (user data). To be able to observe attacks, honeynets have to be vulnerable to a certain extent which means that they cannot be strictly protected by firewalls and that their systems should at least show some of the common vulnerabilities. Honeynets are highly controlled which means that elaborate monitoring and logging facilities capture and document all activity to provide comprehensive data for forensic analysis. All honeynets are "artificial" in a sense that they have no real users and, therefore, no real traffic. Therefore, all traffic in a honeynet is per definition suspicious and traffic originating from a host in a honeynet is an indication that this system has likely been taken over.

In addition to the "honeypot" computers to be scanned, probed or attacked, a "data capture" function is required to make the honeynet useful. In addition to storing all data packets for offline forensic analysis, online monitoring with host and network based Intrusion Detection Systems (IDS) is useful to provide immediate notification about ongoing attacks as well as a basis for targeted forensic analysis. The data capture function can be distributed among several computers or concentrated in a centralized device. Because honeynets are intentionally vulnerable, so called "data control" mechanisms must be implemented to ensure that intruders can not misuse compromised honeypots for further attacks. There are several ways to perform data

control, e.g., limiting the outgoing bandwidth, restrictive outgoing packet filtering or, adding packet loss and high delays to outgoing connections [1,1a]. It is particularly important that only the honeypots are visible and accessible for intruders. Therefore, the data control and capture functions have to be hidden from intruders in order not to reveal the honeynet character of the network. In addition they have to be protected against any manipulation.

The honeynet concept has evolved significantly over the past few years, in particular with respect to implementing data capture and control functions [1,1a]. There is a broad spectrum of realization options for honeynets ranging from software emulating specific aspects of operating systems, applications and services (e.g. Honeyd, see www.honeyd.org) to real networks with hosts providing real services and applications. Whereas simple emulations allow only limited interaction; honeynets with live systems allow full interaction. However, the latter ones require significantly more effort for setup, configuration and maintenance.

## 3 A Practical Honeynet Setup

In our project, the honeynet architecture shown in Fig. 1 was used with 5 honeypots connected via a 100baseT hub. The honeynet was connected to the internet via a router (dual homed Linux machine) providing the data control function. We used packet filtering and additional bandwidth limitations for the outgoing traffic.

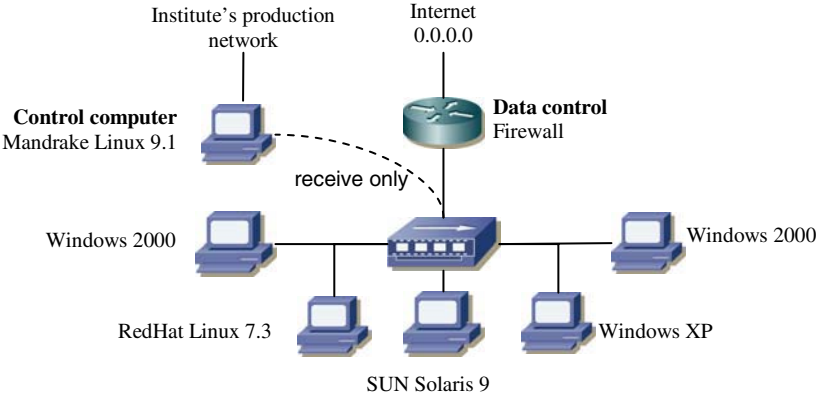



**Fig. 1.** Honeynet setup for this case study

### 3.1 Configuration Aspects

The control computer was set up with two network interfaces. A modified network cable was used to connect one of these interfaces to the honeynet in receive only mode [2]. Therefore, it would have been fairly impossible also for expert intruders to detect the presence of this machine and subsequently to attack it. Furthermore, the log

data and reports collected on this machine could not be modified from the honeynet avoiding that attackers could cover their tracks. Due to these precautions, it was possible to connect the other interface of the control computer to a production network for maintenance and remote data retrieval. The control computer was responsible for monitoring and capturing all network traffic in the honeynet. For traffic monitoring, the Network Intrusion Detection System (NIDS) "SNORT" [3] was installed to identify known attack signatures in the honeynet traffic continuously and in real time. The SNORT log files were automatically archived once a day and automatically processed locally on the control computer by "SnortSnarf" [4] which produced formatted statistical output. These statistics presented in HTML were automatically published on a web server on the control computer and could be remotely inspected from the production network. In addition, the major statistics files were automatically sent to the honeynet operator once a day. For data capturing the software "tcpdump" [5] was deployed. With tcpdump all data traffic occurring in the honeynet was saved into daily dump files including all protocol overhead (addresses, etc.) from OSI layer 2 upwards. To assure the permanent availability of these vital honeynet components, SNORT and tcpdump were monitored by using "Daemontools".[6]

As indicated in Fig. 1, two of the honeypots were configured with the Windows 2000 operating system, one with Windows XP, one with RedHat Linux 7.3 and the last one with Solaris 9. This mix of operating systems was chosen because it is fairly typical for our production networks. With respect to the vulnerability of the honeypots we updated to a patch level which was fairly typical for an environment where there is only limited central administration of the systems and the users have to take responsibility for their systems themselves. All honeypots were equipped with Host Intrusion Detection Systems (HIDS). Since the honeypots were not accessible remotely from the production network for security reasons, the log files of all HIDSs were collected manually in regular intervals. Complete images of the software installations of all honeypots were saved. Therefore, a compromised system could be restored to its original state with minimum effort. Before cleaning up a compromised system, we also saved a complete image for offline analysis and possible reinstallation for further observation.

### 3.2 Deployment Aspects

In order to allow for unbiased measurement of the Internet, our honeynet was located outside the firewalled university network. Since the first day the honeynet was running, activity could be detected confirming the statement [7] that a honeynet will be found and attacked without further actions needed.

In order to find out how the attractiveness of the honeynet can be influenced by its configuration, we carried out a phased deployment study [8] increasing the visibility of the honeynet in every step. The first observation was that the attack frequency doesn't significantly depend on the lifetime of the honeynet, i.e. it neither increases because the network becomes known over time nor decreases because it has been identified as honeynet. As a consequence, no significant "warmup" phase seems to be

required before starting statistical measurements. The most significant increase in attack frequency which could be clearly attributed to a configuration change was observed after the full activation of a DNS server making the network fully visible in the Internet. This effect can be attributed to the fact that DNS lookups are used by the spreading mechanisms of internet worms as shown in section 4.1. Activation of various popular services (http, ftp) on the honeypots did increase the frequency of non-automated attacks as well as their diversity (cf. section 4) with the result that all services provided were eventually attacked. However, the attacks were still unspecific in a sense that a significant part of the attacks on the web server were targeted towards the Microsoft IIS although only an Apache server was running. Actively generating traffic by having honeypots participating in a P2P file sharing network (providing fake content only) proved to be counterproductive. The P2P search and download processes produced an enormous amount of data traffic, but only limited correlation could be detected between the P2P traffic and attack signatures. None of the IP addresses used in the P2P communication was involved in any non-P2P signature. Furthermore there was no temporal correlation between attacks and P2P traffic; also the overall attack frequency didn't increase significantly. From our study it can be concluded that making the honeynet known in the DNS and providing a range of popular services on the honeypots is useful whereas actively generating traffic is not worthwhile and also clogs the log files with irrelevant data.

### 3.3 Operational Aspects

A honeynet is a highly specialized instrument to detect, observe and analyze attacks in detail. To produce meaningful results, honeynets require daily administration and maintenance. During normal operation, the honeynet generated at least 1 Mbyte of SNORT log data and 75 Mbyte (average) of tcpdump logs per day. This raw data was completely archived for statistical and forensic analysis. The statistical evaluation was highly automated as described above.

The portscan log files were transformed into the CSV-format for detailed analysis in MS Excel. The automatic formatting and publishing of the SNORT logs allowed for a quick inspection and gave indications about potentially successful attacks which were then followed up. In addition, the HIDSs of the honeypots had to be collected and inspected on a daily basis so a compromised honeypot could be identified rather quickly. Furthermore, sporadic in depth control and analysis of the honeypots was necessary to minimize the probability of undetected attacks. These routine tasks amounted to a maintenance effort ranging from a minimum of one hour up to several hours per day.

A manual forensic analysis was performed in several cases where successful (non-automated) attacks could be detected. For manual inspection of the tcpdump log data, the program "tcptrace" [14] was used to identify successful TCP connections related to successful attacks. "Ethereal" [15] was then used to fully decode the packets of interesting connections up to the application level. Due to the size of the logs and the need to scan several of them for various attributes, e.g. specific IP addresses, this was a rather significant and time consuming effort.

## 4 Results Relevant to Risk Assessment

Statistical analysis of attack activity was mainly based on the SNORT log files. SNORT classifies attack signatures into severity levels. In the following we distinguish between

- Alarm: dangerous and harmful attacks (SNORT priority 1)
- Warning: suspicious signatures potentially preparing attacks (priority 2)
- Notice: unusual traffic not identified as dangerous (priority 3)

As Fig. 2 shows, there was no obvious correlation between the number of alarms and the number of warnings. From this we concluded that the majority of attacks were blind attacks which were not being prepared by intensively scanning and probing the network first.

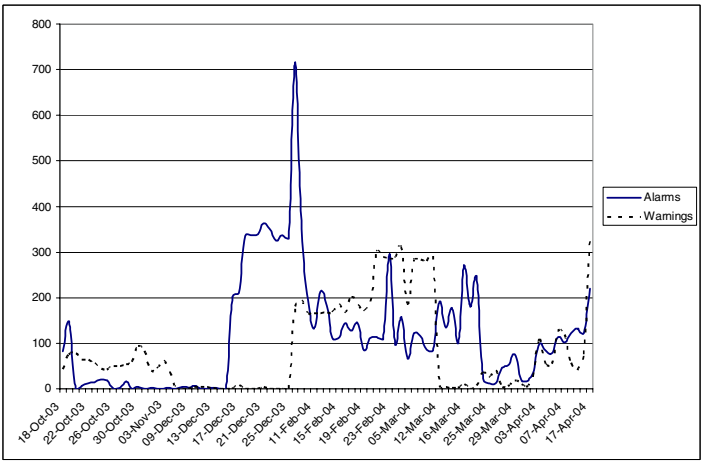



**Fig. 2.** Number of alarms and warnings during the study

When analyzing the targets of attacks, we found that nearly 97% of all attack signatures were specifically targeted towards the Windows systems. This was not unexpected since it makes sense to concentrate the effort to develop attacks on the clearly dominating operating systems. However, the obvious conclusion that windows systems are significantly more threatened would be misleading here, because a more detailed analysis of the alarms showed that 81.25% of all alarm signatures were of the type "NETBIOS DCERPC ISystemActivator bind attempt" [16]. This signature is generated by an attack against Microsoft's DCERPC interface on port 135/tcp and can clearly be linked to worm spreading mechanisms. As a consequence, it is sufficient to apply one single patch to the Windows systems to counter the vast majority of attacks and bring the remaining attack frequency into the range observed for other operating systems. During the study period, we didn't identify any worm related activity targeted towards non-windows systems. However, even if the attack frequency

towards the other operating systems was still lower after filtering out worm activity, attack sophistication seemed to be even higher and also resulted in successful takeovers. The Solaris honeypot, e.g., was compromised by a classical multi-phase attack eventually exploiting a known vulnerability of the "cachefsd" service.

After filtering out the worm activity from the analysis, the impact of providing increasingly more services on the honeypots on the attack frequency became obvious as shown in Figure 3. Furthermore, a more significant correlation between alarms and warnings becomes visible indicating a higher share of more elaborate, multi-phase attacks.

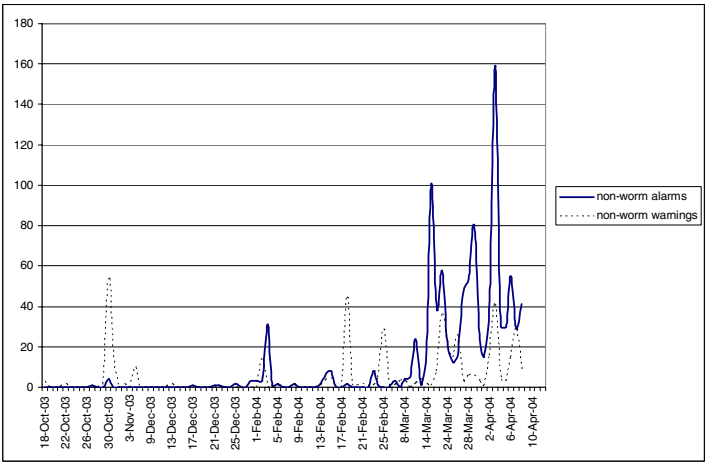



**Fig. 3.** Non-worm related attack activity

### 4.1 Analysis of Worm Related Activity

After identifying worms as the dominant source of attacks, a more detailed evaluation could be performed to establish correlations providing some insight into the mechanisms involved in worm activity. We also analyzed the priority 3 notices logged indicating unusual packets not identified as dangerous, like ICMP PING's, and their correlation to worm related alarms as shown in Figure 4. Since Snort differentiates ICMP ECHO packets by their generating operating system, we found that 95% of all ICMP ECHO packets were generated by Windows systems. Because of the high volume of these ICMP scans in the first half of the study period, we assumed that they were directly related to worm activity.

However, this scanning activity only resulted in a corresponding increase in the frequency of actual attacks starting on Dec. 16$^{th}$ 2003 with an instantaneous rise. This coincides explicitly with the moment at which we configured the local DNS server in the honeynet to perform DNS reverse lookup. This indicates that the Windows worms produced an enormous amount of ICMP packets to identify possible targets, performed a DNS lookup for the identified IP addresses and then attack the systems.

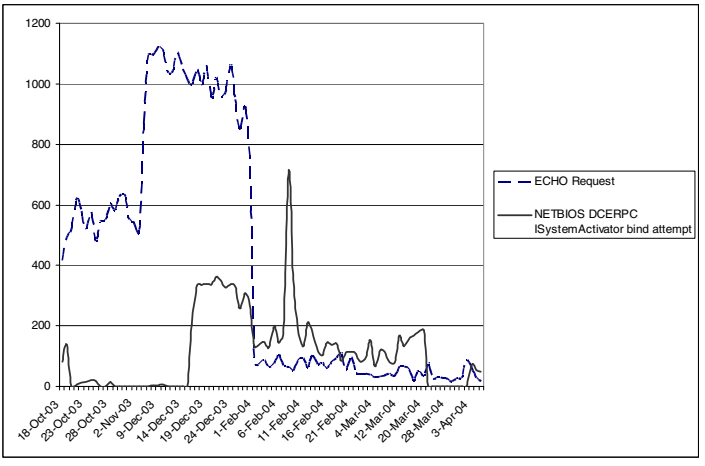



**Fig. 4.** ICMP PING's and Netbios attacks

### 4.2 Worms, IRC, Bot Networks and Spam

It is well known that worm programmers often share their code and, therefore, new worms and their variants are implemented reusing code fragments from older worms thus trying to exploit the same vulnerabilities [10]. It is also known, that worms may inject malicious code into the target system providing, e.g., backdoors or IRC tools. After having detected a successful infection of a Windows 2000 honeypot, we decided to perform a controlled observation of the worm and its activities.

#### 4.2.1 Infection and Worm Identification

The infection was first discovered due to automatic, continuous monitoring of outgoing traffic when the local honeypot tried to establish an outgoing TCP connection on January 20, 2004 to the external host 66.98.168.222 on port 6667/tcp which can be linked to IRC applications [17]. This was a clear indication that the honeypot was compromised, since no applications originating external traffic had been activated by us. As several hundred of these connection attempts were reported that day, an automated activity was assumed. The honeypot was disconnected and an abortive shutdown was performed to avoid deletion of evidence which could be caused by scripts executed during normal shutdown. A complete image of this machine was also created for forensic analysis. Offline comparison with the image of the original system revealed two new executable files in the system folder `c:\windows\system32\` (Metalrock-is-gay.exe, Musirc4.71.exe). The "poor man's tripwire system" [21] used as HIDS on the compromised host also indicated two new registry keys in the "HKEY_LOCAL_MACHINE\SOFTWARE\" branch after offline reboot. After some internet research and the application of McAfee's tool Stinger [18], the W32.Randex.Q worm could positively be identified. Since Randex.Q is an IRC backdoor worm [19] we decided to reconnect the honeypot to observe subsequent activities.

### 4.2.2 Entering a Bot Network

The worm remained inactive for nearly two weeks after reactivation of the system, but on February 15, 2004 the worm successfully contacted a new IRC server 216.65.117.114 on port 6667/tcp indicating that there is a list of alternative servers implemented in the worm's code. After this successful connection attempt, the worm did not change the server address any more. In the sequel, connection attempts occurred after seemingly random time intervals and four states of the worm were observed:

- State W1: inactive and not connected
- State W2: active and attempting to connect
- State W3: active and waiting
- State W4: active, connected and executing or receiving commands

Based on the worm's behaviour, it became obvious that IRC server did not accept connections continuously and three states oft the server could be derived as follows:

- State S1: inactive
- State S2: active but blocked (refusing connections)
- State S3: active and accepting connections

After completing a TCP 3-way handshake with an active server, the worm tries to log into the IRC server. The server sends the message "`Error: all connections in use`" if it is in state S2 and resets the connection. If there are available connections (S3), the server sends an IRC reply and the worm tries to enter an IRC channel. We always detected the channel called "fear" on the server brazilchat.com. The worm program reacted on the server's states. First the worm tried to connect, changing its state from inactive (W1) to connecting (W2). If the server was inactive (S1) or blocked (S2), the worm changed its state to active and waiting (W3), which was observable by periodically attempts to connect. It is important note that the spreading mechanism of the worm program as well as the connection procedure in States W1 to W3 was a fully automated procedure whereas human interaction occurs in State W4. By connecting to the IRC server, the compromised honeypot was included into a so called "bot network" automatically. "Bots" (also called mobile agents) are executable programs sent out to perform specific tasks on remote computers for the controlling user or process [12, 13]. Bots are not by definition malicious, the same concept is also used by search engines. Bot networks are self-configuring application layer networks (similar to peer-to-peer networks) with servers allowing the bots to register. The servers also allow users to connect to the network and to access the currently active bots. Several known bot networks use the IRC protocol for communication to avoid blocking by firewalls and detection by IDSs. Thus bot networks provide a self-organizing pool of fully accessible computers across the internet which can be used, e.g. for "distributed denial of service" attacks. [13]

### 4.2.3 Remote Command Execution and Human Interaction

The Randex worm allows different commands to be sent to the bots. All commands are entered via a command shell using an IRC client tool. There are nearly 20 known

actions and commands for the Randex worm. The most commonly used commands are “ntscan” which starts the spreading mechanisms by scanning a list of randomly generated IP addresses (or a specific computer) with massive SYN-floods and “sysinfo” which sends all important system information (RAM-size, processor type, hard disk size, etc.) of the local system to the IRC server. While the commands sent from the server towards the worm were encrypted, the worm’s replies were plain text such that the activities could be identified. On the same day when the honeypot entered the bot network, it first started a massive TCP port scan on port 445 scanning 4477 IP addresses in the Class-B network of the university (which was blocked by the firewall between the honeynet and the university production network). Subsequently, we observed four similar activities as indicated in Table 1.

**Table 1.** Remote TCP port scans with the ”ntscan“ command

| Scans with “ntscan“ | scanned IP addresses |
|---|---|
| Feb. 15, 2004 | 4477 |
| Feb. 24, 2004 | 4481 |
| Mar. 6, 2004 | 4751 |
| Mar. 8, 2004 | 1134 |
| Mar. 9, 2004 | 4376 |

On March 6, 2003 an attacker used the bot network to download a file called “win.exe” using the http protocol and to install it. The remote host was a corporate web-server (probably hacked) and the file was available there for just a few days. Again, the command for downloading was encrypted but was revealed by the worm’s plain text answers. The file “win.exe” was activated on March 11, 2004 and turned out to be an email-spam client. On this day we recorded many thousands of connection attempts to public mail servers on port 25/tcp. Overall the client established 28244 connections attempts to 6228 mail servers in less then two hours. While the client was still connecting, we turned of the honeypot and disconnected it.

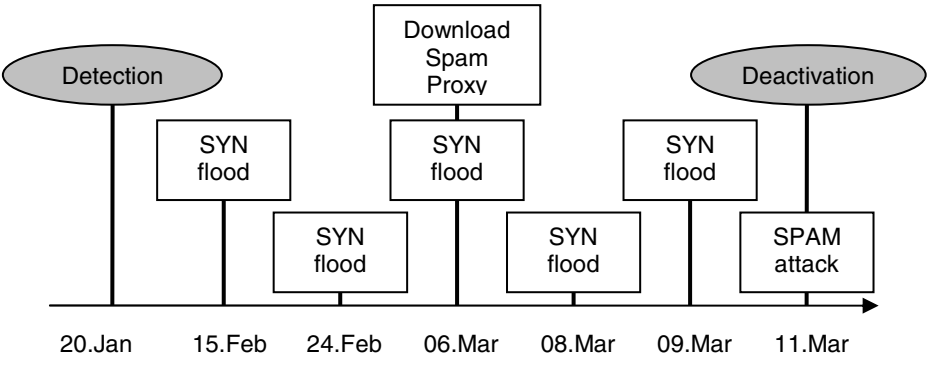



**Fig. 5.** Timeline of the worm infection and spam activity

Figure 5 summarizes the activities during the observation with the white boxes indicating the automated or remotely triggered processes. With that observation we were able to confirm the connection between worms, bot networks and e-mail spam reported around the same time [20]. These findings clearly confirm that attackers can gain control over a large number of "Zombie-PCs" with minimum effort and to flexibly use them for a wide range of (malicious) activities which do not have to be preconfigured into the worm software. The attack against Akamai Inc. [9] has demonstrated that this kind of threat is becoming increasingly virulent.

## 5 Conclusion and Outlook

In this paper, we reported some practical experience gained from setting up, deploying and operating a honeynet over a period of nearly 6 months. We found it feasible to create a fully functional honeynet including honeypots with the most popular Windows and Unix/Linux operating systems by using freely available software components only. The most critical point here was the HIDS, because different tools had to be used for the various operating systems as there is no single one available as free software for all systems.

A study conducted confirmed that a honeynet connected to the internet will be found and attacked immediately. We found that the frequency and diversity of attacks can be influenced to a certain extent by the configuration of the honeynet. Some common automated worm attacks, e.g., require the operation of a DNS server. Non-automated attacks, on the other hand, can be stimulated to some extent by providing some of the commonly used services on the honeypots. From the operational point of view, the honeynet required regular daily maintenance and control despite the attempts to automate routine tasks as much as possible. This should be considered before deciding for a honeynet. Since the majority of attack activities is related to fully automated, identical worm attacks, we are currently investigating methods to filter out these events selectively to reduce the volume of log files in order to also reduce the analysis effort. Another approach to reduce operation and management effort – and to make the honeynet more portable at the same time – is to implement the virtual honeynet concept. We currently evaluate this option, where several virtual honeypots are emulated on one physical machine. Despite the significant effort that has to be spent to set up and operate the honeynet, it proved to be a useful means for qualitative and quantitative risk assessment. Gathering, statistical evaluation and presentation of basic information on frequency and type of attacks could be obtained in a fully automated way. Evaluation of logged attacks according to various criteria, e.g., target operating system, target address could easily be performed as required allowing to classify malicious activities and to identify existing correlations among them.In addition, detailed forensic analysis could be successfully performed in several cases by using the comprehensive IDS and packet level logs in combination with information readily available in the internet. Furthermore, also controlled online observation of attack activities in progress was feasible and provided considerable insight into the impact, potential consequences and correlations of those activities.

As a consequence, we will continue to improve our honeynet implementation with the primary goals to reduce operation and maintenance effort and to make it more easily configurable for selective observations.

## References


[1] The Honeynet-Project: “Know Your Enemy: Revealing the security tools, tactics, and motives of the Black Hat community“, Indianapolis: Addison-Wesley, 2001, http://project.honeynet.org

[1a] The Honeynet-Project: “Know Your Enemy : Learning about Security Threats”, Indianapolis: Addison-Wesley, 2004

[2] Building a “sniffing cable” by IronComet Consulting http://www.ironcomet.com/ sniffer.html

[3] Roesch, Marty; Caswell, Brian: “Snort’s official homepage”, http://www.snort.org/

[4] Official homepage for “Snort Snarf“, http://www.silicondefense.com/software/ snortsnarf/

[5] Official homepage for “tcdump”: http://www.tcpdump.org/

[6] Bernstein, D.J.: Daemontools Homepage, http://cr.yp.to/daemontools.html

[7] The Honeynet Project: “Know your enemy: Statistics”,White paper http://project. honeynet.org/papers/stats/, July 23 2001

[8] “Efficient deployment of honeynets for statistical and forensic analysis of attacks from the Internet”, by the authors of this paper, Oct. 10 2004. submitted to: International Conference on Networking, ICN'05

[9] Lemos, Robert: “'Zombie' PCs caused Web outage”, June 17 2004, http://asia.cnet.com/ news/security/0,39037064,39183708,00.htm

[10] Well-known security vulnerabilities in MS Windows 2000/XP: http://www.microsoft. com/technet/security/current.aspx

[11] BotSpot, collection on bots and agents, http://www.botspot.com/common/whats_bot.html

[12] McLaughlin, Laurianne: “Bot Software Spreads, Causes New Worries”, IEEE Distributed Systems online 1541-4922, Vol. 5, No. 6; June 2004

[13] Puri, Ramneek: “Bots and Botnet – an overview”, Aug. 08 2003, http://www. giac.org/practical/GSEC/Ramneek_Puri_GSEC.pdf

[14] Ostermann, S.: “Tcptrace Official Homepage” http://jarok.cs.ohiou.edu/software/ tcptrace/

[15] Official homepage for “Ethereal”, http://www.ethereal.com

[16] Snort signature reference file, SID 2192, http://www.snort.org/snort-db/sid.html? id=2192

[17] Overview on the IRC protocols and RFC’s, http://www.irchelp.org/irchelp/rfc/

[18] Homepage of McAfee’s stinger tool, http://vil.nai.com/vil/stinger/

[19] Worm description of W32.Randex.Q by Symantec, Oct. 03 2003: http://security response.symantec.com/avcenter/venc/data/w32.randex.q.html

[20] c’T-magazin: “Trojans as spam robots”, Feb.21, 2004 http://www.heise.de/ english/ newsticker/news/44879

[21] “A poor-man Tripwire-like system on Windows 9x/NT” http://www.geocities.com/ floydian_99/poormantripwire.html

# POSSET – Policy-Driven Secure Session Transfer

Philip Robinson[1], Christian Schaefer[2], and Thomas Walter[2]

[1]SAP, Vincenz-Priessnitz-Str. 1, 76131 Karlsruhe, Germany
Philip.Robinson@sap.com
[2]DoCoMo Euro-Labs, Landsberger Str. 312, 80687 Munich, Germany
{schaefer, walter}@docomolab-euro.com

**Abstract.** Ubiquitous networks and seamless terminals are potential enablers for session mobility and session transfer. In a business environment, session mobility is restricted by the security requirements set forth by corporate security policies to protect corporate assets. Session mobility can be supported to the extent that specified corporate assets are still protected even though a session is transferred to another mobile device. We describe a policy-driven approach for secure session transfers. Secure session transfer mechanisms validate whether or not a session transfer is allowed, establish secure interaction channels with target devices, perform security context negotiation and, if all previous steps are successful, facilitate transferring a session from a source to a target device. The protocol is supported by security policies and digitally signed assertion tokens. Policies define the constraints to be met before (i.e. decision whether transfer is possible or not) and after session transfer (i.e. respective security context.), while tokens are utilized to identify suitable mobile devices that claim trustworthiness, which may be target of a session transfer.

## 1 Introduction

Ubiquitous networks can be identified by the following properties [6]:

- *Seamless wireless access*: users can receive services of desired grade on a continual basis, without worrying about the differences in wireless access systems.
- *Seamless terminal*: users can receive services on a continual basis through the most suitable terminal according to time, place, preferences and other conditions.

Seamless wireless access enables intra- and inter-technology transfers, e.g. transfers between WLAN access points (intra-technology) as well as between WLAN and GSM/3G networks (inter-technology). Seamless access paired with ubiquitous communications establishes a computing platform where users can select from a choice of devices and can experience services tailored to their specific needs and for best performance (= seamless terminal). Seamless wireless access, ubiquitous communications and seamless terminal together are required to support service mobility, i.e. one service on several devices and across several domains [9].

We refer to a service as application software in an information system, which offers a specific set of functions to users by a provider through a communications net-

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 13 – 24, 2005.

work. [4]. It fulfils a special purpose for the user abstracting from the pure system capabilities. A session describes the meaningful context of one executed service, i.e., the coordination of all tasks and information that are common to all processes of a service execution [5]. Session mobility refers to the transfer of a session from the source mobile device to another, target mobile device(s); session transfer is the procedure implementing session mobility [9]. As an example of a session and session transfer, one may think of a mobile worker who starts a business task on a PDA and decides to transfer the running business task to his or her laptop for more convenient processing of large spreadsheets or text documents. Such scenarios were explored in the Witness project [11], to which the authors contributed.

While today there have been many advanced systems that support network and security administrators in - even remotely - configuring, monitoring and responding to changes in the security environment of extensive networked information systems, a common problem they face with mobility is the capability of users to work completely offline with respect to the central domain controller. Users may need to remotely connect to the information system over insecure public networks, and may also use intermediary and auxiliary devices, for transferring sessions to them, for supporting their work, which are outside of the control range of the system administrators. In addition, as developers and system administrators normally work in separate departments and phases of system development, there is no coupling of application functionality (including the mechanisms for transfer of the functionality) and security enforcement. Utilizing this decoupling by local policy enforcement based on the enforcement of security policies prescribed by the central, remote administration, we therefore sought to specify and implement a realistic mechanism to transfer sessions that still allows users the freedom of mobility yet assures administrators that application sessions continue to be secured.

Corporate assets should be protected against threats such as disclosure, unauthorized modification and loss due to physical theft of device, unattended application sessions and breaches in communications integrity. Provision of respective security services and mechanisms is done by corporations in order to achieve the required degree of protection. The degree of protection and security measures are defined in enterprise security policies. With each of mentioned elements, security services are associated and utilized in the enterprise network and on mobile devices. However, the enforcement of these policies is still dependent on the diligence and due care of the human end-users, who in many cases neglect the policy (this may include routine things such as “frequently download operating system updates”), such that with an application capability such as session transfer, the need for an enterprise to programmatically mitigate end-user interactions is critical.

In this paper we define a policy-driven approach for secure session transfers. Our session transfer protocol is outlined as follows: first, security policies are checked whether a session transfer can be allowed or not. Second, the applicable security constraints for the session are determined as expressed in terms of security policies, while the resultant security constraints are negotiated with the target mobile device. The described secure session transfer is instrumented by specific security policies and employs signed message tokens for proof of assertions and integrity. All elements are integrated into a prototypical software framework, which contributes the following concepts:

- A framework where security requirements are derived from the needs of business applications;
- Mechanisms for negotiation of security constraints between source and target devices;
- The establishment of secure interaction channels for exchange of session state
- The performance of secure session transfer in a flexible manner by taking security policies into account.

The paper proceeds with Section 2, which identifies related work on session signalling and session transfer, as well as security policy work. Section 3 describes the assumptions and environment within which secure session transfer is completed. In Section 4 we provide an in-depth description of our session transfer protocol, placing the focus on the steps of the session transfer protocol that specifically deal with policies, security requirements and security negotiation.

## 2 Background and Related Work

A session can be determined as the "meaningful context of one executed application, i.e., the coordination of all tasks and information that are common to all processes of one application execution" [5] or in terms of [3], we understand session as the state of an application process which is given by data structures and variables needed. In addition to these process-oriented definitions, we find in the literature definitions that motivate a session by considering communication services; e.g. in [10], a session is defined as "a set of multimedia senders and receivers and the data streams flowing from senders to receivers". In this paper we use the first definition of session as our main reference.

There is a range of work in the area of session transfer; The IETF Session Initiation Protocol (SIP) [13] is basis for some approaches like in [12] where three ways for a session transfer are introduced. It is mainly concerned with signalling for the transfer of streaming sessions from one device to another: the data stream to one device is interrupted and sent to the new device. It does not preserve the state of the application and transfers it.

The iMASH [2] architecture, on the other hand, is concerned with the transfer of an application state. It is a middleware approach to support session transfer with an Application Server (AS) that is responsible for transferring the session from one device to another. A security architecture [3] has been defined that supports end-to-end security through encrypting data at or above transport layer during session transfer. Furthermore authentication and authorisation are introduced to control the access to iMASH devices. This architecture is an improvement to other approaches as security considerations are taken into account. Although it secures the session transfer itself, it lacks the flexibility of our security policy-driven session transfer protocol as well as it does not deal with security requirements of the session itself.

Ponder [14] is a mature platform for policy research, including topics such as specification, deployment, refinement and enforcement. Many policy-related projects and frameworks, including WiTness, have done a specialization of the basic policies defined within the Ponder framework. The basic security-related policies are the Authorisation-Policy, which specifies what operations a subject is allowed to perform on

an object, and the Obligation-Policy, which specifies what operations a subject must perform on an object as a management duty. Subtypes of these base classes of policies include delegation, restraint and trust. Following the WiTness approach, the security policies are described at a lower level and the framework developed with the expectation that the environment consists of heterogeneous devices and networks.

A system for specifying, interpreting and invoking the mechanisms referenced by security policies underlies our proposed solution. Given a machine-interpretable syntax, policies are precisely specified and can be enforced by a supporting software framework. Enforcement in the context of session transfer implies that security constraints (defined by security policies) can be negotiated between mobile devices, such that a consistent set of security services and mechanisms can be invoked on the target device, consistent with the security services and mechanisms on the source device.

## 3 Mobile Business and Foundations of Secure Session Transfer

Consider a mobile business application using the network configuration in Figure 1, (1) where a session transfer is synchronized between an auxiliary device and a mobile device over a wireless network, in what we refer to as the "visitor environment", (2) a local preparation of the data is done on the mobile device, (3) a public network provider is used for contacting the corporate information system, (4) the corporate network enforces its network-level access controls, and (5) the target application servers complete the central update and submission of the synchronized transaction, in what we refer to as the "enterprise environment".

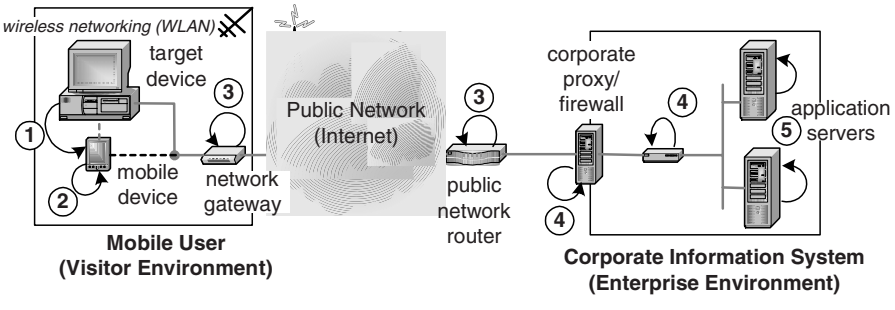



**Fig. 1.** A typical networking infrastructure for mobile business application, showing interaction decision points (1) to (5)

An interaction decision point is a set of rules that determine if interaction/communication with an identified principal (based on e.g. IP-address, content, credential, …) should be accepted or revoked.

The interaction decision points (3), (4) and (5) are therefore essentially handled by the state of the art in security management technology. However addressing the uncertainties of interaction decisions in the visitor environment - i.e. interaction decision points (1) and (2) - remain a challenge for administrators in terms of specification and enforcement. These are therefore the focus of the secure session transfer architecture and protocol.

### 3.1 Towards Secure Session Transfer

As a starting point for designing the architecture of the support system for secure session transfer, we first made some assumptions, based on the WiTness framework [11], about what should be available on the source device. These are summarised as the following:

**Mobile Application (MAP)**: the mobile application software that has been installed by an administrator of the corporate information system

**Application Manager (APM)**: trusted module that intermediates between applications and the operating system (e.g. a virtual machine)

**Application Data Store (ADS)**: a local database or file store accessible to applications via the APM

**Security Module (SEC)**: a library of cryptographic functions and utilities that support encryption/decryption, signing and verification of tokens, and authentication.

**System Properties (SYS)**: a dataset of modifiable system properties referring to attributes of the device and its environment e.g. locale, current user, processes…

**Communications (COM)**: a set of network protocols and interfaces that allow the device to at least communicate over a short range

Additionally, we define a particular order of internal event-interactions between these components within a device executing a secure session transfer, prior to transmission of the session to the target/ auxiliary device. This order of events is depicted in Figure 2 and subsequently outlined:

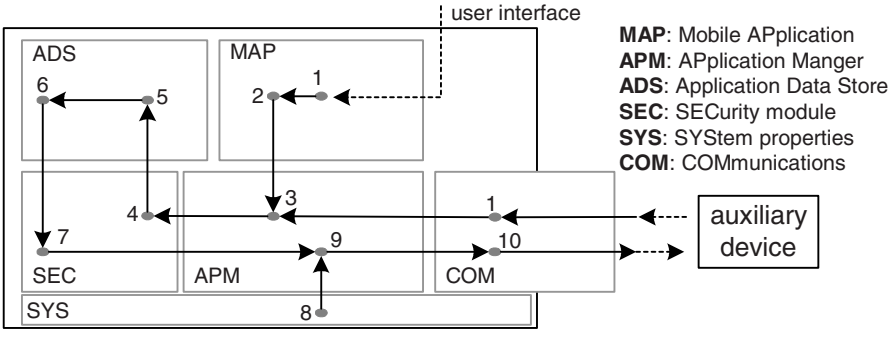



**Fig. 2.** Desired module and interaction diagram for interaction between modules in a device invoking secure session transfer

1. MAP initiates session transfer by specification of application functionality and data to be transferred, and signals the APM with the property required of the target auxiliary device. It is possible that an auxiliary device publishes its availability to the APM as well as part of a discovery process.
2. MAP requests the APM to gather the relevant application data for the session transfer based on a data-query

3. APM must first check the access to the application data by the current device user and the target auxiliary device. This is done by the SEC, based on both user and system credentials/ properties
4. The SEC can then permit access to the data elements in the ADS, allowing the APM or user to validate before transfer
5. The ADS then locates and collects the requested application data
6. Sensitive application data is encapsulated
7. The SEC informs the APM that the session transfer may proceed with the appropriate security settings
8. The SYS provides the current system properties to the APM
9. The APM makes a final configuration decision before initiating the transfer
10. The COM establishes a secure channel with the auxiliary device and transfers the session

Up to this point we have outlined what must occur on a "source device" prior to secure session transfer. Attention is now given to the interaction with the "target device" at the receiving end of the session transfer. Although each device should contain the modules defined above, there can be no assumption about their implementation details. It is therefore desirable to have a means of higher-level secure interaction specification, supporting negotiation between the source and target, as well as confirmation that the target is compatible with the source's specification.

## 4 Extending Session Transfer with Security Policies

Considering the above networking infrastructure, operational assumptions and functional goals, we have identified four types of security policies, which may be considered in security administration procedures: these are authorization, configuration, delegation and federation policies. An authorization policy is generally a "master policy" controlling access to network resources based on a claim by a requesting principal. These policies are typically specified using ACLs (Access Control Lists) or Role-based Permissions, or may be dynamically adjusted based on SLAs (Service Level Agreements). A principal may then claim authorization by proving possession of a verifiable token containing the accepted attributes in the ACL or permissions. The standards for generating, distributing, signing, verifying and revoking these tokens are a decision made by the system administration. There is therefore a need to configure all concerned machines with the appropriate utilities based on the accepted attributes and mechanisms. We have termed the set of rules and constraints governing the installation of a device as a "configuration policy", such that it is considered to be a derivative of the authorization policy. Configuration may also be concerned with utilities that monitor the environment within which claims to system integrity are made, which is an additional attribute of authorization/trust. Furthermore, as employees, business partners and customers tend to interact outside of the enterprise environment, they may share applications and rights to certain services or data. The policies governing how these rights are shared between principals are referred to as "delegation policies". These may also be encoded in the tokens – named authorization certificates – in terms of constraints on role interactions and number of delegations.

There is therefore a recursive validation incurred, in that a delegate must still have the appropriate authorizations before being delegated. Finally, delegation is in most cases more than the sharing of an application or device; it may include the interconnection of devices (similar to Figure 1), an assumed technical capability for session transfer, such that the properties of a delegate or federated device also need to be proven. The constraints on this device interconnection are referred to as a federation policy, such that there is a further logical dependency between configuration, delegation and federation. Figure 2 shows what we refer to as the "policy constraint chain", showing the constraint dependencies enforced between different policy types and the subsequent claims (*) that a requester will make based on a policy.

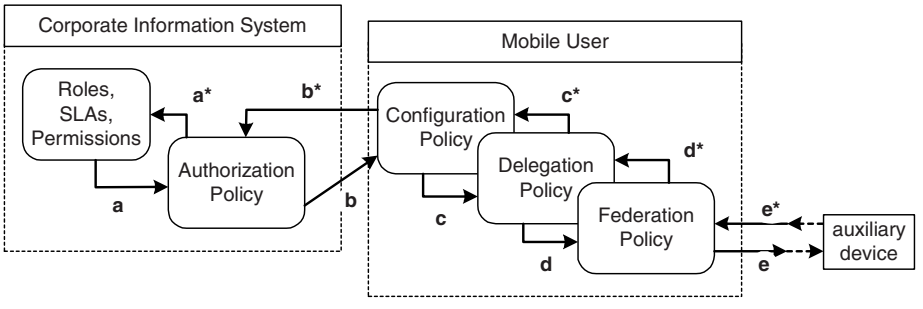



**Fig. 3.** Security policy decision constraints. {a, b, c, d, e} are a set of policy-based constraints while {a*,…,e*} is the set of corresponding policy claims

As mentioned previously, there is an assumed trust between the corporate information system, subsystems, employees and administrators within the physical boundaries of the enterprise environment. Therefore, we do not place focus on how authorizations are specified or enforced, in that systems and standards in that area are mature and increasingly dependable. Our focus remains with enforcing the constraints c, d, e and handling their respective claims c*, d*, e*, which are required in order to complete a secure session transfer.

### 4.1 Policy and Token Specification

Policies are specified rules of interaction while tokens are used to encode claims. To make the utility of the policies and tokens clearer, an example scenario of a mobile employee (Emp#1) making a presentation of a new product at a remote site is used. Emp#1 becomes ill and decides at the last minute to delegate the task to another employee (Emp#2), who is coincidentally also present at the remote site. There is currently no connection to the corporate network, such that the presentation and product data can only be transferred using a short-range wireless connection from Emp#1 to Emp#2. Emp#2 also decides to use the large screen available at the remote site in order to enhance the presentation. However, there are certain elements of product data that should not be transferred to the large screen. A policy-specification language has

therefore been defined for allowing an administrator in the corporate domain to specify these interaction rules, which only allows the interaction to proceed when the conformant claims/ tokens are supplied.

A single policy-template has been defined for all policies in order to keep the operational semantics simple. The policy specifications vary based on the type, namely: Configuration "C", Delegation "D" or Federation "F". A similar approach was taken when describing the assertion tokens, used as the mechanism of policy-compliance-proof within the secure session transfer protocol.

**Table 1.** Policy specification template and example based on scenario

| **POLICY-TYPE** | **"C"**(Configuration) | **"D"**(Delegation) | **"F"**(Federation) |
|---|---|---|---|
| **POLICY-ID** | A unique identifier for distinguishing the particular policy | | |
| **SOURCE** | Application or data | User identifier | Device identifier |
| **TARGET** | Application | User identifier | Device identifier |
| **ACTIVITY** | "read, write, delete, update…" | | |
| **SCOPE** | Config. constraints | Role constraint | Platform constraint |
| **ASSERTIONS** | Proof of config | Proof of role | Proof of platform |
| **ATTEMPTS** | Number of times policy claims can be made | | |
| **LIFETIME** | Expiry date of policy | | |
| **SIGNATURE** | Signature of policy creator and issuer | | |

```
* POLICY-TYPE: "C"; POLICY-ID: "C-ProductPresentation";
SOURCE: "ProductDB"; TARGET: "Presentation";
ACTIVITY: "read,write,view";
SCOPE: "Secured"; ASSERTIONS: {specified-crypto-suite};
ATTEMPTS: "2"; LIFETIME: "01122004";
SIGNATURE: sign(Company.Priv, hash(FEmp1)).
* POLICY-TYPE: "D"; POLICY-ID: "D-Emp1";
SOURCE: "Employees"; TARGET: "Employees";
ACTIVITY: "read, write, modify ";
SCOPE: "Management"; ASSERTIONS: {Emp-token};
ATTEMPTS: "2"; LIFETIME: "01122004";
SIGNATURE: sign(Company.Priv, hash(D-Emp1)).
* POLICY-TYPE:"F"; POLICY-ID:"F-Presentation";
SOURCE: "PDA001"; TARGET: "Any";
ACTIVITY: "read, write";
SCOPE: "Trusted"; ASSERTIONS: {Attestation-Key};
ATTEMPTS:"2"; LIFETIME: "01122004";
SIGNATURE: sign(Company.PrivateKey, hash(F-Emp1)).
```

Trust certificates define properties of a mobile device, e.g. its owner. These properties are evaluated against the constraints in the federation policy. Trust in a mobile

device is established because the device is owned by the same administration or is owned by another administrative organization with an established trust relationship, i.e. cross-certification.

**Table 2.** Token specification and example based on scenario

| | | |
|---|---|---|
| **TOKEN-TYPE** | "AUTH": Authorization | "TRUST": Device trust |
| **TOKEN-HOLDER** | Primary user identifier | Primary device identifier |
| **TOKEN-ISSUER** | Trusted corporation – typically employer | Trusted domain controller or device manufacturer |
| **TOKEN-VALIDITY** | Expiry date of token | |
| **TOKEN-ASSERTIONS** | User capability claim | Device capability claim |
| **SIGNED** | User assertion confirm | Device assertion confirm |
| `* Token-Type: "AUTH"; Token-Holder: Emp#2.PubKey;`<br>`Token-Issuer: Company.PubKey; Token-Validity: 31122004;`<br>`Token-Assertions: "General Management";`<br>`Signed: sign(Company.PrivKey, hash(Emp#2.PubKey)).`<br>`* Token-Type: "TRUST"; Token-Holder: Laptop002.AttestationKey`<br>`Token-Issuer:Manufacturer.PubKey; Token-Validity: 31122004`<br>`Token-Assertions: "Trusted";`<br>`Signed: sign(Manufacturer.PrivKey, hash(Laptop002.)).` | | |

### 4.2 Secure Session Transfer Protocol and Security Context

The session transfer protocol is message-based, allowing low coupling between the source and target device. We also introduce the additional assumption that the target and source have exchanged a lower level session key for full encapsulation of the packets including headers and payload. The objective of the protocol is to bring the source and target to agreement, such that the claims made by the target correspond with constraints specified within the source's federation and delegation policies. We refer to the process of coming to agreement as "security context negotiation." Security context negotiation consists of several steps (see Figure 4), given that the trust certificate of the target device has already been checked. If the target device is un-trusted (based on an unavailable token) no further negotiation occurs and the session transfer is cancelled. Table 3 below contains a description of payload (contents) of the message types exchanged between the source and target device during the negotiation of the security context.

In Case 1 of Figure 4, the target device could not make the appropriate claims to have the capability of supporting the security context, while in Case 2 the secure session transfer is disabled, as the source device doesn't agree with the possible capabilities sent by the target device.

**Table 3.** Message types for security context negotiation

| Message Type | Payload description |
|---|---|
| SESSION_OBLIGATION | Originates at the session source. Specifies the obligations that a target must do or attributes that must be possessed, in order to qualify as a valid host for the application session. The payload is the ASSERTIONS of the **federation** and **delegation** policies associated with the application running on the source. |
| SESSION_ SECURITY_ NEGOTIATION | Originates at the session target as a response to an obligation, which cannot be fulfilled by the target under the conditions proposed by the source. The payload states that a negotiation about the security constraints is necessary. |
| SESSION_ SECURITY_ POSSIBILITIES | Originates at the source as a response to a negotiation message from the target, based on its **configuration**. The payload contains a listing of TOKEN-ASSERTIONS that the source will accept for the transfer of the session. It is also a response from the target to the same message from the source where the payload contains a listing of constraint terms that the target can support for the transfer of the session. |
| SESSION_CONFIRM | Originates as a confirmation from the target that it can perform the obligations specified by the source. In this case the payload contains the target's proof that it can perform these obligations. The message type is also used by the source as an acknowledgement and acceptance of a target as a session host. In this case the payload contains the serialized session binary and user data. |
| SESSION_DISABLE | This message is sent by either the source or target when it is noticed that a stage in the negotiation process has been reached and no further possibilities exist. The session transfer is then terminated |

## 5 Conclusions

We have described an approach for a secure session transfer. The approach comprises respective system architecture and framework as well as a protocol that defines the different steps of the session transfer. Session transfer is done in a secure manner, i.e. it validates the conditions whether a session transfer is allowed or not, it establishes protected communication links to target devices, it performs security context negotiation and, if all previous steps are successful, it transfers the session from source to target device.

The protocol is semantically supported by security policies and verifiable tokens. The former define the constraints to be met before (i.e. decision whether transfer is possible or not) and after session transfer (i.e. respective security context.). Tokens are utilized to claim attributes of suitable trustworthy mobile devices, which may be target of a session transfer.

Our session transfer protocol extends current session transfer protocols by explicitly dealing with security constraints. Not only to secure the session transfer itself (which is done in our framework as well) but additionally checking security constraints of the service or application. Our motivation to look into security requirements of services and applications is driven by the specific usage scenario we are focussing on: mobile business applications where security is a vital element.

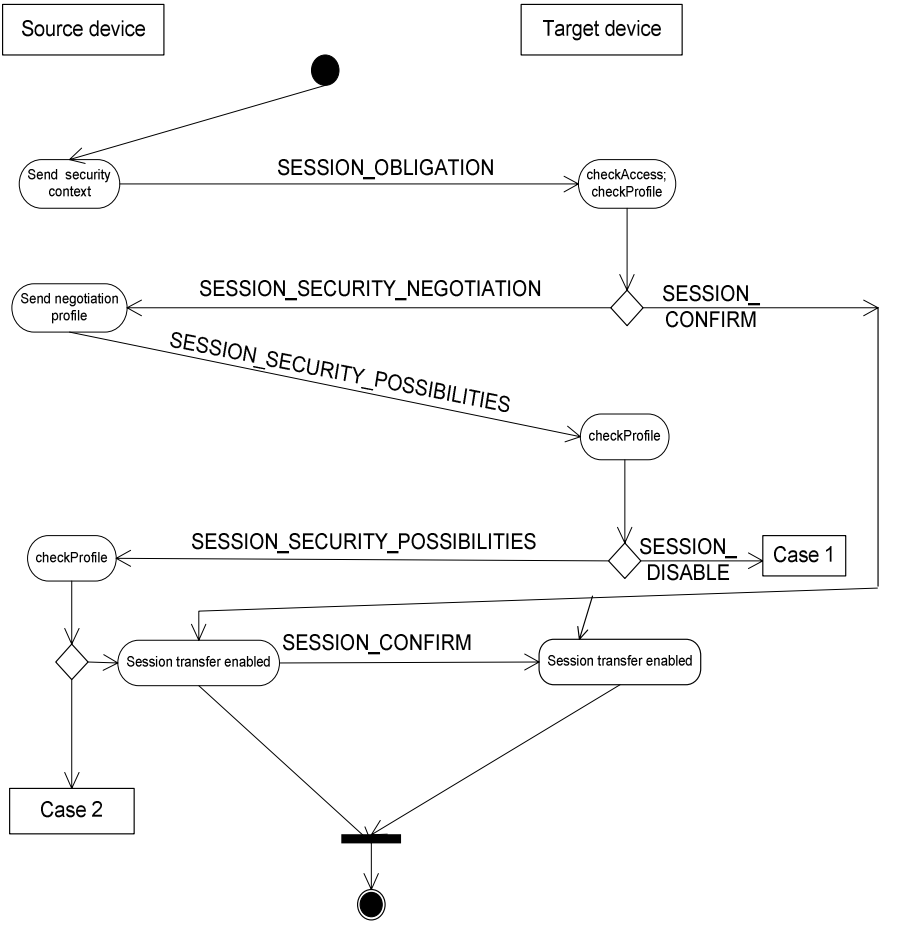



**Fig. 4.** Security Context Negotiation activity diagram

Although our proposed session transfer protocol provides for a transfer and re-installation of the session state as well, this is not a prerequisite. If a scenario does not require an explicit state transfer our protocol is equally applicable: it performs an evaluation of security policies, identifies feasible mobile devices, performs a security context negotiation and does the session transfer by terminating the session on source and re-starting the session on target mobile device. This could be likened to a secure *session-replication.*

In the paper it is assumed that source and target device support the framework as described before. Therefore a challenging task is to support devices not having this framework. For this support a translation mechanism between the different descriptions of security properties is necessary. To indicate which translation mechanism is needed the SESSION_SECURITY_NEGOTIATION message can be extended.

## References

1. G. Kouadri Mostéfaoui, P. Brézillon: A Generic Framework for Context-Based Distributed Authorizations, Proceedings of the Fourth International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'03), Lecture Notes in Artificial Intelligence 2680, Stanford, California (USA), June 23-25, 2003
2. R. Bagrodia, S. Bhattacharyya, F. Cheng et al.: iMASH: Interactive Mobile Application Session Handoff, ACM International Conference on Mobile Systems, Applications, and Services (MobiSys '03), May 2003.
3. E. Skow, J. Kong, T. Phan, F. Cheng et al.: A Security Architecture for Application Session Handoff, International Conference on Communications (ICC 2002), April 28 - May 2, 2002.
4. ITU-T Recommendation X.511: Abstract Service Definition, 1993.
5. H. Berndt, et al.: The TINA Book, Prentice Hall Europe, London, 1999.
6. NTT DoCoMo: All-IP Mobile Network Platform supporting a Ubiquitous Society – cover page, NTT DoCoMo Technical Journal, Volume 4, Number 4, March 2003.
7. R. Shirey: Internet Security Glossary, IETF International Request for Comments 2828, May 2000.
8. P. Robinson, M. Rits, R. Kilian-Kehr: An Aspect of Application Security Management, AOSD Workshop on Application-level security (AOSDSEC), Lancaster, UK, March 2004.
9. S. Thakolsri, W. Kellerer: Application-layer mobility, DoCoMo Euro-Labs Internal Technical Report, January 2004.
10. M. Handley, V. Jacobson: Session Definition Protocol, IETF RFC 2327, April 1998.
11. T. Walter, L. Bussard, Y. Roudier, J. Haller, R. Kilian-Kehr, J. Posegga, P. Robinson: Secure Mobile Business Applications – Framework, Architecture and Implementation, Information Security Technical Report, Vol. 9, No. 4, 2004.
12. H. Schulzrinne, E. Wedlund: Application-Layer Mobility Using SIP, ACM Mobile Computing and Communications Review, Vol. 4, No. 3, July 2000.
13. J. Rosenberg, et al.: SIP: Session Initiation Protocol, IETF RFC3261, June 2002.
14. N. Dulay, E. Lupu, M. Sloman and N. Damianou: A Policy Deployment Model for the Ponder Language, IFIP/IEEE Symposium on Integrated Network Management, Seattle, USA, 2001.
15. L. Bussard, Y. Roudier, R. Kilian-Kehr, S. Crosta: Trust and Authorization in Pervasive B2E Scenarios, 6th Information Security Conference, Bristol, UK, October 2003.

# Modeling and Evaluation of Security Architecture for Wireless Local Area Networks by Indexing Method: A Novel Approach

Debabrata Nayak[1,2], D.B. Phatak[1], and V.P. Gulati[2]

Student IEEE Member, Senior IEEE Member
[1] IIT Mumbai, Kanwal Rekhi School of Information Technology,
IIT Bombay, Powai 400076, India
+91-22-2576 7901/02. Fax: +91-22-2572 0022
{Debu, dbp}@it.iitb.ac.in
http://www.it.iitb.ac.in/~debu/index.html
[2] Institute for Development and Research in Banking Technology,
IDRBT, Reserve Bank of India, Hyderabad
{Vpgulati, dnayak}@idrbt.ac.in

**Abstract.** In this paper, we investigated existing and proposed WLAN security technologies designed to improve 802.11 standard. Security concerns over WLAN vulnerabilities are explored, and associated techniques are provided to mitigate these vulnerabilities. We also analyzed the existing architecture types of AAA integrated network security solutions, 802.1X and VPNs. We have extensively analyzed the effect of crypto parameters over WLAN based on packet level characteristics. We have also analyzed the effect of TCP and UDP traffic over our proposed WLAN testbed architecture. We found that TCP and UDP traffic behaves erratically, when security index changes causing drastically degradation of system performance. In this paper, we present a detail study of performance overhead caused by the most widely used security protocols such as WEP, IPSEC VPN and 801.1X. Furthermore, we analyze the effectiveness of such solution, based on measurement of security indexing model implementation. Performance measurement indicates that 802.1X and VPN method can be used based on the service time in future wireless systems, while it can simultaneously provide both the necessary flexibility to network operators and a high level of confidence to end users.

**General Terms:** Mobile security, Wireless privacy, And port based Access point.

**Keywords:** EAP, WPA, WVPN, WVLAN, 802.1X. Security index.

## 1 Introduction

The 802.1X standard has been introduced to provide a centralized authentication and dynamic key distribution for 802.11 architecture utilizing the 802.1X standard with RADIUS [2,5,8]. 802.1X is an authentication standard for 802-based LANs using port-based network access control. The 802.1X standard is used for communication

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 25 – 35, 2005.

between wireless clients and APs, while RADIUS operates between an AP and an authentication server WLAN risks can be mitigated by applying both basic and AAA infrastructure countermeasures to address specific attacks and threats [9,11,]. Basic countermeasures involve altering the existing security functions provided within wireless equipment. These countermeasures provide only limited defense against casual attacks; for determined adversaries, organizations should consider AAA integrated solutions [13,16]. The AAA infrastructure countermeasures provide integrated solutions using existing AAA infrastructure components such as the RADIUS protocol and public key infrastructure (PKI), with network solutions such as VPN and the 802.1X standards [2,15,18].

## 2 802.1X Model

- *Index 1 No security:* this is the default security setting provided by vendors. There is no security mechanism activated with default configuration.
- *Index 2 MAC address authentication:* this Index provides MAC address authentication carried out at the AP.
- *Index 3 WEP authentication:* the shared key authentication method specified in the 802.11 standard is used.
- *Index 4 WEP authentication with 40-bit WEP encryption:* this Index combines the encryption algorithm to provide data privacy.
- *Index 5 WEP authentication with 128-bit WEP encryption:* the 128-bit shared key used is proprietary-based.
- *Index 6 EAP-MD5 authentication:* this is one of the 802.1X standard's authentication methods, using password or username.
- *Index 7 EAP-TLS authentication:* this is the PKI-based authentication method supported by 802.1X.
- *Index 8 EAP-MD5 with 128-bit WEP encryption:* the combined effect of these tools provides strong data protection.
- *Index 9 EAP-TLS with 128-bit WEP encryption:* the combined effect of these tools provides the strongest Index of encryption and authentication using per-session keys.
- *Index 10 EAP-TTLS with 128-bit WEP encryption:* the combined effect of these tools provides the strongest Index of encryption and authentication using dynamic per-session keys.

## 3 VPN Model

- *Index 1 No security:* this is the default security setting. Both the 802.lX and VPN models have this in common.
- *Index 2 PPTP tunneling with CHAP:* authenticated tunnel provided using PPTP tunneling and CHAP authentication.
- *Index 3 IPSec tunneling with CHAP:* authenticated tunnel using IPSec tunnel and CHAP authentication.

- *Index 4 Firewall with PPTP and CHAP:* introducing a firewall into the architecture to filter the network traffic.
- *Index 5 Firewall with IPSec and CHAP:* a firewall is introduced into an IPSec based network. From this index onward, all the security Indexes will be based on IPSec design.
- *Index 6 Firewall with IPSec and EAP-TLS:* applying user-based PKI with device based certificate authentication.
- *Index 7 IPSec with CHAP and DES:* provides DES encryption to IPSec with CHAP user authentication.
- *Index 8 IPSec with EAP-TLS and DES:* applies DES encryption to EAP-TLS user authentication.
- *Index 9 IPSec with CHAP and 3DES:* provides strongest encryption (3DES) with CHAP.
- *Index 10 IPSec with EAP-TLS and 3DES:* encrypts data traffic with the strongest encryption and user authentication methods.

## 4 Experimental Testbed

All systems use RHL 9.0 kernel 2.4.20. Routers, Hosts are IBM systems (Pentium IV 2.8 GHZ). Moreover, Sharp Zaurus (Intel XScale 400 MHz with Linux Embedix), iPAQ (Intel StrongARM 206 MHZ with Familiar Linux) and IBM Laptop (Celeron Processor 2.4GHZ with RHL 9) are used as Roaming host. Open source softwares such as FreeSwan for IPSEC Xsupplicant for 802.1x supplicant, Free Radius for Radius server, OpenSSL for SSL, Mobile IP from Dynamic, Ethereal (packet analyzer), Netperf and ttcp (network monitoring utilities) are used for different functionalities in the testbed.

1. The experimental environment consists of windows XP OS. (As XP has built in implementation of 802.1X)
2. Server ML-530, XEON, 1 GHz, 1 GB RAM, 128 GB HD
3. Cisco Aironet 1100 - wireless access point RAM Installed (Max): 16MB, Flash Memory Installed (Max): 8 MB flash. Ethernet, Fast Ethernet, IEEE 802.11b. SNMP, Telnet, HTTP, DHCP support, BOOTP support, VLAN support, manageable.
4. Windows XP clients with 2.4 Ghz, 512 RAM, ORINICO USB client and OriNiCO gold card.
5. Mobile Pentium Centrino with 2.4 GHz, 512 MB RAM, 60 GB HDD, 8X CD/DVD writer, Wireless LAN with windows XP.
6. PIX 535 Processor: 1.0-GHz Intel Pentium III, Random Access Memory: 512 MB, or 1 GB of SDRAM, Flash Memory: 16 MB, Cache: 256 KB level 2 at 1 GHz, System BUS: Dual 64-bit, 66-MHz PCI; Single 32-bit, 33-MHz PCI.

## 5 Performance Analysis

The graphs shown in Figures below present's overviews of the security mechanisms and its impacts on performance. The 802.1X model provides better response times and throughputs .The IPSec-based VPN model provides end-to-end security that results in higher performance overheads.

### 5.1 Mean Response Time Variation

We found in fig-1 and fig-2 that FTP performed better than HTTP because the later requires more interaction between the server and the client. Providing same data file sizes for both traffic types would represent a better measurement for HTTP.

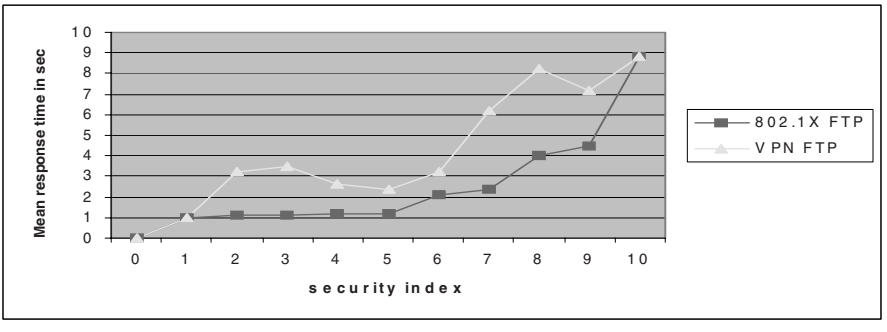



**Fig. 1.** FTP Mean response time for 802.1X and VPN

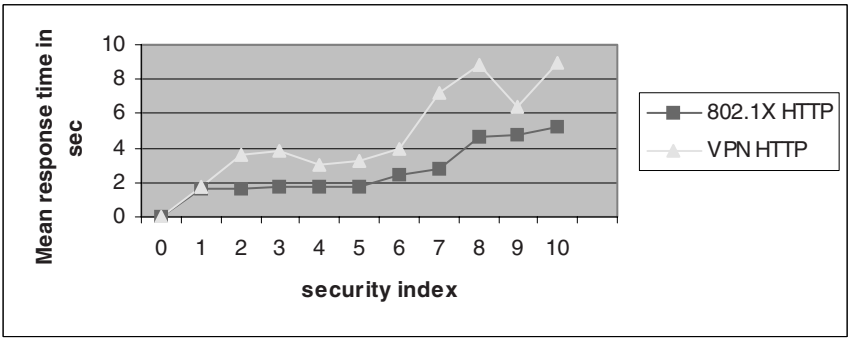



**Fig. 2.** HTTP Mean response time for 802.1X and VPN

### 5.2 Throughput Variation

An inverse relationship was found in both the 802.1X and VPN models between response time and throughput as response time increased throughput decreased as shown in fig-3 and fig-4.

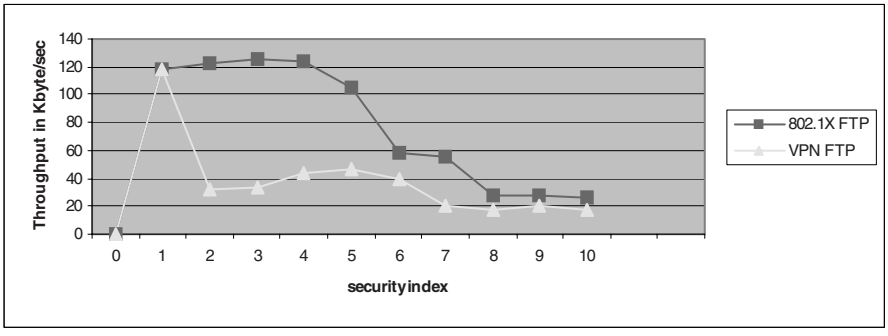



**Fig. 3.** FTP Throughput for 802.1X and VPN

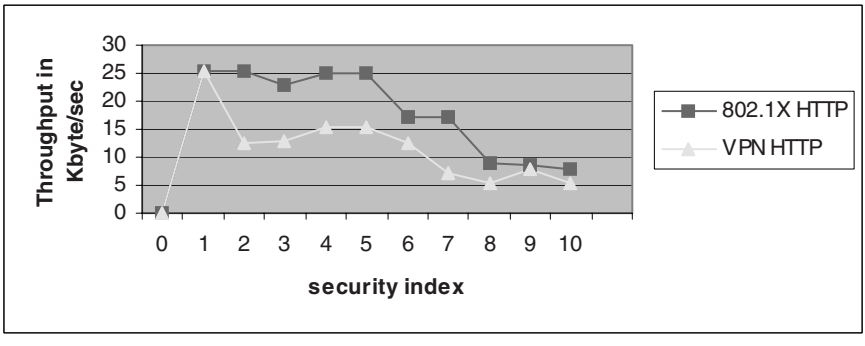



**Fig. 4.** HTTP Throughput for 802.1X and VPN

### 5.2.1 Impact of Packet Size on Mean Response Time

Varying the transmission packet size has a direct influence on mean response time in wireless network. This is because the larger the packet size, the longer is the packet transmission, propagation and processing times. The minimum and the maximum packet size consider in our experiment are from 100 byte to 2000 byte. The mean response time increases 4.25 ms per 100 bytes of packet length.

### 5.2.2 Impact of Packet Size on Throughput

Since wireless networking will be continuing to support HTTP and FTP based applications. We investigate the average throughput variation through the impact of packet size. Throughput increases from 27.5 to 155.4 kb/s when the packet size is increased from 100 to 1000 bytes. This corresponds to the increase of 392% in throughput. When the packet size increases to 1500 bytes the communication throughput reaches to 160 Kb/s as shown in fig-5.

We have increased the packet size to 2000 bytes and we found that the throughput gradually decreases. This is mainly happens due to the fragmentation of packets after 1500bytes.We observed that the throughput becomes 139.5 Kbytes/sec, which is 15.8% less in case of FTP traffic. This is happening due to there is the probability that the packet is get corrupted as shown in fig-6.

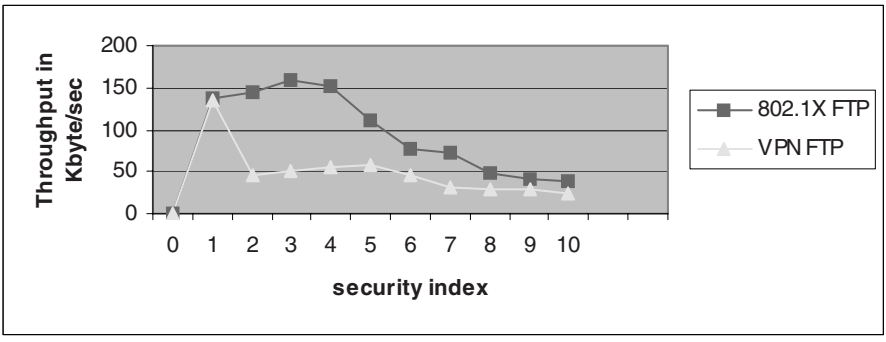



**Fig. 5.** FTP Throughput for 802.1X and VPN With packet size of 1000 byte

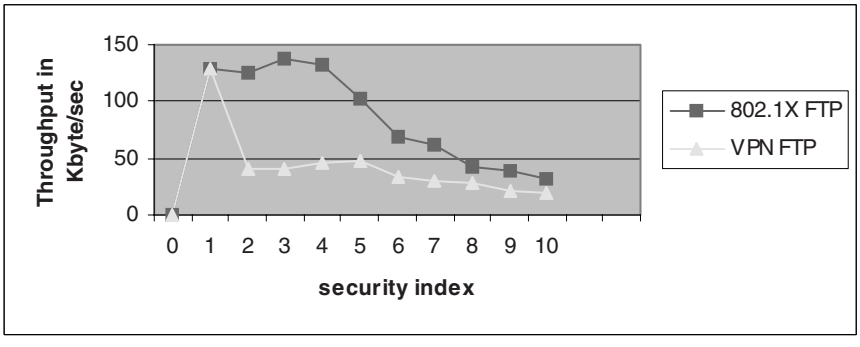



**Fig. 6.** FTP Throughput for 802.1X and VPN With packet size of 2000 byte

Throughput increases 5.14 Kbytes to 30 Kbytes for increase of packet size form 40 to 1000 bytes. When packet size increases to 2000 bytes then we observe that throughput decreases to 24.5 Kbytes for HTTP traffic. This is due to the defragmentation of packet and it reduces the throughput to 18.2% as shown in fig.7 and fig 8 below.

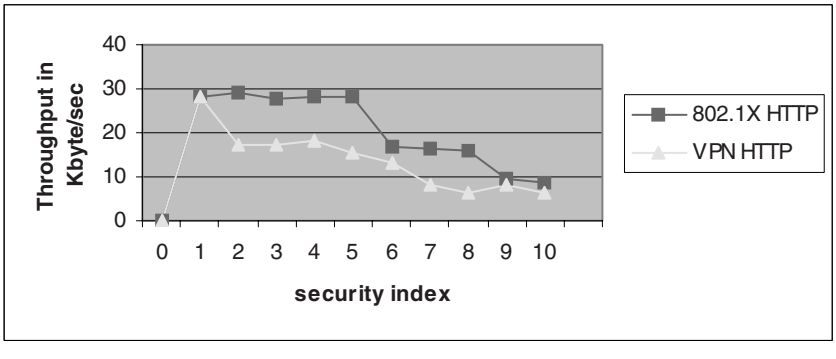



**Fig. 7.** HTTP Throughput for 802.1X and VPN With packet size of 1000 byte

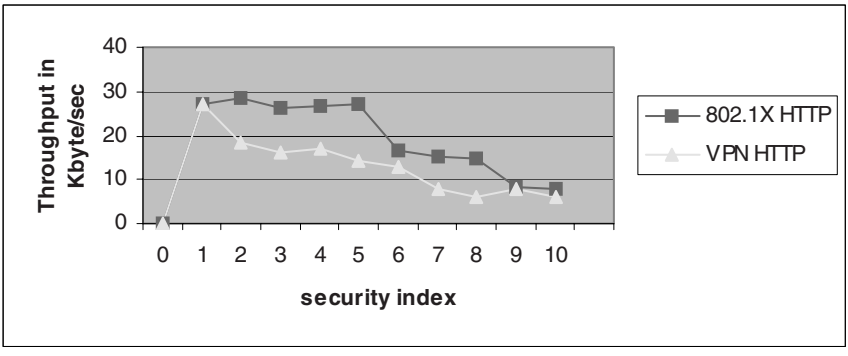



**Fig. 8.** HTTP Throughput for 802.1X and VPN With packet size of 2000 byte

### 5.3 802.1X Model Experimental Result

From result we found that MAC address authentication produces no performance overheads compared to default settings. WEP has minor impact on FTP throughput but decreases HTTP by 7.5%[8,11]. Since the effect is small, WEP authentication should be deployed. Using 802.1X authentication methods degraded network performance significantly compared to WEP authentication. Further EAP-TLS produced greater performance impact than EAP-MD5, as it provides mutual authentication and key management. By using encryption of WEP with 128 bits will reduce FTP performance by less than 20%[2,9,18]. But the use of EAP-MD5 and EAP-TLS increases the response time by 100%; there will be performance degradation of 47.8%. When we analyses the combined effect of the encryption and authentication FTP response time increases by 268% and throughput decreases by 73%[3,6].

### 5.4 VPN Model Experimental Result

In VPN model experimentation based authentication tunnel creates delay of response time of FTP by 245% and 113% for HTTP. So the throughput is reduced by 50%. But when employ EAP-TLS it takes delay time of 20%, which in turn decreases the throughput by 17%. By enabling DES and 3DES encryption for VPN model we found that there will be increase of response time by 130%, so the throughput decreases by 50%[4,8,12].

## 6 Simulation of WLAN with TCP and UDP Traffic

### 6.1 Mean Response Time Variation

Mean response time of UDP rises to 201%, where as in case of TCP it increases to 512% as shown in fig 9.

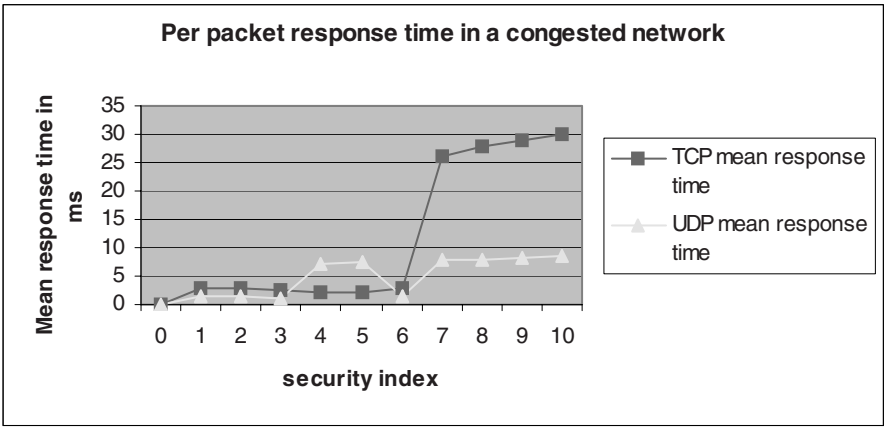



**Fig. 9.** Variation of Mean response time in highly loaded condition

### 6.2 Mean Throughput Variation

Under lightly loaded condition i.e. the bandwidth of 1Mbps,TCP throughput is 13.8% more than UDP throughput as shown in fig 10. But in case of heavily loaded condition i.e. in fig 11,UDP throughput is 22.3% more at bandwidth of 12 Mbps.

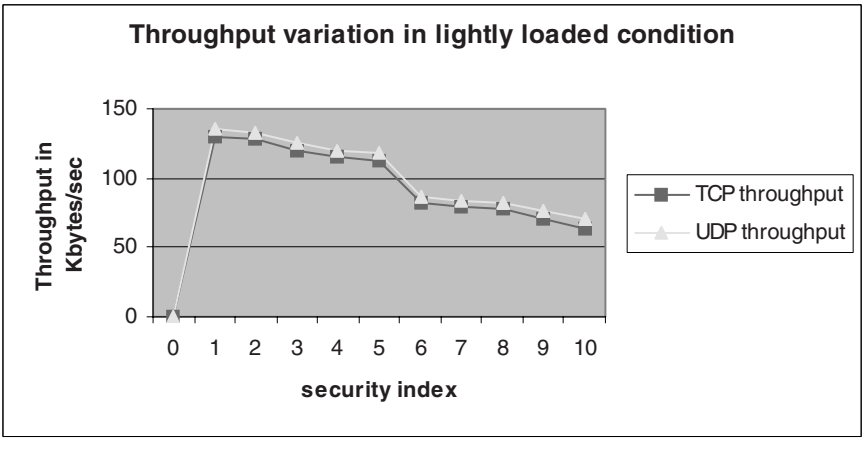



**Fig. 10.** Throughput variation in lightly loaded condition

In congested network, the overhead produced encrypting in each individual packets are significantly higher than that of EAP-TLS.Throughput drastically decreases when number of station are increasing as shown in fig 12.

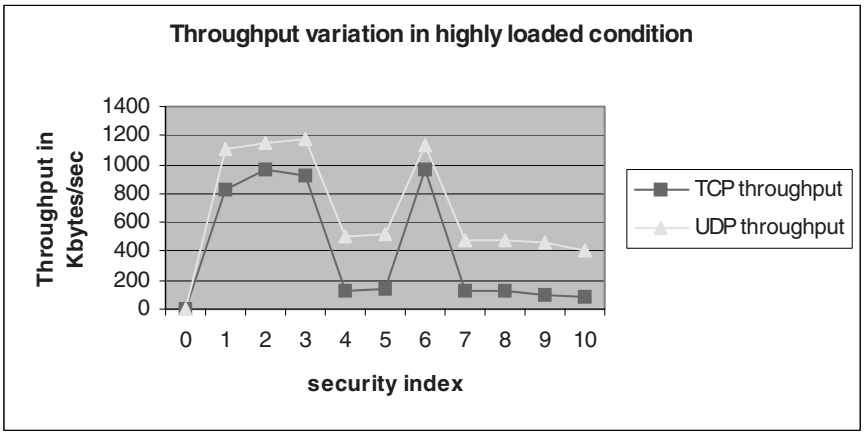



**Fig. 11.** Throughput variation in heavily loaded condition

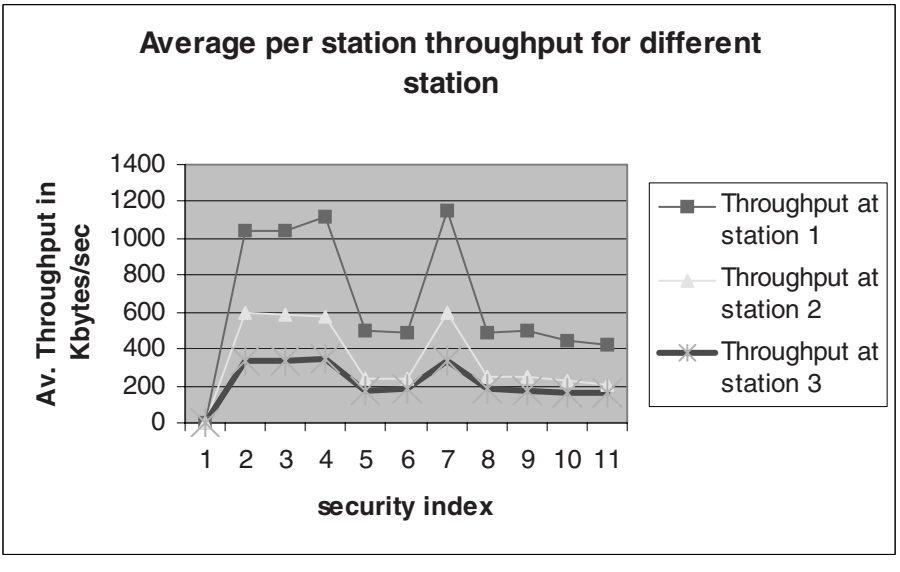



**Fig. 12.** Throughput variation for multi client scenario Throughput Variation

## 7 Conclusion

In this paper, we implemented the performance impact incurred with various security mechanisms considering different security indexes, and found that the more secured a network became, the higher the performance impact. The VPN model incurred greater performance degradation than the 802.1X model as we expected; the VPN model provided end-to-end security with double authentication (device and user), a stronger

encryption method as well as better key management and tunneling technology which in turn provides greater security. Combing different authentication and encryption methods produced significant impacts on performance. Keeping the encryption algorithm constant, then the choice of user authentication methods created substantial performance overheads and longer response times regardless of the encryption algorithm used. For DES, FTP is affected more than double the HTTP effect resulting in a 36.8 % delay and 32.3 % drop in throughput. Since we have implemented all this in our testbed, therefore our measurement provides first-hand valuable results, which will be very useful to the design and use of wireless security protocol for secure and flexible quality of service in future wireless networks.

## 8 Future Directions

A variety of techniques and some of the new Developments in Wireless Security environment are generally coming in near future like, IEEE802.11 (802.11e) working on extensions: WEP2/TKIP (Temporal Key Integrity Protocol), RrK (Rapid reKeying), ESN (Enhanced Security Network), AES (Advanced Encryption Standard), combination of these. Integration of WLANs (802.11) and WWANs (3G). e.g. handoff between 802.11b and CDMA networks secure handoff with existing security architectures.

## References

[1] Randall K. Nichols “Wireless security”, McGraw-Hill Telecom International Edition, 02.
[2] Bob Askwith, Madjid Merabti, Qi shi, Keith whiteley, “Achieving User privacy in Mobile Networks” proceedings of the 13th Annual computer security Applications Conference, IEEE 1997,pp.108-116.
[3] Mathew j.mayer, “A survey of security Issues in Multicast Communication”. IEEE transaction on computer networking, November/ December 1999, pp 12-23.
[4] Richard E.Smith, “Internet Cryptography”, Addison-Wesley Publishing Company, 1999.
[5] Sandry kay Miller “Facing challenge of the wireless security ”. IEEE Transaction on Computer, July 2001,pp 16-18.
[6] Erika Sanchez, Robert Edwards “Optimization of the Establishment of secure communication channel in wireless Mobile Networks”, proceedings of the international parallel and distributed processing symposium, IEEE 2002.
[7] Do van Thanh “Security issues in Mobile ecommerce ”, Proceedings of the 11th International workshop on database and Expert system Applications, IEEE 2000,pp.1-14.
[8] V.bharghavan, “Security issues in mobile communication”, Proceedings of the second international symposium on Autonomous Decentralized systems, IEEE 1995,pp.19-24.
[9] David A.cooper, Kenneth p. Birman, “Preserving privacy in A Network of Mobile computers”, proceedings of the IEEE Symposium on Security and Privacy, IEEE 1995,pp.26-38.
[10] Daniel patiyoot, S.j. shepherd, “Cryptographic security Techniques for wireless Networks”, IEEE pp.36-50.
[11] William stalling, “Cryptography and network Security”, Prentice Hall, Second Edition, 2000.

[12] Srinivas Ravi, Anand Raghunathan and Nachiketh Potlapally "Securing wireless data: system architecture Challenges ", acm journal, October 2002, pp.195-200.

[13] Paul Ashley, Heather Hilton, Mark Vandenwauver "Wired Versus Wireless Security", Internet: white paper /2002.

[14] Harris and Hunt, "TCP/IP security threats and attack Methods", Computer communications, Vol.22, 1999, pp.885.

[15] Ray Hunt, "Internet/Intranet firewall Security-policy, Architecture and transaction services", Computer Communications, Vol.21, August 1998, pp.1107-1123.

[16] JesiekB."InternetSecurity-Firewalls",andInternet http://www.ee.mtu.edu/course/ee465/groupb/fwll.html.

[17] Richard H. Baker, "Network Security", Tata Mc-Graw Hill, 2nd Edition, 1995.

[18] IEEE *Standard 802.11i / Draft 3.0. Draft Supplement to ISO/IEC 8802-11/1999(I) ANSI/IEEE Std802.11, 1999 edition: Specification for Enhanced Security*. November 2002. Pp.5-6.

[19] Rager, A.T. *WEPCrack Project Webpage*. Retrieved 9 May 2003 from http://sourceforge.net/projects/wepcrack/

[20] Mishra, A., N. L. Petroni, & B. D. Payne. (2003). *Open1x -- Open Source Implementation of IEEE 802.1x.* http://www.open1x.org/. June.

[21] Convery, S., & D. Miller. (2003). *SAFE: Wireless LAN Security in Depth, version 2*. White paper.Cisco Systems, Inc. http://www.cisco.com/warp/public/cc/so/cuso/epso/sqfr/safwl_wp.pdf

[22] Microsoft. (2002). *Wireless 802.11 Security with Windows XP*. Microsoft. http://www. microsoft.com//windowsxp/pro/techinfo/administration/wirelesssecurity/XP 80211 Security.doc

# Robust Routing in Malicious Environment for Ad Hoc Networks*

Zhongchao Yu[1], Chuk-Yang Seng[1], Tao Jiang[2], Xue Wu[1], and William A. Arbaugh[1,3]

[1] Dept. of Computer Science, University of Maryland, College Park, MD 20742, USA
{yuzc, sengcy, wu, waa}@cs.umd.edu
[2] Dept. of Electronic and Engineering, University of Maryland, College Park, MD 20742, USA
{tjiang}@glue.umd.edu
[3] University of Maryland Institute of Advanced Computer Studies, College Park, MD 20742, USA

**Abstract.** Secure routing in ad hoc networks has been extensively studied in recent years. The vast majority of this work, however, has only focused on providing authenticity of the route. Availability of the network in a malicious environment has largely been ignored.

In this paper, we divide the secure routing problem into two layers. The first layer provides authenticated routing and the second layer provides a route selection algorithm that selects a route with the highest probability of successful delivery rather than the shortest route. We provide a metric for evaluating this probability. We provide simulation results that demonstrate that our approach increases the throughput by at least ten percent in a network where fifty percent of the nodes are malicious when compared to an approach that selects the shortest route. Furthermore, our approach incurs only a small delay when compared to the delay along the shortest route.

## 1 Introduction

A mobile ad hoc network (MANET) is a dynamic collection of wireless nodes, communicating with each other over possible multi-hop paths. There is no predefined infrastructure to provide services, and each node within the network acts as a router, forwarding data packets for other nodes. Because each node acts as a router, the correct behavior of each node is vital to the efficacy of the network. In an adversarial environment, we always wish to make sure that our data packet falls into trusted hands.

Previous works on secure routing protocol focused on providing authenticated routes, by which we mean that each node on the route is authenticated.

* Portions of this work were funded by a Critical Infrastructure Grant from the U.S. National Institute of Standards and Technology.

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 36–47, 2005.

We call these protocols *authenticated routing protocols* in this paper. Some examples include Ariadne [1], SAODV [4], ARUN [5] and etc. Authenticated routing protocols prevent many of the attacks such as message fabrication, packet modification, impersonation and so on. However, as detailed in Section 2, authenticated routing protocols alone do not guarantee the correct behavior of the nodes. This is because authentication only verifies identity. It does not guarantee that the node will behave correctly. Therefore, these protocols must be augmented by other approaches which select routes on which nodes are not only authenticated, but also exhibit correct behavior in forwarding data.

We introduce a technique based on risk analysis to improve the robustness of routing. By robustness of routing, we are referring to the likelihood of successful delivery of data packets. To improve the robustness of routing in a malicious environment, we seek to choose the route that provides the minimum risk. We evaluate risk based on (partial) trust relationships between nodes.

We build a protocol based on risk – Risk-Based Protocol (RBP) to complement authenticated routing protocols. A protocol intended to secure routing in ad hoc networks without authenticating the nodes and routes is bound to be vulnerable to Sybil attacks [8]. While a protocol that ignores the behavior of the nodes within the network will suffer greater packet loss as the number of malicious nodes increases. An important insight is recognizing that *both* approaches must be used to provide secure routing.

Consequently, we present a novel routing paradigm for securing ad hoc networks. Our routing paradigm consists of a two layer architecture. The bottom layer provides authenticated routes. While the top layer evaluates risk by estimating the behavior of nodes on the route. This is the first paper which explicitly separates behavior from authentication in securing ad hoc routing protocols as well as recognizing the importance of using both together. The work in this paper, RBP, focuses on the top layer of the architecture– evaluation and selection of routes.

The rest of the paper is organized as follows. We will detail the rationale of this work in Section 2, followed by related work in Section 3. Then we introduce the trust model and assumptions we used in our work in Section 4, followed by the actual protocol in Section 5 and the evaluation of the protocol in Section 6. We conclude in Section 7.

We will use the following notation in the remaining parts of the paper.

1. We always use $S$ to represent the source node and $D$ to represent the destination node.
2. We use $R$ to refer to a specific route.

## 2 Motivation

In networking, we need to make sure that any ID claimed by a node really exists. Otherwise a node can claim to be any other node. However, even if an ID is authentic, it does not mean that it represents a well-behaved node. For

example, in a military scenario, nodes can be captured by the enemy and their key material for authentication may be disclosed. Moreover, it may be possible for intruders to exploit any software vulnerabilities to take control of a node. In such situation, the compromised node may carry out malicious acts without the knowledge of the node owner.

In the context of routing, an authenticated routing protocol thwarts attacks such as routing message modification, impersonation and packet fabrication. However, it is still possible for authenticated nodes to behave maliciously. It is quite possible for intermediate nodes to follow the protocol faithfully in the route discovery phase and then drop data packets later.

To handle the deficiencies of authenticated routing protocols, we must consider the behavioral aspect of the nodes along a route. In this paper, we use partial trust information to predict the behavior of nodes. We will define our notion of trust later.

Ideally, whenever a node $S$ sends a data packet, if it can always find a route on which every node is trusted by $S$ to behave appropriately, then we are done. If no such route can be found, the only thing we can do is to find a route with a minimum probability of being compromised. This process is all about risk inherently.

## 3 Related Work

Much of the previous works on securing routing protocols for ad hoc networks have focused on authenticating mobile nodes and routing messages [1, 2, 4, 5], while a smaller amount researches have focused on the behavior aspects of the nodes within the network [6, 7, 13].

For authenticated routing protocols, either the authenticity of nodes on the source route or the authenticity of metrics critical to the selection of routes, such as number of hops, is guaranteed. Ariadne [1], ARUN [5] and BISS [13] are examples of protocol that provides authenticity of nodes. As previously argued, authentication alone is insufficient.

SEAD [2] and SAODV [4] are two protocols which authenticate the number of hops or sequence numbers in the packets using hash chains. It does not assure integrity of the source or destination. Thus they have to rely on extra mechanisms to authenticate them. Like the above methods, they do not provide information about the behavior of authenticated nodes.

Some previous work try to solve the security problems based on estimating the behavioral aspects of nodes. For example, *reputation systems* [6, 7]. In general, nodes are rated according to their observed behaviors. The routing decision is then based on the *reputation* of the nodes along the route. However, these works do not address the problem of authentication. Therefore, if a node's reputation becomes bad, it can change its identity and restart its malicious behaviors. Moreover, they do not address the integrity of the reputation information.

Other approaches use a reward and charging scheme to enforce good behavior [9, 10, 11, 12]. In these schemes, nodes are assumed to be inherently selfish

but rational. This means they will not do anything that will cause significant harm to themselves. However, the assumption that nodes are selfish but rational is only reasonable when all the nodes in the network are resource constrained. Consider the possibility that some rogue nodes are resource abundant such that the rewards are not attractive to them.

## 4 Trust Model and Assumptions

In our research, we assume that authenticity is provided by an existing authenticated routing protocol, such as Ariadne. Hence the authenticity of identities, control messages and routes is always assured. We focus on the upper layer of our architecture, risk analysis.

We define the meaning of trust according to the context of routing protocols. We formally define "forwarding trust" in the following.

**Definition 1.** *Given a route $R$ from $S$ to $D$ and two nodes $X$ and $Y$ on the route, we call $X$ an upstream node of $Y$ (following $R$) if $X$ is closer to $S$ on the route. Correspondingly, we call $Y$ a downstream node of $X$ (following $R$).*

**Definition 2.** *$X$ has forwarding trust of $Y$ (we call $X$ trusts $Y$ in short for the rest of this paper), if and only if the following two conditions hold.*

1. *For all routes $R$, if $X$ sends or forwards any packets for $S$ following $R$ and $Y$ is a downstream node of $X$, then with some level of certainty, $Y$ forwards the packets along $R$ unless it is the final destination;*
2. *For all routes $R$, if $X$ sends or forwards any packets towards $D$ following $R$ and $Y$ is an upstream node of $X$, $Y$ forwards the packets following $R$ with some level of certainty.*

Informally, when $X$ trusts $Y$, $X$ believes that $Y$ will forward packets for $X$, with a certain level of confidence. Note that trust is subjective. $X$ may trust $Y$ although other nodes may distrust $Y$. Similarly, a malicious node may exhibit malice only to certain set of nodes. Therefore if $X$ trusts $Y$, the above definition implies that $X$ has a certain level of confidence that $Y$ will also forwards packets even when $X$ is not the source. Since $X$ is committed to forward packets for the source, if $Y$ drops packets, $Y$ is not only exhibiting malicious behavior towards the source, but also towards $X$. Therefore, by dropping packets originated from the source, $Y$ is also betraying $X$'s trust. In the event that $Y$ does not wish to forward packets for the source, $Y$ can either choose not to participate in the route discovery or $Y$ can drop the packets and therefore $Y$ betrays $X$'s trust. The term "some level of certainty" in Definition 2 takes into account situations where $Y$ betrays $X$. This also reflects that there is a risk involve in trusting a node. The same argument can also be applied to condition 2.

To address the level of certainty, we need to quantify trust. We map trust from $X$ to $Y$ to a value $tr_{XY}$ between the interval $[0, 1]$, representing the probability that $Y$ will behave well according to definition 2, from $X$'s point of view. When

$X$ trusts $Y$ completely, we have $tr_{XY} = 1.0$. When $X$ distrusts $Y$ definitely (i.e. $X$ thinks $Y$ is a bad node definitely) we have $tr_{XY} = 0$. Between these two extremes, $X$ can assign $Y$ a trust value $\beta$ in the interval of $(0, 1)$. $\beta$ depends on the information (evidence) available. For simplicity, we assume that all the nodes which are neither fully trusted nor fully distrusted have the same $\beta$ values. This assumption does not harm the generality of the method.

The assignment of values to $tr_{XY}$ is dependent on trust establishment method. Nodes may choose their own system of establishing trust. For example, while some nodes may choose the method in [7], a node who is paranoid may chose to give a neutral rating to nodes with high reputations. It is for this reason that we emphasized that trust is subjective. We do not address trust establishment in this paper. Trust is such a subjective issue that no single solution works for everyone, since everyone has their own notion of trust as illustrated above. Rather than developing our own method to establish trust and arguing its relative merits with existing works, we allow different approaches to be used.

Since not all nodes on a route are trusted, the source node needs to decide which nodes to trust using partial trust relationships. Consider a route $(S, A, B, D)$, in which $S$ only trusts $A$. With this partial trust relationship, $S$ needs to determine whether $B$ can be trusted. Suppose $A$ trusts $B$. Then with some level of certainty, $B$ will forward messages for $A$. If $B$ drops $S$'s data packets, $B$ betrays $A$'s trust. Hence it is implied that $S$ can trust $B$ too. This seems to suggest that forwarding trust is transitive. However, this is not the case. Suppose later on, if $S$ needs to send data to $D'$ using the route $(S, B, D')$, the relation $S$ trusts $B$ is no longer valid for this route. This is because in the absence of $A$, $B$ can drop $S$'s data packets without betraying $A$'s trust. We call this the property of conditional transitivity.

**Definition 3.** *The conditional transitivity of trust implies that for all distinct nodes $X$, $Y$ and $Z$, if $X$ trusts $Y$ and $Y$ trusts $Z$, then $X$ trusts $Z$ iff $X$, $Y$ and $Z$ belongs to the same route.*

Using the property of conditional transitivity, we define a trust chain, with respect to a route, as follows.

**Definition 4.** *A trust chain, with respect to a route, is a sequence of nodes $(X_1, X_2, \ldots, X_k)$ $(k \geq 2)$, such that (1)$X_i$ trusts $X_{i+1}$ $(1 \leq i < k)$, (2)the sequence preserves either the order or reverse order of the route, and (3)the sequence is not a subset of any other sequences obtained from the same route. As a special case, a sequence of a single node is also a trust chain.*

*A trust chain which either starts with $S$ and preserves the order of $R$ or starts with $D$ and preserves the reverse order of $R$ is called a routeable trust chain.*

As an example, consider Figure 1 where an arrow going from $X$ to $Y$ represents $X$ trusts $Y$. The trust chains are: $(S, B, C, D)$, $(D, B, S)$ and $(C, A, S)$. However, $(S, B)$ is not a trust chain since it is a subset of $(S, B, C, D)$. Although $(C, A, S)$ is a trust chain, it is not a routeable trust chain since it does not begin with $S$ or $D$. To simplify route evaluation, we only consider routeable trust chain in our protocol.

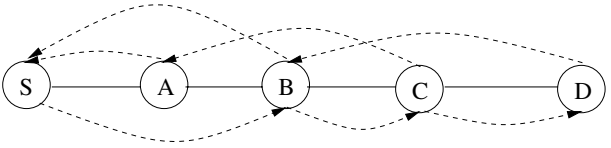



**Fig. 1.** An example of trust assertions

We assume that the source and the destination are always trusted with respect to routing. We do not consider the case in which a destination drops packets irrationally since by doing so, it defeats the purpose of establishing a route for the source and destination to communicate. Note that we are only referring to forwarding trust.

To bootstrap the route evaluation process, we assume the existence of established trusts. For example, trusts exist between nodes who are "friends" with each other, or reputation ratings from existing work can also be used as an initial measurement of trust.

## 5 Protocol

In this section, we design, as a proof of concept, a routing protocol RBP, using risk as the decision function for route evaluation.

### 5.1 Overview

RBP is a routing protocol based on DSR [14]. Like DSR, RBP consists of two basic operations: route discovery and route maintenance. Each node also maintains a route cache, which contains routes to various nodes. If there are no routes in the cache to a given destination, a route discovery process is initiated. Route maintenance mainly deals with route errors.

Our ultimate goal in RBP is to improve the throughput in a malicious environment with little additional overhead and delay. A natural way to achieve this is to improve the "quality" of the selected route.

**Definition 5.** *The quality of a route is the probability that by taking this route, the packet is successfully transmitted to the destination. The quality of a path $p$ is denoted by $q(p)$, and by definition, $q(p) \in [0, 1]$.*

**Definition 6.** *The candidate path set from the source $S$ to the destination $D$ is the set of all the paths to $D$ in $S$'s route cache at a given time.*

The quality metric is related to risk. The higher the quality, the lower the risk. The source node aims to reduce the risk of its packets being dropped by choosing routes with high quality. Hence, this is the risk analysis component of RBP. In order to improve throughput, RBP first improves the overall quality of

the candidate path set in the route discovery process. Specifically, RBP allows nodes to assert their trust relationships with other nodes on the same route during route discovery. The trust assertions are then used to evaluate the quality of the route. An intermediate node will only forward a route request if the partial route in the route request is of "good" quality. At the end of route discovery, the source node will the route in the candidate path set that has the highest quality.

### 5.2 Trust Assertion Propagation

In the route request and route reply phase, each intermediate node will check the current partial route to see if it has trust relationship with any of the nodes on the partial route. If it does, it will add a *trust assertion* for each of the trusted nodes to the route request packet.

**Definition 7.** *A trust assertion* $T(X, Y, tr_{XY})$ *is a data structure which asserts that* $X$ *trusts* $Y$ *with a value of* $tr_{XY}$*.*

In this paper, unless specified, $tr_{XY} = 1.0$ and $T(X, Y)$ represents $T(X, Y, 1.0)$. $T(X, Y)$ is also represented in diagrams as a dashed arrow going from $X$ towards $Y$.

In the route request (from $S$ to $D$), all the trust assertions are from downstream nodes to upstream nodes because a node does not know the downstream nodes at the time when it receives the route request. In the route reply packet, trust assertions added are from upstream nodes to the downstream nodes.

### 5.3 Route Evaluation

Central to our method is the evaluation of routes. Source node evaluates routes in its candidate path set to choose the route with the best quality. During route discovery, intermediate nodes evaluate partial routes, as described in details later. For this reason, we present route evaluation technique before route discovery. An example route from $S$ to $D$ is given in Figure 1.

We extract all the routeable trust chains. Then all the nodes on these routeable trust chains could be viewed as trusted nodes according to the conditional transitivity of trust property.

For each trusted node, $X_i$, in the routeable trust chain, we assign a trust value of $tr_{X_{i-1},X_i}$ or $tr_{X_{i+1},X_i}$, depending on whether the routeable trust chain starts from $S$ or $D$. For unknown nodes, we assign a trust value of $\beta$. As an example, in Figure 1, because of valid trust chains $(S, B, C, D)$ and $(D, B, S)$, $B$ and $C$ can be viewed as trusted while $A$ is unknown. Therefore, $S$ assigns trust values to the nodes as follows: $tr(A) = \beta$, $tr(B) = tr_{SB} = tr_{DB}$, and $tr(C) = tr_{BC}$.

The quality of route $R = (S, X_1, \ldots, X_k, D)$ is given by:

$$q(R) = \Pi_{i=1}^{k} tr(X_i) \tag{1}$$

Recall that in our work, we let $tr_{XY} = 1.0$ for any nodes $X$, $Y$. Therefore in Figure 1, the quality of the route is $\beta$.

Although we assign a trust value of 1.0 to each trusted node in this work, it does not have to be the case. We choose to assign 1.0 to simplify things. In actual fact, the source node is free to assign any values according to its own policies. Similarly, if $S$ has multiple values to choose from, such as deciding whether $tr(B) = tr_{SB}$ or $tr(B) = tr_{DB}$, the outcome of the decision depends on $S$'s policies.

### 5.4 Route Discovery

RBP's route discovery process is similar to that of DSR. However, we perform selective request broadcasting in this process to single out good partial paths.

When $S$ has data to send to the destination $D$, it first looks up its route cache. If there is no path to $D$ in its route cache, $S$ broadcasts a route request packet:

$$S \rightarrow *: \texttt{Route\_Request}, S, D, seqNo, R' = (S), \\ T\_Set = \{\}$$

where $seqNo$ is the sequence number and the current partial route $R'$, which only contains $S$ at the moment. $T\_Set$ represents the set of trust assertions. At this moment $T\_Set$ is empty.

When an intermediate node $I$ receives a route request, it first checks whether it has sent out a route request for $S$ with a sequence number greater than $seqNo$. If it has, it discards the route request packet.

If it is the first route request from $S$ with $seqNo$, unlike in DSR, $I$ does not immediately broadcast it because the partial route contained may not have a high enough quality. Also it might be a packet for rushing attacks [3].

Our solution is to buffer the route request for a short time, depending on the quality of the partial route. To compute the quality of the partial route, $I$ first adds trust assertions, if any, to the upstream nodes of $R'$. It then computes the quality of $R'$, following Equation (1) as if it were the final destination $D$.

Let the quality of the partial path be $q(R')$. $I$ buffers the route request for $\alpha(1-q(R'))$ seconds (called the *request buffer time window*), where $\alpha$ is a system constant[1].

If during the interval of the request buffer time window, $I$ receives another route request from $S$ with the same sequence number, it checks whether the partial route $R''$ has higher quality. If it does, the original route request is discarded and the new route request is buffered for $\alpha(1 - q(R''))$ seconds. This process continues until the timer expires. This buffered route request must contain a partial route with the highest quality among all the partial routes from $S$ seen so far. $I$ then broadcasts the expired route request after adding itself to $R'$.

When a route request arrives at the destination $D$, it appends itself to the partial route and also adds trust assertions for those nodes on the partial route. Then it reverses the route contained in the route request packet and unicasts a

[1] It is not clear whether a linear formula is the best strategy here. But, our simulations showed a linear request buffer time window works just fine.

route reply to the source as DSR does. $D$ replies to every route request it receives from $S$ so that $S$ have multiple paths to $D$. In the process of route reply, nodes will also add trust assertions into $T\_Set$, the set of trust assertions, if they trust some of their downstream nodes.

One attack might be that malicious nodes do not buffer the route request packet it receives and tries to send it out immediately. The consequence is that routes consisting of malicious nodes tends to broadcast faster and $S$ might lose some data packets by following these bad routes when good routes are not available at the moment. These attacks can only succeed in the initial short time window when the bad route arrives at the source while other good routes have not. But after the short time window, since bad routes have lower qualities, they will be ignored. So this attack is greatly limited.

Another potential attack from malicious nodes might be for them to remove trust assertions inserted by other nodes. However, this actually makes the route worse because this potentially reduces the number of trust chains and reduces the number of trusted nodes to the source.

Since we assume the assurance of route authenticity, it is not possible for malicious nodes to fake arbitrary trust assertions for other nodes to increase the quality of the routes. However it can freely assert trust on other nodes. The success likelihood of this attack depends on the trust establishment method, which is out of scope of this paper. A good trust establishment may be such that the probability that a good node trusts a malicious node is very low. Hence the probability that the trust assertions made by the malicious nodes to appear on any valid trust chains is also very low.

### 5.5 Route Selection

Section 5.4 allows the source $S$ to have multiple paths to $D$. Rather than selecting the route with the smallest number of hops, we select the route with the highest quality. Only when two or more routes have the same highest quality do we select the route with the smallest number of hops.

### 5.6 Route Maintenance

All the intermediates nodes as well as the source will have a chance to purge relevant route entries in their route cache according to the route error packet. Note that route errors should also be authenticated so that malicious nodes cannot forge fake route errors. That is, other nodes should have a way to verify that the route error does originate from the node originally finding the link failure.

However, malicious nodes can hide route error packets generated from other good nodes from its downstream so that its upstream nodes do not know the fact. There are three alternatives to address this problem.

First, if the reporting node has multiple routes to the source, it can send the route error packets along multiple paths to the source. A second method is to periodically send a packet to the destination and requires acknowledgment

from the destination. However, this method requires a little bit of overhead. The third method is to attach an age to each route to the destination. We attach a timer with each route. The timeout is inversely proportional to the quality of the route. When all the routes time out, we initiate a new route request process.

## 6 Evaluation

### 6.1 Performance

To evaluate the performance of RBP, we implement it in *ns-2* [16]. DSR is not a route authentication protocol but we assume that it is authenticated. We call it DSR_auth in the following. We do not simulate attacking behaviors which exploit authentication related security holes. We present a very simple attacking model here. Malicious nodes will drop data packets and route error messages. They will allow other control messages to pass through.

**System Parameters.** As described previously, in RBP, we need to decide on the following parameters: $\alpha$ and $\beta$.

Literally, $\alpha$ represents the maximum buffering time for route request messages. Let the delay between two neighboring nodes be $T_D$.

In the simulation code, the timeout for ARP is about 0.03s. Thus we can estimate $T_D = 0.015$ (actually this is an upper bound value). As a guideline, $\alpha$ should neither be too much greater than $T_D$ nor too much smaller than $T_D$. In our simulation, $\alpha$ is 0.04.

The accuracy of $\beta$ depends on how much extra information (evidence) is available. If we could estimate the fraction of malicious nodes, $b$, in the system, we could set $\beta = 1 - b$. In general cases, we often assume that the number of malicious nodes are less than good nodes, thus we can use $\beta = 0.5$ in most cases.

**Simulation Setting.** We set 50 nodes in a 670 m by 670m area. Each node in our simulation moves according to the *random waypoint* model [15], at a velocity of 20 m/s. Nodes transmit at a constant bit rate (CBR) of 4 packets per second, each packet is of 64 bytes.

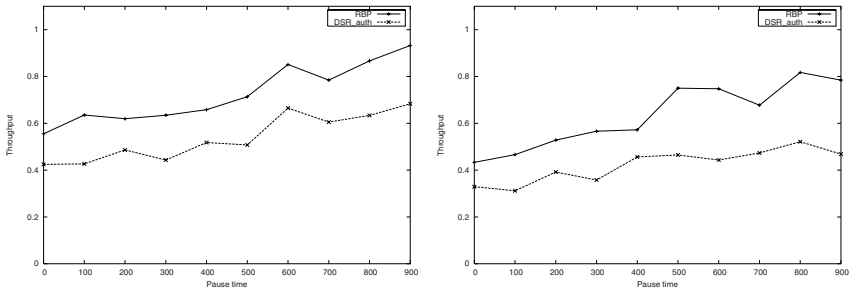
We are going to compute the following metrics for the performance:

- *Throughput*: The fraction of CBR data packets sent that are received.
- *Delay*: The average time between a CBR data packet is sent and received.

All the above metrics are averaged over 10 runs with different CBR and mobility scenarios. RBP and DSR_auth run on the same scenarios each time.

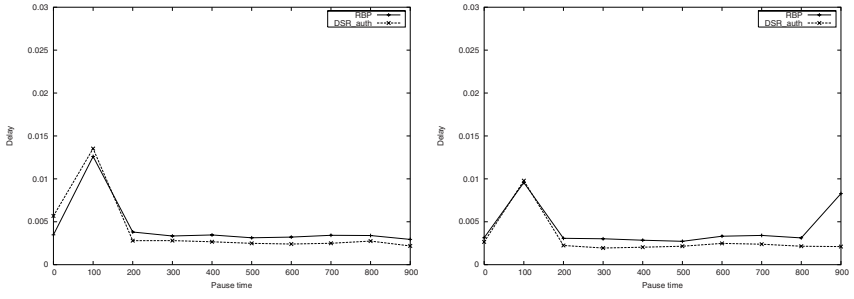
**Simulation Result.** According to $b$, the fraction of malicious nodes in the network, we categorize the attacks into three degrees: light attack ($b = 0.1$), medium attack ($b = 0.3$) and severe attack ($b = 0.5$).

In most of the simulations, we will use the following typical parameters to show how RBP performs under attacks compared to DSR_auth: $\alpha = 0.04$ and $\beta = 1 - b$.




(*a*) $b = 0.3$ (*b*) $b = 0.5$

**Fig. 2.** Throughput comparison in a typical setting for RBP, where $\alpha = 0.04$ and $\beta = 1 - b$




(*a*) $b = 0.3$ (*b*) $b = 0.5$

**Fig. 3.** Delay comparison in a typical setting for RBP, where $\alpha = 0.04$ and $\beta = 1 - b$

Unfortunately, due to space constraints, we are unable to reproduce every diagrams. In such situations, we will only quote the results we obtained.

Figure 2 and Figure 3 plot the throughput and delay respectively, of the RBP with typical settings, compared to DSR_auth.

We observe that under all three levels of attacks, RBP greatly improves the throughput while involving only trivial delays.

## 7 Conclusions

We have proposed a secure architecture for routing in ad hoc networks. Our architecture emphasizes the importance of authenticity and behavior of nodes. It is our hope that this architecture will promote the awareness of the coupling relationship between these two issues, since previous work only addresses these issues in isolation.

We have also proposed RBP. One of the highlights of RBP is through the use of partial information in the form of trust chains, RBP computes the probability of successful delivery of data along a given route. In the absence of complete and

reliable information, the best that a node can do is to make use of the partial information it has to improve the delivery rate.

Our simulation shows that RBP increases throughput in a malicious environment. Results show that throughput is increased by at least ten percent over non-optimized DSR even when fifty percent of the nodes in the network are malicious. The increase in throughput is achieved with only a small delay.

## References


1. Y.-C. Hu, A. Perrig and D. B. Johnson. Ariadne: a secure on-demand routing protocol for ad hoc networks. *MobiCom 2002*.
2. Y.-C. Hu, D. B. Johnson and A. Perrig. Secure efficient distance vector routing in mobile wireless ad hoc networks. In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications* (WMCSA). June 2002.
3. Y.-C. Hu, A. Perrig and D. B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. *WiSe2003*.
4. M. G. Zapata and N. Asokan. Securing ad hoc routing protocols. *WiSe 2002*.
5. K. Sanzgiri and B. Dahill. A secure routing protocol for ad hoc networks. *ICNP 2002*.
6. S. Buchegger and Jean-Yves Le Boudec. Performance Analysis of the CONFIDANT Protocol (Cooperation of Nodes: Fairness In Dynamic Ad-hoc Networks). *MobiHoc2002*.
7. Sergio Marti and T. J. Giuli and Kevin Lai and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. *MobiCom 2000*.
8. John R. Douceur. The sybil attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, 2002.
9. L. Buttyan and J.-P. Hubaux. Enforcing service availability in mobile ad hoc WANs. In *Proceedings of the IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Boston, August 2000
10. L. Buttyan and J.-P. Hubaux. Stimulating cooperation in self-organization mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications*, 8(5), Oct 2003.
11. M. Jackobsson, J.-P. Hubaux and L. Buttyan. A micro-payment scheme encouraging collaboration in multi-hop cellular networks. In *Proceedings of Financial Cryptography*, 2003
12. S. Zhong, Y. R. Yang and J. Chen. Sprite: a simple, cheat-proof, credit-based system for mobile ad hoc networks. In *Proceedings of INFOOM*. IEEE, 2003
13. Srjian Capkun and J.-P. Hubaux. BISS: building secure routing out of an incomplete set of security associations. *WiSe 2003*.
14. D. Johnson, D. Maltz and J. Broch. DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks. In *Ad Hoc Networking, edited by Charles E. Perkins*, Chapter 5, pp. 139–172. Addison-Wesley, 2001.
15. J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'98)*, Dallas, TX, USA, Oct. 1998, pages 85–97.
16. The Network Simulator ns-2. *http://www.isi.edu/nsnam/ns/*

# Short Linkable Ring Signatures for E-Voting, E-Cash and Attestation

Patrick P. Tsang and Victor K. Wei

Department of Information Engineering,
The Chinese University of Hong Kong,
Shatin, Hong Kong
{pktsang3, kwwei}@ie.cuhk.edu.hk

**Abstract.** A ring signature scheme can be viewed as a group signature scheme with no anonymity revocation and with simple group setup. A *linkable* ring signature (LRS) scheme additionally allows anyone to determine if two ring signatures have been signed by the same group member. Recently, Dodis et al. [18] gave a short (constant-sized) ring signature scheme. We extend it to the first short LRS scheme, and reduce its security to a new hardness assumption, the Link Decisional RSA (LD-RSA) Assumption. We also extend [18]'s other schemes to a generic LRS scheme and a generic linkable group signature scheme. We discuss three applications of our schemes. Kiayias and Yung [22] constructed the first e-voting scheme which simultaneously achieves efficient tallying, public verifiability, and write-in capability for a typical voter distribution under which only a small portion writes in. We construct an e-voting scheme based on our short LRS scheme which achieves the same even for all worst-case voter distribution. Direct Anonymous Attestation (DAA) [6] is essentially a ring signature scheme with certain linking properties that can be naturally implemented using LRS schemes. The construction of an offline anonymous e-cash scheme using LRS schemes is also discussed.

## 1 Introduction

A *group signature* scheme [15] allows a member to sign messages anonymously on behalf of his group. The group manager is responsible to form the group and assign to the members the ability to sign. However, in the case of a dispute, the identity of a signature's originator can be revealed (only) by a designated entity.

A *ring signature* scheme [29] can be viewed as a group signature scheme with no anonymity revocation and with simple group setup. Formation of a group is *spontaneous*: diversion group members can be totally unaware of being conscripted to the group. Applications include leaking secrets [29] and anonymous identification/authentication for ad hoc groups [5, 18].

*Linkable ring signatures* [23] are ring signatures, but with added linkability: such signatures allow anyone to determine if they are signed by the same group member (i.e. they are *linked*). If a user signs only once on behalf of a group, he still enjoys anonymity similar to that in conventional ring signature schemes. If

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 48–60, 2005.

the user signs multiple times, anyone can tell that these signatures have been generated by the same group member. Applications include leaking sequences of secrets and e-voting [23]. Concepts similar to linkability also appeared in one-show credentials [7], linkable group signatures [26, 27], and DAA [6].

Early constructions of (linkable) ring/group signature schemes have large signature sizes, which are usually $O(n)$ where $n$ is the group size. Subsequent results incorporating various techniques reduced the sizes of state-of-the-art group signatures to a constant independent of group size. Consult [11, 1, 8, 3, 9, 28] for details. Essentially all ring signatures have sizes $O(n)$. Recently, Dodis, et al. [18] gave a short ring signature scheme construction. In this paper, we extend their technique to construct a short LRS scheme. We also extend [18]'s generic ring (resp. group) signature scheme constructions to their linkable version.

*Tracing-by-linking versus tracing-by-escrowing in group signatures.* The following two papers came to our attention after the completion of this research: Teranishi, et al. [30] and Wei [34]. They achieve *tracing-by-linking*, i.e. tracing the double signer's public key without identity escrowing to an Open Authority (OA). In comparison, traditional group signatures use the *tracing-by-escrowing* technique, and give the Open Authority the unnecessary power to open an honest signer's identity even when there is no dispute to investigate. Comparing the two tracing-by-linking group signatures: [30]'s has smaller size. [34]'s has larger, but still $O(1)$, size, but it is more flexible and supports features such as tracing the double signer's secret key, tracing the double signer's identity without going through the public key, etc. Another paper containing tracing-by-linking ring signature is due to Tsang, et al. [33].

Constant-sized LRS schemes have many applications. We describe three of them briefly in the following.

*E-Voting.* There are three basic paradigms for cryptographically secure ballot elections. Under the *blind signature* [13] paradigm, the voters obtain ballots from the authorities, certified but privacy-preserved. This enables them to embed any form of ballot (including write-ins). This approach requires the employment of an anonymous channel between the voter and the tallying authorities to hide the identity of the user at the "ballot casting stage." Note that universal verifiability is missing and robustness is usually achieved by thresholding the authority.

Under the *homomorphic encryption* [16] paradigm, the ballots are encrypted and then "compressed" via a homomorphic encryption scheme into a tally. This compression property allows fast tallying, and is what makes this approach attractive. However the drawback is that pure "compressible" homomorphic encryption is not suitable to deal with write-in ballots.

Under the *mix-net* [12] paradigm, the tallying officials move the ballots between them and permute them in the process while changing their representation (e.g., partially decrypting them). Practical implementations of this approach in its fully robust form is still considered a slow tallying process.

*Offline Anonymous Electronic Cash (E-cash).* Most of the e-cash systems found in the literature makes use of *blind signatures*. In such systems, the users withdraw electronic coins, which consist of numbers generated by users and

blindly signed by the bank. Each signature represents a given amount. These coins are then spent in shops which can authenticate them by using the public signature key of the bank. The users retain anonymity in any transaction since the coins they use have been blindly signed. Existing schemes of this category are fruitful, some of the important ones are: [13, 14, 4, 10].

E-cash systems by *group signatures* recently received much attention. The idea is as follows: the group members in the group signature scheme forms a group of users. The bank (acting as the GM) is capable of issuing electronic coins (which are actually the ability to sign) to the users. When a user spends, he/she signs a group signature for the shop. The anonymity inherited from the group signature scheme provides privacy for the users. Examples: [24, 31, 25].

*Direct Anonymous Attestation (DAA).* In the context of the Trusted Computing Group (TCG), DAA is a solution to the following problem: The user of such a platform communicates with a verifier who wants to be assured that the user indeed uses a platform containing such a trusted hardware module, i.e., the verifier wants the trusted platform module (TPM) to authenticate itself. However, the user wants her privacy protected and therefore requires that the verifier only learns that she uses a TPM but not which particular one.

The first solution [21] has the drawback of requiring a TTP to be online in every transaction. Also, anonymity is lost when the TTP and the verifier collude. [6] solves the problem by making use of a group signature scheme variant based on the Camenisch-Lysyanskaya group signature scheme [7, 8]. Among other differences from the original scheme, the two crucial ones are (1) disabling anonymity revocation and (2) including a pseudonym in the signatures.

**Contributions.**

- We extend the short ring signature scheme construction of Dodis, et al.[18] to the first short LRS scheme construction, and reduce its security to a set of assumptions including a new hardness assumption, the Link Decisional RSA (LD-RSA) Assumption.
- We also extend [18]'s generic ring (resp. group) signature scheme constructions to their linkable version.
- Motivated by [22], who presented the first e-voting scheme that simultaneously achieved efficient tallying, universal verifiability, and write-in capability for typical voter distribution under which only a small portion writes in, we discuss that e-voting scheme constructed from LRS [23] schemes also achieve the same three properties even for all worst-case voter distributions.
- We discuss an efficient implementation of direct anonymous attestation [6] using linkable ring signatures, and the construction of an e-cash scheme using linkable group signatures.

### 1.1 Paper Organization

The paper is organized as follows: In Section 2, we give some preliminaries. Then we define linkable ring signatures and notions of security in Section 3. Constructions and their security analysis are presented in Section 4. We discuss three applications in Section 5. We finally conclude the paper in Section 6.

## 2 Preliminaries

We review literature results and introduce new terminologies.

**Strong RSA Assumption.** There exists no PPT algorithm which, on input a random $\lambda$-bit safe prime product $N$ and a random $z \in QR(N)$, returns $u \in \mathbb{Z}_N^*$ and $e \in \mathbb{N}$ such that $e > 1$ and $u^e = z(\mathrm{mod}N)$, with non-negligible probability and in time polynomial in $\lambda$.

**Simulation-Sound, Computationally Zero-Knowledge Proof System.** We adopt the definition of this concept from Bellare, et al. [2]. In a nutshell, it is a three-move zero-knowledge proof-of-knowledge system that is *simulation sound*, meaning that oracles specified in the security model can be successfully simulated, and that is *computational zero-knowledge*, meaning no PPT program can distinguish between real world and ideal world. For details, consult [2].

**Accumulator with One-Way Domain.** We adopt this concept introduced in Dodis, et al. [18]. Just a brief summary below. An *accumulator family* is a pair $(\{F_i\},\{X_i\})$ where $F_i$ is a family of functions whose member is s.t. $f : U_f \times X_i \rightarrow U_f$ and the accumulator family satisfies *efficient generation*, *efficient evaluation*, and *quasi-commutativity*. An accumulator is *collision resistant* if it is rare to have two different sequences accumulated to the same value.

An *accumulator with one-way domain* is a quadruple $(\{F_i\}, \{X_i\}, \{Z_i\}, \{\mathcal{R}_i\})$ where $(\{F_i\}, \{X_i\})$ is a collision-resistant accumulator, each $\mathcal{R}_i$ is a relation over $X_i \times Z_i$ satisfying *efficient verification*, *efficient sampling*, and *one-wayness*. For details consult the original paper [18].

**Definition 1 (The Link Decisional RSA (LD-RSA) Assumption).** *Let $N = pq = (2p' + 1)(2q' + 1)$ be a sufficiently large safe prime product. Let $g \in QR(N)$, with order $p'q'$. Let $p_0$, $q_0$, $p_1$, $q_1$ be four sufficiently large and distinct primes. Let $b$ be a fair coin flip, $n_0 = p_0q_1$, $n_1 = p_1q_1$. Given $n_0$, $n_1$, $g^{p_b+q_b}$, the* LD-RSA Assumption *says that no PPT algorithm can compute $b$ correctly with probability non-negligibly over 1/2.*

**Definition 2 (PK-bijectivity).** *Let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ be a one-way efficiently samplable NP-relation. Let $\mathcal{X}_\mathcal{R} = \{x \in \mathcal{X} : \text{ there exists } y \in \mathcal{Y}, (x, y) \in \mathcal{R}\}$. A mapping $\theta : \mathcal{X}_\mathcal{R} \rightarrow \mathcal{Z}$ is* PK-bijective with respect to $\mathcal{R}$ *if it satisfies the first two of the following three properties. It is* special PK-bijective *if it satisfies all three of the following properties.*

1. *The mapping $\theta$ is one-way and bijective.*
2. *Let $(x_0, y_0)$ and $(x_1, y_1)$ be two random samples of $\mathcal{R}$ with $y_0 \neq y_1$. Let $b \in \{0, 1\}$ be a fair coin flip and $z = \theta(x_b)$. Then there is no PPT algorithm who can, given $z$, distinguish between the the two cases $b = 0$ and $b = 1$ with success probability non-negligibly over half.*
3. *Let $(x_0, y_0)$ and $(x_1, y_1)$ be any two samples of $\mathcal{R}$ with $y_0 \neq y_1$. Let $b \in \{0, 1\}$ be a fair coin flip and $z = \theta(x_b)$. Then there is no PPT algorithm who can, given $z$, distinguish between the the two cases $b = 0$ and $b = 1$ with success probability non-negligibly over half.*

The property of PK-bijectivity will be important to the L-anonymity of linkable ring signatures. Details later.

## 3 Security Model

We give our security model and define relevant security notions.

### 3.1 Syntax

**Linkable Ring Signatures.** A linkable ring signature (LRS) scheme is a tuple (Init, LRKg, LRSig, LRVf, Link).

- Init takes as input the security parameter $1^\lambda$, and outputs system-wide public parameters param. Typically, param includes an *sk-pk* relation $\mathcal{R}$ which is efficiently samplable, an one-way NP-relation, lengths of keys, ..., etc.
- LRKg takes as inputs the security parameter $1^\lambda$, the group size $n$, and returns a tuple ($gpk$, $gsk$), where $gpk$ is group public key, $gmsk$ is group manager's secret key, and group secret key $gsk$ is an $n$-vector with $gsk[i]$ being the secret signing key for player $i$, $1 \leq i \leq n$. Often $gpk$ is also an $n$ vector with $(gsk[i], gpk[i]) \in \mathcal{R}$.
- LRSign takes inputs group public key $gpk$, a secret signing key $gsk[i]$ and a message $M$, returns a linkable ring signature $\sigma$ of $M$.
- LRVf takes inputs the group public key $gpk$, a message $M$, and a signature $\sigma$ for $M$, returns either 1 or 0 for valid or invalid.
- Link takes as inputs two valid signatures $\sigma$ and $\sigma'$, returns either 1 or 0 for linked or unlinked. It returns nothing or $\perp$ if the signatures are not both valid.

CORRECTNESS. An LRS scheme is *correct* if (1) LRVf($gpk$, $M$, LRSign($gpk$, $gsk[i]$, $M$))=1, and (2) Link(LRSign($gpk$, $gsk[i]$, $M$), LRSign($gpk$, $gsk[i]$, $M'$))=1 for all $gpk$, $gsk$, $i$, $M$, $M'$. The two checks are sometimes called *verification correctness* and *linking correctness*, resp.

### 3.2 Notions of Security

Security of LRS schemes has these aspects: unforgeability, L-anonymity and linkability. The following oracles define the attacker's capabilities.

- $sk_i \leftarrow \mathcal{CO}(pk_i)$. The *Corruption Oracle*, on input a public key $pk_i \in \mathcal{Y}$ that is an output of LRKg, returns the corresponding secret key $sk_i \in \mathcal{X}$.
- $\sigma \leftarrow \mathcal{SO}(gpk, s, M)$. The *Signing Oracle*, on input $gpk$, a designated signer $s$, returns a valid signature $\sigma$ which is computationally indistinguishable from one produced by LRSign using the real secret key $gsk[s]$ on message $M$.

*Remark*: An alternative approach is to exclude $s$ from $\mathcal{SO}$'s input and have $\mathcal{SO}$ randomly select the signer. We do not pursue that alternative here.

**Unforgeability.** Unforgeability for LRS schemes is defined in the following game between the Simulator $\mathcal{S}$ and the Adversary $\mathcal{A}$ in which $\mathcal{A}$ is given access to oracles $\mathcal{CO}$ and $\mathcal{SO}$:

1. $\mathcal{S}$ generates and gives $\mathcal{A}$ the system parameters $\mathsf{param}$.
2. $\mathcal{A}$ may query the oracles according to any adaptive strategy.
3. $\mathcal{A}$ delivers $gpk$, $M$, $\sigma_g$.

$\mathcal{A}$ wins the game if: (1) $\mathsf{LRVf}(gpk,M,\sigma_g)$=1, (2) all public keys in $gpk$ are outputs of $\mathsf{LRKg}$, none has been input to $\mathcal{CO}$, and (3) $\sigma_g$ is not an output of $\mathcal{SO}$ on any input containing $M$ as the message. $\mathcal{A}$'s *advantage* is its probability of winning.

**Definition 3 (unforgeability).** *An LRS scheme is* unforgeable *if no PPT adversary $\mathcal{A}$ has a non-negligible advantage in the game above.*

**L-Anonymity.** Anonymity for LRS schemes is defined in the following game in which $\mathcal{A}$ is given access to oracles $\mathcal{CO}$ and $\mathcal{SO}$:

**Game LA**

1. (*Initialization Phase*) $\mathcal{S}$ generates and gives $\mathcal{A}$ the system parameters $\mathsf{param}$.
2. (*Probe-1 Phase*) $\mathcal{A}$ queries the oracles with arbitrary interleaving.
3. (*Gauntlet Phase*) $\mathcal{A}$ gives $\mathcal{S}$ $gpk$, $s$, message $M$, where $gpk[s]$ has never been queried to $\mathcal{CO}$ and has never been the designated signer in any $\mathcal{SO}$ query. Then $\mathcal{S}$ flips a fair coin to select $b \in \{\mathsf{real}, \mathsf{ideal}\}$. Case $b$=$\mathsf{real}$: $\mathcal{S}$ queries $\mathcal{CO}$ with $gpk[s]$ to obtain $gsk[s]$ and compute $\sigma = \mathsf{LRSign}\ (gpk,gsk[s],M)$. Case $b$=$\mathsf{ideal}$: $\mathcal{S}$ computes $\sigma = \mathcal{SO}(gpk,s,M)$.
4. (*Probe-2 Phase*) $\mathcal{A}$ receives $\sigma$, queries the oracles adaptively, except that $gpk[s]$ cannot be queried to $\mathcal{CO}$ or be designated signer in any $\mathcal{SO}$ query.
5. (*End Game*) $\mathcal{A}$ delivers an estimate $\hat{b} \in \{\mathsf{real}, \mathsf{ideal}\}$ of $b$.

$\mathcal{A}$ wins the game if $\hat{b} = b$. Its *advantage* is its winning probability minus half.

**Definition 4 (L-Anonymity).** *An LRS scheme is* L-anonymous *if for no PPT adversary has a non-negligible advantage in Game LA.*

*Remark*: In this paper, the statistical distance between the $\mathsf{real}$ world and the $\mathsf{ideal}$ world is non-negligible. Therefore, we use a distinguishability test between $\mathsf{real}$ and $\mathsf{ideal}$. The other popular approach, distinguishing between two possible signers, is not suitable. Our attacker model is not fully active due to restrictions that the *gauntlet* public key (i.e. $gpk[s]$ in the Gauntlet Phase) cannot be queried.

**Linkability.** Linkability for LRS schemes is defined in the following game between the Simulator $\mathcal{S}$ and the Adversary $\mathcal{A}$ in which $\mathcal{A}$ is given access to oracles $\mathcal{CO}$ and $\mathcal{SO}$:

1. $\mathcal{S}$ generates and gives $\mathcal{A}$ the system parameters $\mathsf{param}$.
2. $\mathcal{A}$ queries the oracles according to any adaptive strategy.
3. $\mathcal{A}$ delivers $(gpk_i, M_i, \sigma_i)$, $i$=1,2.

$\mathcal{A}$ wins the game if (1) all public keys in $gpk_1 \cup gpk_2$ are outputs of $\mathsf{LRKg}$, and at most one has been queried to $\mathcal{CO}$. (2) $\mathsf{LRVf}(gpk_i, M_i, \sigma_i)$=1, $i$=1,2. and (3) $\mathsf{Link}(\sigma_1, \sigma_2)$=0. The Adversary's *advantage* is its probability of winning.

**Definition 5 (Linkability).** *An LRS scheme is* linkable *if no PPT adversary has a non-negligible advantage in winning the game above.*

**Security.** Summarizing we have:

**Definition 6 (Security of LRS Schemes).** *An LRS scheme is secure if it is unforgeable, L-anonymous, and linkable.*

## 4 Constructions of Linkable Ring Signature Schemes

In this section we present several LRS scheme constructions.

### 4.1 Generic Constructions Without Accumulators

We present two generic LRS scheme constructions with signature size $O(n)$. Let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ denote an efficiently-samplable, one-way NP-relation that is typical of *sk-pk* relations. Let $\theta_d : \mathcal{X} \to \tilde{\mathcal{Y}}$ denote a one-way bijective mapping. Two LRS schemes are constructed as follows:

- $\mathsf{LRKg}$: Upon input $1^\lambda$, output a pair of $n$-vectors $(gpk, gsk)$ where $(gsk[i], gpk[i]) \in \mathcal{R}$, each $i$, $1 \le i \le n$.
- $\mathsf{LRSign}$: Upon inputs $gpk$, $sk = gsk[s]$, message $M$, $\mathcal{R}$, and a one-way bijective mapping $\theta_d$, produces a linkable ring signature following either Eq (1) or Eq (2) below (thereby resulting in two different constructions):

$$\text{(LRS1) } SPK\{sk : (\vee_{1\le i\le n}(sk, gpk[i]) \in \mathcal{R}) \wedge (\tilde{y} = \theta_d(sk))\}(M), \text{ or} \quad (1)$$

$$\text{(LRS2) } SPK\{sk : \vee_{1\le i\le n}(((sk, gpk[i]) \in \mathcal{R}) \wedge (\tilde{y}_i = \theta_d(sk)))\}(M). \quad (2)$$

  We adopt the notation of [11] in the above.
- $\mathsf{LRVf}$: Straightforward. Accepts if the corresponding PoK is verified to be correct.
- $\mathsf{Link}$: For LRS1, examine the *linkability tag*, $\tilde{y}$, from each of the two input signatures and outputs $\mathsf{linked}$ if and only if they are equal. For LRS2, compare the lists of linkability tags of the signatures and outputs $\mathsf{linked}$ if and only if they contain at least one tag in common.

Theorem 1 in [18] implies the existence of efficient implementations of LRS1 and LRS2. The complexity of the signatures implied by the theorem is polynomially growing, but is high in practice. In Appendix A of the full version of this paper [32], we provide efficient instantiations of LRS1 and LRS2 that use two popular mechanisms to achieve 1-out-of-$n$ proof-of-knowledge, namely Cramer, et al.'s partial proof-of-knowledge [17] and Rivest, et al.'s ring structure [29].

### 4.2 Generic Construction with Accumulators

We construct a short LRS scheme, using methods motivated by [18]. Define $f(u, \{z_1, \cdots, z_n\}) \doteq f(\cdots(f(f(u, z_1), z_2), \cdots, z_n)$, the accumulation of $z_1, \cdots, z_n$. The signing algorithm of the scheme is given by the following:

– LRSign: Let $gpk$=$(y_1, \cdots, y_n)$, and $gsk[s] = x$. Upon inputs $gpk$, $gsk[s]$, message $M$, compute $v = f(u, \{y_1, \cdots, y_n\})$, and $w = f(u, \{y_1, \cdots, y_n\} \setminus \{y_s\})$. The signature is computed as $\sigma =$

$$\text{(LRS3) } SPK\{(w, y_s, x) : (y_s, x) \in \mathcal{R} \wedge f(w, y_s) = v \wedge \tilde{y} = \theta_d(x)\}(M) \quad (3)$$

### 4.3 The Short Linkable Ring Signature Scheme Construction

The generic construction in the previous section is further instantiated by the following parameter choices: Use the accumulator $f(u, x) = u^x \bmod N$ where $N = pq = (2p' + 1)(2q' + 1)$ is a sufficiently large safe prime product. Let $\mathcal{R}\{(2e_1e_2 + 1, (e_1, e_2)) : |e_1 - 2^\ell|, |e_2 - 2^\ell| < 2^\mu\}$. The parameters $\ell$ and $\mu$ are selected according to methods in [8] to ensure security against coalition attacks. We have in mind $\ell \approx \lambda/2$, $\mu$ sufficiently small, $p'$ and $q'$ at least $\lambda/2$ plus a few bits long. Let $\theta_d(e_1, e_2) = g_\theta^{e_1+e_2}$, where $g_\theta \in RQ(N)$ and is fairly generated. Given the above instantiations, LRS3 becomes:

$$\begin{aligned}\text{(LRS3')} \; SPK\{(w, x, e_1, e_2) : w^x = v \bmod N \wedge x = 2e_1e_2 + 1 \\ \wedge |e_1 - 2^\ell|, |e_2 - 2^\ell| < 2^\mu \wedge \tilde{y} = \theta_d((e_1, e_2))\}(M)\end{aligned} \quad (4)$$

Writing down the $\Sigma$-protocol explicitly, LRS3' becomes

$$\begin{aligned}\text{(LRS4)} \quad & SPK\{(r, w, x, e_1, e_2) : T_1 = g^r \wedge T_2 = g^x h^r \wedge T_3 = g^{e_2} s^r \\ & \wedge T_4 = wy^r \wedge T_5 = g^{e_1} t^r \wedge T_1^x = g^{a_1} \wedge T_1^{e_2} = g^{a_2} \wedge T_4^x = vy^{a_1} \\ & \wedge T_5^{2e_2} g = g^x t^{2a_2} \wedge |e_1 - 2^\ell|, |e_2 - 2^\ell| < 2^\mu \wedge \tilde{y} = g_\theta^{e_1+e_2}\}(M)\end{aligned} \quad (5)$$

where $a_1 = xr$, $a_2 = e_2 r$.

The above instantiates a $O(1)$-sized linkable ring signature scheme, provided the list of public keys $y_1, \cdots, y_n$ is not included in the signature.

### 4.4 Security Theorems

We give in this section theorems on the security of some of our constructions and leave the proofs in Appendix B in the full version of this paper [32].

**Theorem 1.** *Assume $\theta_d$ is a special PK-bijection. The linkable ring signature LRS3 (resp. LRS3', LRS4) is*

1. correct.
2. L-anonymous *if Eq. (3) (resp. (4), (5)) is a simulation-sound, computational zero-knowledge proof system.*
3. unforgeable *if Eq. (3) (resp. (4), (5)) is a sound non-interactive proof system.*
4. linkable *if Eq. (3) (resp. (4), (5)) is a sound non-interactive proof system.*

**Theorem 2.** *If one-way functions exist and $\theta_d$ is a special PK-bijection, then there exist an efficient instantiation of the linkable ring signature LRS3 (resp. LRS3', LRS4) which has correctness, unforgeability, linkability, and L-anonymity.*

The linkable ring signature LRS3 (resp. LRS3') instantiated by the proofs of [20, 19, 18] are asymptotically polynomial time. But they are too complex for practical system parameters. The linkable ring signature instantiation LRS4 is efficient even for practical system parameters. This construction closely followed the short ring signature of [18], and therefore inherits several of their properties. In particular, there is no rigorous proof of unforgeability and unlinkability. We have only been able to obtain the following security reduction of L-anonymity.

**Theorem 3.** *The linkable ring signature LRS4 has L-anonymity provided the DDH Assumption and the LD-RSA Assumption hold in the RO Model.*

### 4.5 Discussions

*Common errors.* If we change the mapping $\theta_d((e_1, e_2)) = g_\theta^{e_1+e_1}$ to $\theta_{d,2}((e_1, e_2)) = g_\theta^{e_1}$, then linkability is lost beause a single user in possession of an RSA secret $(e_1, e_2)$ can produce two unlinked signatures with $\tilde{y} = g_\theta^{e_1}$ and $\tilde{y} = g_\theta^{e_2}$. If we replace with $\theta_{d,3}((e_1, e_2)) = g_\theta^{e_1,e_2}$, then L-anonymity is lost because the proof system LRS4 Eq. (5) is no longer computational zero-knowledge. However, a (probably) secure alternative choice is $\theta_{d,4}((e_1, e_2)) = (g_\theta^{e_1}, g_\theta^{e_2})$. Note that $\theta_{d,2}$ while $\theta_{d,3}$ are not special PK-bijective.

A related security requirement is to that, given a random sample $y_1$, it is hard to compute $y_2$ such that there exist $x_1$, $x_2$, satisfying $(x_1, y_1)$, $(x_2, y_2) \in \mathcal{R}$, $\theta_d(x_1) = \theta_d(x_2)$. This stronger concept may be needed in further study of the current topic, but it is not needed in the present paper.

It is straightforward to extend our LRS's to *linkable group signatures* [26, 27]. Simply also escrow the user identity (or the user public key) to an Open Authority (OA) in the signatures. The escrow can be done by verifiably encrypt the identity (or public key) to the OA by methods in [2], for example.

*Dynamic group setting.* We have used the static group setting, where memberships to the group remain unchanged during the course of signature generation and verification. Our scheme and security model can be adapted for dynamic group settings. The Join operation can be implemented. The group membership manager maintains a list of all members' public keys.

*Other kinds of user key pairs.* LRS schemes in which user key pairs are from cryptosystems other than RSA can be similarly constructed. We have in mind modifications to the usual form of public key for added security against potential coalition attacks. E.g., for DL user key pairs $\mathcal{R} = \{(x, y = 2g^x + 1)\}$, for pairings $\mathcal{R} = \{(x, \text{first coordinate of } P^x)\}$, etc.

## 5 Applications

We give in this paper a brief discussion on applying LRS schemes to E-voting, offline anonymous electronic cash and direct anonymous attestation. For a more detailed discussion, refer to the full version of this paper [32].

### 5.1 E-Voting

Remarkable advances in group/ring signatures in recent years have given new options to e-voting scheme constructions. In fact, many papers on group/ring signatures have included e-voting as applications. Using group/ring signatures contributes to a new paradigm of e-voting construction.

Kiayias and Yung [22] hybridized homomorphic encryption and mix-net to achieves simultaneously (1) efficient tallying, (2) universal verifiability and (3) write-in capability under typical voter distribution where only a small proportion of voters write-in. E-voting schemes constructed from LRS schemes are capable of achieving the same even under worst-case voter distributions.

### 5.2 Direct Anonymous Attestation (DAA)

In essence, DAA [6] is a group signature without revocability, and with an additional feature of *rogue tagging*. Double signers can be detected, or linked, yet their identities are not revealed. When a double signer is detected, a *rogue tag* is produced to prevent it from signing again: future signatures (attestations) identified with a known rogue tag is not accepted. Double signers of different transactions with the same *basename*, $\mathsf{bsn}$, are detected. But signing twice with different basename is not detected.

The linkable ring signature is ideally suited to implementing DAA. It is a group signature without revocation. Its linkability feature can be used to detect double signers, and when linked output the linkability tag, $\tilde{y} = g_\theta^{sk}$, as the rogue tag. Future signatures whose $\tilde{y}$ equals a known rogue tag is not accepted. The value $g_\theta$ can be made a function of the basename but not the transaction, e.g. $g_\theta = Hash(\mathsf{bsn}, \cdots)$. Then double signing on different transactions with same basename is linked, while double signing on different basename will not be linked.

### 5.3 E-Cash

Our LRS scheme (LRS4) can also be used to construct an e-cash scheme. It serves as a new alternative to e-cash schemes of the "group signature approach", as described in the introduction.

Double spenders of the e-cash are detected as double signers of the linkable ring signature scheme. However, methodologies differ after detection. In *non-accusatory* linkability, the suspect can only be *tagged* and prevented from further double spending afterwards. The drawbacks are time delay to effective tagging and small punishment for the offense. In *accusatory* linkability, the linking algorithm outputs a suspect. But there are issues of *non-slanderability* and *deniability* that can be quite subtle.

## 6 Conclusion

In this paper, we have presented generic LRS scheme constructions with provable security and the first short LRS scheme construction, with security reducible to the LD-RSA problem. We have shown a generic LGS scheme construction with provable security. Also, we have given the first bandwidth-conserving e-voting scheme that simultaneously achieves efficient tallying, universal verifiability, and write-in capability, even in the worst case of voter distribution. We have discussed how to implement DAA and e-cash systems using LRS/LGS schemes.

**Acknowledgement.** The authors would like to thank the anonymous reviewers for their comments.

## References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO 2000*, pages 255–270. Springer-Verlag, 2000.
2. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA 2005*, 2005. To appear.
3. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Eurocrypt 2004*, volume 3027 of *LNCS*. Springer–Verlag, 2004.
4. S. Brands. Untraceable off-line cash in wallet with observers. In *Proceedings of the 13th annual international cryptology conference on Advances in cryptology*, pages 302–318. Springer-Verlag, 1994.
5. E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. In *Crypto'02*, volume 2442 of *LNCS*, pages 465–480. Springer-Verlag, 2002.
6. E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. Cryptology ePrint Archive, Report 2004/205, 2004. http://eprint.iacr.org/
7. J. Camenisch and A. Lysyanskaya. Efficient non-transferable anonymous multishow credential system with optional anonymity revocation. In *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer-Verlag, 2001.
8. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN 2002*, volume 2576 of *LNCS*, pages 268–289. Springer-Verlag, 2003.
9. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps, 2004.
10. J. Camenisch, J.-M. Piveteau, and M. Stadler. An efficient fair payment system. In *Proceedings of the 3rd ACM conference on Computer and communications security*, pages 88–94. ACM Press, 1996.

11. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In *CRYPTO'97*, pages 410–424. Springer-Verlag, 1997.
12. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.
13. D. Chaum. Blind signatures for untraceable payments. In *Crypto'82*, pages 199–203. Plenum Press, 1982.
14. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Proceedings on Advances in cryptology*, pages 319–327. Springer-Verlag, 1990.
15. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer-Verlag, 1991.
16. J. D. Cohen and M. J. Fischer. A robust and verifiable cryptographyically secure election scheme. In *FOCS 85*, pages 372–382, 1985.
17. R. Cramer, I. Damgard, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO'94*, pages 174–187. Springer-Verlag, 1994.
18. Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 609–626. Springer-Verlag, 2004.
19. U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. In *CRYPTO'89*, pages 526–544, 1989.
20. O. Goldreich, S. Micali, and A. Wigderson. Proof that yields nothing but their validity or all languages in NP have zero-knowledge proof system. *Journal of the ACM*, 38(3):691–729, 1991.
21. Trusted Computing Group. Trusted computing platform alliance (tcpa) main specification, version 1.1a. republished as trusted computing group (tcg) main specifcation, version 1.1b, 2001. http://www.trustedcomputinggroup.org
22. A. Kiayias and M. Yung. The vector-ballot e-voting approach. In *FC 2004*, volume 3110 of *LNCS*, pages 72–89. Springer-Verlag, 2004.
23. J. K. Liu, V. K. Wei, and D. S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In *ACISP'04*, volume 3108 of *LNCS*, pages 325–335. Springer-Verlag, 2004.
24. A. Lysyanskaya and Z. Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In *FC'98*, pages 184–197. Springer-Verlag, 1998.
25. G. Maitland and C. Boyd. Fair electronic cash based on a group signature scheme. In *ICICS'01*, volume 2229 of *LNCS*. Springer-Verlag, 2001.
26. T. Nakanishi, T. Fujiwara, and Watanabe H. A linkable group signature and its application to secret voting. In *4th Int'l Symp. on Communicatin Theory and Appl.*, 1997.
27. T. Nakanishi, T. Fujiwara, and Watanabe H. A linkable group signature and its application to secret voting. *Trans. of Information Processing Society of Japan*, 40(7):3085–3096, 1999.
28. L. Nguyen and R. Safavi-Naini. Efficient and provably secure trapdoor-free goup signature schems from bilinear pairings. In *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 372–386. Springer-Verlag, 2004.
29. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT 2001*, pages 552–565. Springer-Verlag, 2001.
30. I. Teranishi, J. Furukawa, and K. Sako. $k$-times anonymous authentication. In *Asiacrypt 2004*, volume 3329 of *LNCS*, pages 308–322. Springer-Verlag, 2004.
31. J. Traoré. Group signatures and their relevance to privacy-protecting off-line electronic cash systems. In *ACISP'99*, pages 228–243. Springer-Verlag, 1999.

32. Patrick P. Tsang and Victor K. Wei. Short linkable ring signatures for e-voting, e-cash and attestation. Cryptology ePrint Archive, Report 2004/281, 2004. http://eprint.iacr.org/.
33. Patrick P. Tsang, Victor K. Wei, Man Ho Au, Tony K. Chan, Joseph K. Liu, and Duncan S. Wong. Separable linkable threshold ring signatures. In *Indocrypt 2004*, volume 3348 of *LNCS*, pages 384–398. Springer-Verlag, 2004.
34. Victor K. Wei. Tracing-by-linking group signatures. Cryptology ePrint Archive, Report 2004/370, 2004. http://eprint.iacr.org/

# Tracing Traitors by Guessing Secrets. The $q$-Ary Case

Marcel Fernandez, Miguel Soriano, and Josep Cotrina*

Department of Telematics Engineering, Universitat Politècnica de Catalunya, C/ Jordi Girona 1 i 3. Campus Nord, Mod C3, UPC, 08034 Barcelona, Spain
{marcelf, soriano, jcotrina}@mat.upc.es

**Abstract.** In this paper we present, by solving a variant of the guessing secrets problem defined by Chung, Graham and Leighton [3], a sequential traitor tracing scheme equipped with an efficient identification algorithm. Sequential traitor tracing schemes are used to detect piracy in multimedia content broadcast systems, where the traitors illegally rebroadcast the content they receive to unauthorized users.

## 1 Introduction

In the original "I've got a secret" TV game show [4], a contestant with a secret was questioned by four panelists. The questions were directed towards guessing the secret. A prize money was given to the contestant if the secret could not be guessed by the panel.

A variant of the game called "guessing secrets" was defined by Chung, Graham and Leighton in [3]. In their variant , there are two players **A** and **B**. Player **A** draws a subset of $c \geq 2$ secrets from a set $S$ of $N$ objects. Player **B** asks a series of boolean (binary) questions. For each question asked **A** can adversarially choose a secret among the $c$ secrets, but once the choice is made he must answer truthfully. The goal of player **B** is to come up with an strategy, that using as few questions as possible allows him to unveil the secrets.

The problem of guessing secrets is related to several topics in computer science such as efficient delivery of Internet content [3] and the construction of schemes for the copyright protection of digital data [1]. In [1] Alon, Guruswami, Kaufman and Sudan realized there was a connection between the guessing secrets problem and error correcting codes. Using this connection they provided a solution to the guessing secrets problem equipped with an efficient algorithm to recover the secrets.

In this paper we modify the condition established by Chung, Graham and Leighton, and allow questions over a larger alphabet. We call this modified version the *$q$-ary guessing secrets problem*. In our version of the problem, player **A**

* This work has been supported in part by the Spanish Research Council (CICYT) Projects TIC2002-00818 (DISQET) and TIC2003-01748 (RUBI) and by the IST-FP6 Project 506926 (UBISEC).

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 61–73, 2005.

first chooses a set of $c$ secrets, $\mathbf{u}^1, \ldots, \mathbf{u}^c$ from the pool of objects $S$. The goal of player **B** is to guess **A**'s secrets, by asking a series of questions. The difference is that now the questions are not restricted to have binary answers, but are over a larger alphabet that, without loss of generality, we take to be the finite field $\mathbb{F}_q$. For each question asked, player **A** can choose adversarially one of the secrets. but once the choice is made, he must answer truthfully.

The series of questions that **B** asks and that should allow him to guess as many secrets as is possible, will be called a *strategy*. We will impose the requirement that any strategy that **B** uses must be *invertible*, that is, given the sequence of answers there exists an efficient algorithm to recover the secrets.

Our contribution starts in Section 3 where we show that the adopted strategy will lead us to a class of error correcting codes with large minimum distance called *c-traceability* codes, in particular to the Martirosyan–van Trung code [9], that in turn leads to *sequential traitor tracing* schemes [7].

The technical centerpoint of the paper focuses on the second part of the $q$-ary guessing secrets problem. Since the $q$-ary guessing secrets problem leads us to error correcting codes, we will discuss how to recover the secrets by using soft-decision decoding techniques.

The paper is organized as follows. Section 2 introduces the coding theory concepts that will be used. In Section 3, a formal description of the game of guessing secrets is presented. In Section 4, we present a tracing algorithm based on soft-decision decoding techniques. Finally, in Section 5 we give our conclusions.

## 2 Overview of Coding Theory Concepts

If $C$ is a set of vectors of a vector space, $\mathbb{F}_q^n$, then $C$ is called a *code*. The field, $\mathbb{F}_q$ is called the *code alphabet*. A code $C$ with length $n$, size $|C| = M$, and alphabet $\mathbb{F}_q$ is denoted as a $(n, M)$ $q$-ary code. We will also denote code $C$ as a $(n, M, q)$ code. The *Hamming distance* $\mathbf{d}(\mathbf{a}, \mathbf{b})$ between two words $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^n$ is the number of positions where $\mathbf{a}$ and $\mathbf{b}$ differ. The *minimum distance* of $C$, denoted by $d$, is defined as the smallest distance between two different codewords. A code $C$ is a *linear code* if it forms a subspace of $\mathbb{F}_q^n$. A code with length $n$, dimension $k$ and minimum distance $d$ is denoted as a $[n, k, d]$-code.

We use the terminology in [8] to describe *identifiable parent property* (IPP) codes and *traceability codes*. Let $U \subseteq C$ be any subset of codewords such that $|U| = c$. The set of *descendants* of $U$, denoted $\mathbf{desc}(U)$, is defined as

$$\mathbf{desc}(U) = \{\mathbf{v} \in \mathbb{F}_q^n \ : v_i \in \{a_i : \mathbf{a} \in U\}, 1 \leq i \leq n\}.$$

Let $\mathbf{z} \in \mathbf{desc}(U)$ and let $\mathbf{u}$ be a codeword such that $\mathbf{u} \in U$, then $\mathbf{u}$ is called a *parent* of $\mathbf{z}$.

Codes with the identifiable parent property (IPP) were introduced in [5]. For a code $C$ and an integer $c \geq 2$, let $U_i \subseteq C$, $i = 1, 2, \ldots, t$ be all the subsets of $C$ such that $|U_i| \leq c$. A code $C$ is a $c$-IPP (identifiable parent property) code, if for every descendant $\mathbf{z}$ of a subset of at most $c$ codewords, we have that

$\bigcap_{\{i:\mathbf{z}\in \mathrm{desc}(U_i)\}} U_i \neq \emptyset$. In other words, $C$ is a $c$-IPP code if the intersection of all possible sets of parents $U_i$ of $\mathbf{z}$ is non-empty.

An important subclass of IPP codes are *traceability* (TA) codes. For $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ we can define the set of *matching positions* between $\mathbf{x}$ and $\mathbf{y}$ as $\mu(\mathbf{x}, \mathbf{y}) = \{i : x_i = y_i\}$. Let $C$ be a code, then $C$ is a $c$-TA code if for all $i$ and for all $\mathbf{z} \in \mathbf{desc}_c(U_i)$, there is at least one codeword $\mathbf{u} \in U_i$ such that $|\mu(\mathbf{z}, \mathbf{u})| > |\mu(\mathbf{z}, \mathbf{v})|$ for any $\mathbf{v} \in C \backslash U_i$.

**Theorem 1 ([8]).** *Let $C$ be an $(n, M, q)$ code with minimum distance $d$, if $d > n(1 - 1/c^2)$ then $C$ is a $c$-traceability code($c$-TA).*

### 2.1 Asymptotically Good TA Codes

We now present an "asymptotically good" family of TA codes due to van Trung and Martirosyan [9]. Their construction is based on IPP code concatenation.

A concatenated code is the combination of an *inner* $[n_i, k_i, d_i]$ $q_i$-ary code, $C_{inn}$, $(q_i \geq 2)$ with an *outter* $[n_o, k_o, d_o]$ code, $C_{out}$ over the field $\mathbf{F}_{q_i^{k_i}}$. The combination consists in a mapping $\phi$, from the elements of $\mathbf{F}_{q_i^{k_i}}$ to the codewords of the inner code $C_{inn}$, $\phi : \mathbf{F}_{q_i^{k_i}} \rightarrow C_{inn}$ that results in a $q_i$-ary code of length $n_i n_o$ and dimension $k_i k_o$.

**Theorem 2.** *[9] Let $c \geq 2$ be an integer. Let $n_0 > c^2$ be an integer and let $s_0$ be an integer with the prime factorization $s_0 = p_1^{e_1} \cdots p_k^{e_k}$ such that $n_0 \leq p_i^{e_i}$ for all $i = 1, \ldots, k$. Then, for all $h \geq 0$ there exists an $(n_h, M_h, s_0)$ $c$-IPP code, where*

$$n_h = n_{h-1} \cdot n^*_{h-1},\ M_h = M_{h-1}^{\lceil \frac{n^*_{h-1}}{c^2} \rceil}, n^*_{h-1} = n^*_{h-2}{}^{\lceil \frac{n^*_{h-2}}{c^2} \rceil},\ M_0 = s_0^{\lceil \frac{n_0}{w^2} \rceil},\ n^*_0 = n_0^{\lceil \frac{n_0}{w^2} \rceil}$$

The codes in Theorem 2 have the best known asymptotic behavior. As stated in [9], the results in Theorem 2 can be applied to TA-codes, if the IPP codes used in the recursion are replaced by TA-codes.

### 2.2 The Koetter-Vardy Soft-Decision Decoding Algorithm

In this section we give a brief overview of the Koetter-Vardy (KV) soft-decision decoding algorithm [6], for Reed-Solomon codes.

In *list decoding*, the decoder outputs a list of codewords, thus offering a potential way to recover from errors beyond the error correction bound of the code. In *soft-decision* decoding, the decoding process takes advantage of "side information" generated by the receiver and instead of using the received word symbols, the decoder uses probabilistic reliability information about these received symbols. The simplest form of soft-decision decoding is called *errors-and-erasures* decoding. An erasure is an indication that the value of a received symbol is in doubt. In this case, when dealing with a $q$-ary transmission, the decoder has $(q+1)$ output alternatives: the $q$ symbols from $\mathbb{F}_q$, $\gamma_1, \gamma_2, \ldots, \gamma_q$ and $\{*\}$, where the symbol $\{*\}$ denotes an erasure.

The input to the Koetter-Vardy algorithm is a $q \times n$ matrix, called the *reliability matrix* and denoted by $\mathcal{R}$. If we order the elements of the field $\mathbb{F}_q$ as $(\gamma_1, \gamma_2, \ldots, \gamma_q)$, then the entry $r_{i,j}$ of $\mathcal{R}$ is denotes the probability that the symbol that was sent in the $j$th position of the codeword is $\gamma_i$.

We are interested in knowing what codewords the KV algorithm will return. With this intention, given two $q \times n$ matrices $A$ and $B$ over the same field, the following product is defined

$$\langle A, B \rangle := trace(AB^T) = \sum_{i=1}^{q} \sum_{j=1}^{n} a_{i,j} b_{i,j} \tag{1}$$

Also a word $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ over $\mathbb{F}_q$ can be represented by the $q \times n$ matrix $[\mathbf{v}]$, with entries $[\mathbf{v}]_{i,j}$ defined as follows

$$[\mathbf{v}]_{i,j} := \begin{cases} 1 \text{ if } v_j = \gamma_i \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

In [6] Koetter and Vardy state the following theorem:

**Theorem 3 ([6]).** *If codeword* $\mathbf{u}$ *is transmitted, word* $\mathbf{v}$ *is received and the entries of reliability matrix* $\mathcal{R}$ *are the* $r_{i,j}$ *as defined in this section, then the KV soft-decision decoding algorithm outputs in polynomial time a list that contains the sent codeword* $\mathbf{u} \in \mathrm{RS}(n, k)$ *if*

$$\frac{\langle \mathcal{R}, [\mathbf{u}] \rangle}{\sqrt{\langle \mathcal{R}, \mathcal{R} \rangle}} \geq \sqrt{n - d} + o(1) \tag{3}$$

*where* $o(1)$ *is a function that tends to zero in the asymptotic case.*

### 2.3 Sequential Traitor Tracing Schemes

The term *traitor tracing* was first introduced in [2], and broadly speaking consists in a set of mechanisms that protect a digital multimedia content distributor against piracy acts committed by dishonest authorized users. As an application example, we can consider pay-TV systems, where each authorized user is given a set of keys that allows her to decrypt the content. These keys are usually contained in a decoder box. In a collusion attack, a coalition of dishonest users (traitors) get together, and by combining some of their keys, they construct a pirate decoder that tries to hide their identities. If the pirate decoder is found, a traitor tracing scheme allows the distributor to identify at least one of the guilty users, using the keys inside the decoder.

In a sequential traitor tracing scheme, the content is divided into *segments*. Let $W = \{w_1, w_2, \ldots, w_M\}$ be the set of users and let $Q = \{1, 2, \ldots, q\}$ be the mark alphabet. Using $Q$ together with a $q$-ary watermarking scheme, $q$ versions of each segment are obtained. For each segment, one of the versions of the segment is associated with a particular user according to an $M \times n$ array called a *mark allocation table* $T$. More precisely, the entry in $T(i, j)$ is the version of segment

$j$ that is allocated to user $w_i$. Therefore, the $j$th column of the mark allocation table is used by the distributor to assign segment versions to users in the $j$th time interval.

A coalition of dishonest users $U \subset W$ committs piracy by choosing one of their segment versions and rebroadcasting it. The distributor (tracer) intercepts the rebroadcasted content, and for each segment extracts the mark and appends it to a sequence $\mathbf{z}$, called the *feedback sequence*, in the following way: $\mathbf{z}$ is initially an empty sequence ($\mathbf{z}_0()$). In segment $j$ the distributor appends the extracted mark $z_j$ to the sequence $\mathbf{z}_{j-1}$ to construct the sequence $\mathbf{z}_j = (z_1, z_2, \ldots, z_j)$. For each segment position $j$, we define the set $V_j(U) = \{z_j | z_j \in T(i,j) : w_i \in U\}$. A *c-feedback sequence* is a feedback sequence $\mathbf{z} = (z_1, \ldots, z_n)$ that is formed by a coalition $U$ of size at most $c$. In this case, we have that $z_j \in V_j(U)$ for $j = 1, \ldots, n$. Traitors are identified by looking at the feedback sequence a sufficient number of segments. Whenever a traitor is traced, he is immediately removed from the system. The distributor continues monitorizing the rebroadcasted sequence and disconnects all the subsequent identifiable traitors. After the observation of $n$ segments, all traitors are traced and the algorithm converges.

A sequential traitor tracing scheme, $c$-TA scheme, consists of a mark allocation table $T$ and a tracing algorithm $A$, such that:

1. $T = (t_{ij})$ is an $M \times n$ array with entries from $Q$.
2. $A$ is a mapping $A : Q^* \rightarrow 2^U$ with the property that for any *c-feedback sequence* $\mathbf{z}$, there exists a sequence of integers $0 < d_1 < d_2 < \cdots < d_k \leq n$, $(k \leq c)$ such that

$$A(F_j) = \begin{cases} U_j, \emptyset \neq U_j \subseteq U, \ j = d_1, d_2, \ldots, d_k \text{ and } \bigcup_{l=1}^{k} U_l = U \\ \emptyset, \quad \text{otherwise} \end{cases}$$

The quantity $d_k$ is called the *convergence length* of the tracing process, and is the maximum number of steps needed in the algorithm to identify up to $c$ colluders.

**Constructions of $c$-TA Schemes Using Error Correcting Codes.** In [7] it is shown that sequential traitor tracing schemes can be constructed using $q$-ary error correcting codes.

**Theorem 4.** *[7] Let $C$ be an $(n, M, q)$ error correcting code with minimum distance $d$. If $d > n - \frac{n}{c^2} + \frac{1}{c}$ then $C$ is a sequential $c$-traceability scheme which converges in $d = c(n - d + 1)$ steps.*

The following corollary will allow us to use the terms $c$-TA scheme and $c$-TA code indistinctly.

**Corollary 1.** *[7] Mark allocation table of a sequential $c$-TA scheme is a $c$-TA code.*

The lemma below, gives a condition to identify traitors that created a given $c$-feedback sequence.

**Lemma 1.** *[7] Let $C$ be an $(n, M, q)$ error correcting code with minimum distance $d$, let $\mathbf{z}_j = (z_1, z_2, \ldots, z_j)$, $j \geq c(n-d)+1$ be a $c$-feedback sequence produced by $U \subseteq C$, $|U| \leq c$, and let $\mu(\mathbf{z}_j, \mathbf{u})$ denote the number of agreements between $\mathbf{z}_j$ and the codeword $\mathbf{u}$. If $\mu(\mathbf{z}_j, \mathbf{u}) = c(n-d)+1$, then $\mathbf{u} \in U$.*

## 3 Guessing Secrets and Error-Correcting Codes

In this section, we give a strategy for the $q$-ary guessing secrets problem that we defined in Section 1.

We recall, that in our version of the problem player **A** has drawn $c \geq 2$ secrets from a pool $\mathcal{S}$ of $N$ objects, and player **B** has to guess the secrets by asking questions whose answers belong to a $q$-ary alphabet.

First of all, we observe that guessing all the secrets is a very strong requirement for player **B**. In fact, in the worst case situation, the best that **B** can hope for is to guess at most one of the $c$ secrets. This is because, without breaking any rule, **A** can always answer according to the same secret. So the requirement we impose on **B** is that he should be able to recover *at least one* of the secrets chosen by **A**.

We model each question **B** asks as a function $p_i$ from the pool of objects to the potential answers, $p_i : S \rightarrow \mathbb{F}_q$. In this case, for every question $p_i$, $1 \leq i \leq n$, that **B** asks, each object $\mathbf{u}$ in the pool has an associated sequence of answers $\mathbf{u} = (u_1, \ldots, u_n)$, where $u_i := p_i(\mathbf{u})$, $u_i \in \mathbb{F}_q$ and $n$ is the number of questions asked. We call the sequence $\mathcal{C}$, of functions $p_i$, $1 \leq i \leq n$ **B**'s strategy. A strategy will solve the $q$-ary guessing secrets problem, if from the answers to the $p_i$ questions we are able to "reduce" all possible sets of $c$ secrets down to one of the $c$ secrets chosen by **A**.

In the rest of the paper, we will refer to the sequence of answers, given by **A** about the $c$ secrets he holds, as a *reply*, In other words, a reply is an ordered sequence of length $n$ of symbols from the field $\mathbb{F}_q$.

Since every secret $\mathbf{u}$ can be represented by a $q$-ary vector of length $\lceil \log_q |S| \rceil$, we can define the mapping $\mathcal{C}$ from the universe of objects to the sequences of answers as, $\mathcal{C} : \mathbb{F}_q^{\lceil \log_q |S| \rceil} \rightarrow \mathbb{F}_q^n$. The mapping $\mathcal{C}$ illustrates the connection between strategies and codes, since $\mathcal{C}$ maps $q$-ary sequences of length $\lceil \log_q |S| \rceil$ into sequences of length $n$. Since in general $n > \lceil \log_q |S| \rceil$, then the mapping $\mathcal{C}$ adds redundancy or encodes. Note that this reasoning allows us to refer to an strategy using its associated code. Therefore, in the rest of the paper, we will unambiguously use the terms strategy and code. Moreover, we will also refer to the codeword associated with a secret as simply a secret.

We will now infer the parameters of such codes. The length of the code will be equal to the number of questions $n$. Since **A** holds $c$ secrets $U = \{\mathbf{u}^1, \ldots, \mathbf{u}^c\}$, then at least $\lceil n/c \rceil$ of the answers will correspond to one of the secrets, say $\mathbf{u}^j$. To guarantee the identification of secret $\mathbf{u}^j$, we have to ensure that the reply

given by $\mathbf{A}$ about the secrets in $U$ only agrees with at most $\lceil n/c \rceil - 1$ of the corresponding answers associated with any of the secrets, say $mathbfv$, not in $U$. More precisely, let us denote by $\mathbf{d}(\mathbf{a}, \mathbf{b})$ the number of answers that are different between secrets $\mathbf{a}$ and $\mathbf{b}$, let us also define $d = \min_{(\mathbf{a},\mathbf{b})\in\mathcal{S}} \mathbf{d}(\mathbf{a}, \mathbf{b})$.

Using this notation, we need to ensure that for $\mathbf{u}^j$ and for all $\mathbf{v} \notin U$

$$\sum_{\mathbf{u}^i \in U} (n - d(\mathbf{u}^i, \mathbf{v})) \leq c(n-d) < \frac{n}{c} \leq \lceil \frac{n}{c} \rceil$$

so

$$d > n - \frac{n}{c^2} \tag{4}$$

We observe that the parameter $d$, indicates the minimum distance of the code associated with the strategy, therefore according to Theorem 1 any code associated with a strategy must be a $c$-TA code. Using the results in Section 2.1, we can assert that the Martirosyan-van Trung family of codes, provide the best known possible strategy for the $q$-ary guessing secrets problem.

### 3.1 Guessing Secrets Versus $c$-TA Codes Versus Sequential $c$-TA Schemes

From the above reasoning, we see that comparing (4) with Theorem 1, a $c$-TA code solves the $q$-ary guessing secrets problem. Moreover, from Corollary 1 we see that $c$-TA codes and sequential $c$-TA schemes are equivalent. Therefore, we have that the terms secrets, parents and traitors are also equivalent, as reply, descendant and $c$-feedback sequence also are. With this in mind, below we will mainly use the terms secrets and reply.

## 4 Tracing Algorithm for the Martirosyan-Van Trung Code

In Section 1 we insisted that a useful strategy should be invertible, that is, given a reply we must be able to trace back the secrets in an efficient manner.

Now we show that if the strategy for the $q$-ary guessing secrets problem is based on a $c$-traceability code, then the strategy is invertible since, for any reply, the Koetter-Vardy algorithm can be used to trace back the secrets. As we pointed out in Section 3, we cannot expect to find all secrets, since player $\mathbf{A}$ may choose to give few answers about them. So given a reply, we call any secret that is provably identifiable as one of $\mathbf{A}$'s secret a *positive secret*. The condition for a secret to be a positive secret is given in Theorem 5 below.

**Lemma 2.** *Let $C$ be a $c$-traceability $(n, M, q)$ code with minimum distance $d$ used as a strategy in the $q$-ary guessing secrets problem, then given a reply there always exists a secret (codeword) in $C$ that agrees with the reply in at least $c(n-d)+1$ answers.*

*Proof.* Since there are at most $c$ secrets, one of them must contribute with at least $\lceil n/c \rceil$ answers to in the creation of the reply, so it suffices to prove that $c(n-d)+1 \leq \lceil n/c \rceil$. If $d > n - n/c^2$ (by Theorem 1 existence of a $c$-traceability code) this is clearly the case. ∎

**Theorem 5.** *[7] Let $C$ be a $c$-traceability $(n, M, q)$ code with minimum distance $d$, used as a strategy in the $q$-ary guessing secrets problem, if a secret (codeword) agrees in at least $c(n-d)+1$ positions with a given reply then this secret must be a positive secret.*

*Proof.* The existence of the secret follows from Lemma 2. If the code has minimum distance $d$ then two secrets can agree in at most $n-d$ positions, therefore a reply agrees is at most $c(n-d)$ positions with any other secret not in the coalition. Then any secret that agrees with the reply in at least $c(n-d)+1$ positions is a positive secret. ∎

**Corollary 2.** *Let $C$ be a $c$-traceability $(n, M, q)$ code with minimum distance $d$, used as a strategy in the $q$-ary guessing secrets problem,. Let $\mathbf{z}$ be a reply. Suppose that $j$ already identified positive secrets $(j < c)$ jointly match less than $n-(c-j)(n-d)$ positions of $\mathbf{z}$, then any secret that agrees with $\mathbf{z}$ in at least $(c-j)(n-d)+1$ of the unmatched positions is also a positive secret.*

### 4.1 Identifying Secrets in the Martirosyan-Van Trung Code

We now present our tracing algorithm for the Martirosyan-van Trung $c$-TA code.

If we recall the code construction from Theorem 2, we started with codes:

$C_0 : (n_0, M_0, s_0)$ $c$-IPP ($c$-TA) code with $M_0 = s_0^{\lceil \frac{n_0}{c^2} \rceil}$, and

$C_1^* : (n_0^*, M_1, M_0)$ $c$-IPP ($c$-TA) code with $n_0^* = n_0^{\lceil \frac{n_0}{c^2} \rceil}$ and $M_1 = M_0^{\lceil \frac{n_0^2}{c^2} \rceil}$.

Denoting code concatenation with the symbol $||$, we have the following sequence of codes: $C_1 = C_0||C_1^*$ ; $C_2 = C_1||C_2^*$ ; ... ; $C_h = C_{h-1}||C_h^*$. Where the $C_j$ are the inner codes, the $C_i^*$ are the outter codes, and each $C_i^*$ is an $(n_{i-1}^*, M_i, M_{i-1})$ $c$-TA code. Now, the mapping $\phi_i : M_{i-1} \to C_{i-1}$ maps the symbols of the alphabet $M_{i-1}$ (that is the alphabet of the code $C_i^*$) to the codewords of the code $C_{i-1}$.

Due to the recursive nature of the code, the decoding will be done in two stages. In the first stage we will need to decode the $(n_0, M_0, s_0)$ code $C_0$, this will be accomplished by an algorithm that we call **Reply_To_SecretList**.

Since at the start there is no side information at all, we construct from the reply $\mathbf{z} = (z_1, \ldots, z_n)$, a reliability matrix $\mathcal{R}$, in which we suppose that all symbols in the reply are "correct". This is done as follows: since the $i$th position in the reply corresponds to the $i$th column of $\mathcal{R}$, then in column $i$, we set the row corresponding to the symbol in $z_i$ to 1, and all the other rows in this column to zero. Note that this is consistent with the definition of $r_{i,j}$ in Section 2.2. We apply this matrix to the KV algorithm. Note that at the output of the KV algorithm we can identify at least one positive secret.

Once some positive secrets are identified, the algorithm computes the number of remaining secrets to be found. Also all symbol positions where these already identified secrets match the reply are erased. This is done as follows: suppose that a positive secret and $\mathbf{z}$ agree in position $j$. Again, since position $j$ in $\mathbf{z}$ corresponds to the $j$th column of $\mathcal{R}$, then in column $j$, we set all entries to $1/s_0$. Note that this is consistent with the definition of an erasure. We construct a new reliability matrix and make another run of the KV algorithm to see if any other positive secrets can be identified. This step is repeated until it becomes clear that there are no more positive secrets.

**Reply_To_SecretList**($C$,$\mathbf{z}$):
*Input:* $C$: Reed-Solomon $(n, M, q)$ $c$-TA code; Reply $\mathbf{z}$.
*Output:* A list $L$ of all positive secrets whose reply is $\mathbf{z}$.
// *Local variables* $\Rightarrow$ $i$: iteration step; $c_i$: remaining secrets to be found;
$\mathcal{E}_i$: erased positions in the reply.

1. Set $i := 1$, $c_i := c$ and $\mathcal{E}_i := \{\emptyset\}$.
2. $j := 0$.
3. Using the reply $\mathbf{z}$, compute the entries of the $q \times n$ matrix $\mathcal{R}$ as follows:

$$r_{s,t} := \begin{cases} 1/q \text{ if } t \in \mathcal{E}_i \\ 1 \quad \text{if } z_t = \gamma_s \text{ and } t \notin \mathcal{E}_i \\ 0 \quad \text{otherwise} \end{cases} \tag{5}$$

4. Apply $\mathcal{R}$ to the KV soft-decision algorithm. From the output list take all codewords $\mathbf{u}_{i_1}, \ldots, \mathbf{u}_{i_{j_w}}$, that agree with $\mathbf{z}$ in at least $(c_i(k-1)+1)$ of the positions not in $\mathcal{E}_i$, and add them to $L_l$.
   Set $j := j + j_w$.
5. If $j_w \neq 0$ then
   (a) $\mathcal{E}_i := \{m \ : \ (z_m = u_m) \ \forall \ \mathbf{u} \in L\}$.
   (b) Go to Step 3
6. Set $i := i + 1$, $c_i := c_{i-1} - j$ and
   $\mathcal{E}_i = \{m \ : \ (z_m = u_m) \ \forall \ \mathbf{u} \in L\}$.
7. If $j = 0$ or $c_i = 0$ or if $|\mathcal{E}_i| \geq (n - c_i(k-1))$ output $L$ and quit, else go to Step 2.

In the second stage of the tracing algorithm we will decode the code $C_h$ by first decoding codes $C_1^*, \ldots, C_{h-1}^*$. To decode code $C_i^*$, we will use the function **SymbolList_To_SecretList**. This function has the particularity that instead of accepting a codeword at its input, it accepts a set of lists of alphabet symbols. These lists can be processed by using soft-decision decoding techniques (each list corresponding to a column of $\mathcal{R}$), and this is were our advantage comes from, being able to deal with more that one symbol for each position allows us to pass the maximum amount of information from the inner code to the outter code.

In the following function we consider the ordering $\{\gamma_1, \gamma_2, \ldots, \gamma_{|M_{i-1}|}\}$ of the elements of the alphabet $M_{i-1}$.

**SymbolList_To_SecretList**($C_i^*$,$S\_list_1, \ldots, S\_list_{n^*_{i-1}}$):
*Input:*

- $C_i^*$: Reed-Solomon $(n^*_{i-1}, M_i, M_{i-1})$ $c$-traceability code;
- $n^*_{i-1}$ lists, $S\_list_l = \{m_1^l, \ldots, m^l_{|S\_list_l|}\}$ $1 \leq l \leq n^*_{i-1}$ where $m_i^l \in M_{i-1}$.

*Output:* A list $O\_list$ of codewords of $C_i^*$.

1. Set $j := 0$, $i := 0$ and $c_i := c$.
2. Using the lists $S\_list_1, \ldots, S\_list_{n^*_{h-1}}$ construct the $(M_{i-1} \times n^*_{h-1})$ reliability matrix $\mathcal{R}$ as follows:
$$r_{s,l} := \begin{cases} \frac{1}{|S\_list_l|} & \text{if } \exists\ m_t^l\ =\ \gamma_s \\ 1/M_{i-1} & \text{if } |S\_list_l| = 0 \\ 0 & \text{otherwise} \end{cases} \quad (6)$$
3. Apply $\mathcal{R}$ to the KV soft-decision algorithm. From the output list take all codewords $\mathbf{u}^j$, such that $u_l^j \in S\_list_l$ for at least $(c_i(k-1)+1)$ values of $l$, and add them to $O\_list$.
4. Set $i := i + 1$, $c_i := c_{i-1} - j$ and
$S\_list_l := S\_list_l - \{m_i^l : (m_i^l = u_l)$ for some $\mathbf{u} \in O\_list\}$.
5. If $j = 0$ or $c_i = 0$ output $O\_list$ and quit, else go to step 2.

The overall tracing algorithm uses the algorithms **Reply_To_SecretList** and **SymbolList_To_SecretList** to identify the positive secrets given a reply $\mathbf{z}$ when the strategy is a Martirosyan-van Trung code $C_h$.

The algorithm first breaks the reply in "reply blocks" of $n_0$ symbols and identifies all positive secrets of the "reply blocks" using the function **Reply_To_SecretList**. In then works its way up to the recursive Martirosyan-van Trung code using the function **SymbolList_To_SecretList**.

**Tracing Algorithm:**
*Input:*

- $c$: positive integer;
- $C_h$: Reed-Solomon $c$-traceability Martirosyan-van Trung code;
- Reply $\mathbf{z}$.

*Output:* A list $Out\_PLlist_1$ of all positive secrets whose reply is $\mathbf{z}$.

1. For $cont = 1$ to $\prod_{k=0}^{h-1} n_k^*$.
   - Take symbols $\mathbf{z}_{cont} = (z_{(cont-1)n_0+1}, \ldots, z_{(cont)n_0})$
   - $Out\_PList^{(0)}_{cont}$:=**Reply_To_SecretList**($C_0$,$\mathbf{z}_{cont}$)
2. Set $j := 1$.
3. For $cont' := 1$ to $\prod_{k=j}^{h-1} n_k^*$.
   - With the secret (codeword) lists :
$$Out\_PList^{(j-1)}_{(cont'-1)n^*_{j-1}+1}, \ldots, Out\_PList^{(j-1)}_{(cont')n^*_{j-1}+1}$$
   use the mapping $\phi : M_{j-1} \rightarrow C_{j-1}$
   to obtain the lists of symbols $SL^{(j-1)}_{(cont'-1)n^*_{j-1}+1}, \ldots, SL^{(j-1)}_{(cont')n^*_{j-1}}$,
   where $SL_l = \{h_1, \ldots, h_{|SL_l|}\}$, $h_i \in M_{j-1}$.

– $Out\_PList^{(j)}_{cont'} :=$ **SymbolList_To_SecretList**$(C^*_j, SL^{(j-1)}_{(cont'-1)n^*_{j-1}+1}, \ldots, SL^{(j-1)}_{(cont')n^*_{j-1}})$

4. Set $j := j + 1$.
5. If $j > h$ output $Out\_PList^{(h)}_1$ (there is only one "surviving" list) else go to Step 3.

Note that since for the code concatenation $C_h = C_{h-1}||C^*_h$, the size of codes $C_h$ and $C^*_h$ is the same, we output the secrets as codewords of the code $C^*_h$.

### 4.2 Analysis and Correctness of the Algorithm

For $c$-IPP codes the runtime complexity of the tracing algorithm is in general $O\left(\binom{M}{c}\right)$, whereas for $c$-traceability codes this complexity is in general $O(M)$, where $M$ is the size of the code. This is where the advantage of $c$-traceability codes over $c$-IPP codes comes from. In the Martirosyan-van Trung construction the code $C_0$ and all $C^*_i$ $(1 \leq i \leq n)$ codes are $c$-traceability codes, this implies that there exists a tracing algorithm with running time complexity $O(M)$. We achieve the running time $poly(\log M)$ promised in [9], by using the Koetter-Vardy algorithm that runs in time polynomial.

To prove the correctness of the algorithm we need to show that given a reply based on the strategy of a Martirosyan-van Trung recursive code, there is at least one secret present in the output list $Out\_PList^{(h)}_1$. To do this, it suffices to show that both of the algorithms **Reply_To_SecretList** and **SymbolList_To_SecretList** identify all positive secrets.

We first show that for a $c$-TA $(n, M, q)$ code the algorithm **Reply_To_SecretList** outputs all positive secrets. Suppose that we have a reply $\mathbf{z}$ with $n - m$ positions erased (all values in the corresponding column in the reliability matrix have value $1/q$). In this case, we have that $\langle \mathcal{R}, \mathcal{R} \rangle = m + \frac{n-m}{q}$.

Suppose furthermore, that there are still $c - j$ unidentified secrets. From Corollary 2 we have that a positive secret $\mathbf{u}$, agrees with $\mathbf{z}$ in at least $(c-j)(n-d)+1$ of the non-erased positions, and therefore $\langle \mathcal{R}, [\mathbf{u}] \rangle = (c-j)(n-d)+1+\frac{n-m}{q}$.

If $m \leq (c-j)[(c-j)(n-d)+1]$, its immediate to see that

$$\frac{\langle \mathcal{R}, [\mathbf{u}] \rangle}{\sqrt{\langle \mathcal{R}, \mathcal{R} \rangle}} \geq \frac{(c-j)(n-d)+1+\frac{n-m}{q}}{\sqrt{c-j}\sqrt{(c-j)(n-d)+1+\frac{n-m}{(c-j)q}}}$$

it follows that (3) is satisfied and the positive secret $\mathbf{u}$ is present in the output list of the KV algorithm.

If $m > (c-j)[(c-j)(n-d)+1]$, then we have that at least one of the positive secrets $\mathbf{v}$ agrees with the reply $\mathbf{z}$ in at least $m/(c-j)$ positions. In this case, $\langle \mathcal{R}, [\mathbf{u}] \rangle = \frac{m}{c-j} + \frac{n-m}{q}$ and $\langle \mathcal{R}, \mathcal{R} \rangle = m + \frac{n-m}{q}$. It follows that

$$\left( \frac{\langle \mathcal{R}, [\mathbf{v}] \rangle}{\sqrt{\langle \mathcal{R}, \mathcal{R} \rangle}} \right)^2 = \frac{\frac{m^2}{(c-j)^2} + \frac{2m(n-m)}{(c-j)q} + \frac{(n-m)^2}{q^2}}{m + \frac{n-m}{q}}$$

and since

$$\frac{\frac{m^2}{(c-j)^2}}{m} = \frac{m}{(c-j)^2} \text{ then } \frac{\frac{m^2}{(c-j)^2} + \frac{2m(n-m)}{(c-j)q} + \frac{(n-m)^2}{q^2}}{m + \frac{n-m}{q}} \geq \frac{m}{(c-j)^2} > n-d$$

where the last inequality follows from the fact that $m > (c-j)[(c-j)(n-d)+1]$. It follows that again (3) is satisfied and the positive secret $\mathbf{v}$ is present in the output list of the KV algorithm. The algorithm will iterate until all positive secrets are identified.

For the **SymbolList_To_SecretList** algorithm, we have that the worst case situation is when in the input lists there is the minimum information required to identify the associated positive secrets. If there are $c-j$ unidentified secrets, this worst case situation is clearly the one in which there are (c-j)(n-d)+1 lists of size $c-j$ and $n-[(c-j)(n-d)+1]$ empty lists. We set the entries of $\mathcal{R}$ according to (6), and therefore $\langle \mathcal{R}, \mathcal{R} \rangle = \frac{(c-j)(n-d)+1}{c-j} + \frac{n-[(c-j)(n-d)+1]}{M_{i-1}}$. Since a positive secret, say $\mathbf{w}$ must contribute in at least $c-j)(n-d)+1$ of the positions, we have that $\langle \mathcal{R}, [\mathbf{w}] \rangle = \frac{(c-j)(n-d)+1}{c-j} + \frac{n-[(c-j)(n-d)+1]}{M_{i-1}}$ After a simple calculation

$$\frac{\langle \mathcal{R}, [\mathbf{w}] \rangle}{\sqrt{\langle \mathcal{R}, \mathcal{R} \rangle}} = \sqrt{(n-d) + \frac{1}{c-j} + \frac{n-[(c-j)(n-d)+1]}{M_{i-1}}}$$

it follows that again (3) is satisfied and therefore **SymbolList_To_SecretList** traces all positive secrets for all codes $C_i^*$.

## 5 Conclusions

The importance of the $q$-ary guessing secrets problem lies in its connection to the traitor tracing problem. The focus of this paper is on the design of efficient identification algorithms for tracing traitors. In particular we discuss an algorithm that using the Koetter-Vardy algorithm as its underlying routine decodes the Martirosyan-van Trung code construction.

## References

1. N. Alon, V. Guruswami, T. Kaufman, and M. Sudan. Guessing secrets efficiently via list decoding. *ACM, SODA 02*, pages 254–262, 2002.
2. B. Chor, A. Fiat, and M. Naor. Tracing traitors. *Advances in Cryptology-Crypto'94, LNCS*, 839:480–491, 1994.
3. F. Chung, R. Graham, and T. Leighton. Guessing secrets. *The Electronic Journal of Combinatorics*, 8(1), 2001.
4. I've got a secret. *A classic '50's and '60's television gameshow. See http://www.timvp.com/ivegotse.html.*
5. H. D. L. Hollmann, J. H. van Lint, J. P. Linnartz, and L. M. G. M. Tolhuizen. On codes with the Identifiable Parent Property. *J. Comb. Theory*, 82(2):121–133, May 1998.

6. R. Koetter and A. Vardy. Algebraic soft-decision decoding of Reed-Solomon codes. *ISIT'00*, 2000.
7. R. Safavi-Naini and Y. Wang. Sequential traitor tracing. *IEEE Trans. Inform. Theory*, 49(5):1319–1326, May 2003.
8. J. N. Staddon, D. R. Stinson, and R. Wei. Combinatorial properties of frameproof and traceability codes. *IEEE Trans. Inform. Theory*, 47(3):1042–1049, 2001.
9. T. V. Trung and S. Martirosyan. New constructions for ipp codes. *Proc. IEEE International Symposium on Information Theory, ISIT '03*, page 255, 2003.

# Probabilistic Analyses on Finding Optimal Combinations of Primality Tests in Real Applications*

Heejin Park[1], Sang Kil Park[2], Ki-Ryong Kwon[3], and Dong Kyue Kim[2]

[1] College of Information and Communications,
Hanyang University, Seoul 133-791, South Korea
[2] School of Electrical and Computer Engineering,
Pusan National University, Busan 609-735, South Korea
[3] Division of Digital & Information Engineering,
Pusan University of Foreign Studies, Busan 608-738, South Korea

**Abstract.** Generating a prime is an iterative application of generating a random number $r$ and testing the primality of $r$ until $r$ is a prime. Among them, the primality test on $r$ is much more time-consuming than the random number generation and thus it occupies most of the running time of the prime generation. To reduce the running time of the primality test, real applications combine several primality test methods. The most widely used combination is the combination of the trial division and the probabilistic primality test. Although this combination is widely used in practice, few analyses were given on finding the optimal combination, i.e., on finding the optimal number of small primes used in trial division that minimizes the expected running time of this combination.

In this paper, we present probabilistic analyses on finding the optimal combinations of the trial division and the probabilistic primality test. Using these analyses, we present three optimal combinations. One is for the primality test and the others are for the safe primality test. The optimal combinations are universal in that they are presented as functions of `div` and `ppt` where `div` is the time required for dividing the random number $r$ by a small prime and `ppt` is the time required for the probabilistic primality test of $r$. Thus, in any situation that `div` and `ppt` can be measured, the optimal combinations can be calculated from these functions. The experimental results show that our probabilistic analyses predict the optimal combinations well. The predicted optimal combinations can be used as useful guidelines in designing a primality or a safe primality test. The usefulness of the optimal combinations is more evident when the primality test is implemented on embedded systems or crypto-processors because finding optimal combinations using experiments is very time-consuming and inefficient.

* This research was supported by the Program for the Training of Graduate Students in Regional Innovation which was conducted by the Ministry of Commerce, Industry and Energy of the Korean Government, and supported by KOSEF grant R01-2002-000-00589-0. Contact Author: dkkim@islab.ce.pusan.ac.kr.

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 74–84, 2005.

## 1 Introduction

A prime $p$ is an integer that it is divisible only by 1 and $p$. A safe prime $p$ is a prime such that $(p-1)/2$ is also a prime. For example, 7 is a safe prime because 7 is prime and $(7-1)/2 = 3$ is also a prime. However, 13 is not a safe prime because $(13-1)/2 = 6$ is not a prime.

Generating large primes is important in cryptography. The securities of cryptosystems and signature schemes largely depend on the size of primes. The larger the prime is, the more secure the cryptosystems and signature schemes are. Thus, cryptographic algorithms such as RSA [12] cryptosystem and ElGamal [4] cryptosystem and signature schemes such as DSS [3] signature scheme require generating large (512-bit or 1024-bit) primes. In addition, recent research showed that safe primes enhance the security of cryptosystems and signature schemes [8]. For example, DH key agreement protocol can be protected from the subgroup attacks if safe primes are used.

The prime generation is an iterative application of the following two steps until $r$ is a prime.

1. *Random number generation*: Generate a random number $r$.
2. *Primality test on* $r$: Test if $r$ is a prime. If it is not, goto step 1.

Similarly, the safe prime generation is an iterative application of the following three steps until $r$ is a safe prime.

1. *Random number generation*: Generate a random number $r$.
2. *Primality test on* $r$: Test if $r$ is a prime. If it is not, goto step 1.
3. *Primality test on* $(r-1)/2$: Test if $(r-1)/2$ is a prime. If it is not, goto step 1.

Since the random number generation is much faster than the primality test, we focus on improving the primality tests. The primality tests are divided into two categories, which are the *deterministic* primality test and the *probabilistic* primality test. The deterministic primality test certifies a random number is a prime with probability 1. That is, if a random number passes the primality test, it is surely a prime. The *trial division* [2] which divides an $n$-bit random number by primes up to $\sqrt{n}$ is a kind of the deterministic primality test. However, this test is too slow to use in practice. There are other deterministic primality tests such as *Pocklington's test* [10] and *Jacobi sum test* [1], but they are also too slow. The probabilistic primality test certifies a random number is a prime with a probability very close to 1, such as $1 - 1/2^s$ for large $s$. The probabilistic primality test includes the *Fermat test* [2], the *Solovay–Strassen test* [13], and the *Miller–Rabin test* [7, 11].

In real applications, some of the tests are combined to enhance the speed of the primality test. Some popular combinations are the combinations of the trial division and the probabilistic primality test such as the Fermat test and the Miller–Rabin test. It is employed by OpenSSL [14] in the following way.

1. The trial division using small primes.
2. The Fermat test on the numbers that passed the trial division.

The rationale of combining the trial division with the probabilistic primality test is that the trial division is quite efficient to show that a random number is a composite divisible by small primes, while the probabilistic primality test is not. For example, consider showing that 1056789 (=3*352263) is a composite. The trial division requires only a division by 3 but the Fermat test and the Miller-Rabin test require at least a modular exponentiation which is much expensive than the division. Considering that the primality test is used to certify generated random numbers are composites in most cases, using both the trial division and the probabilistic primality test surely enhances the speed of the primality test.

However, few efforts were made to find the optimal combination of the trial division and the probabilistic primality test, and thus there is no known results on the proper number of small primes used in the trial division. Thus, the only way to compute the optimal combination has been to perform experiments on various combinations and then choose the optimal combination. However, the optimal combination obtained in this way is not a universal one because the optimal combination depends on the time required for a division and the time required for the probabilistic primality test. Moreover, computing the optimal combination in this way is not applicable when the primality test is implemented on embedded systems or crypto-processors because finding optimal combinations using experiments is very time-consuming and inefficient. Hence, finding the universal optimal combination using the probabilistic analysis is required.

In this paper, we present probabilistic analyses on finding the optimal combinations of the trial division and the probabilistic primality test. Using these analyses, we present three optimal combinations. One is for the primality test and the others are for the safe primality test. The optimal combinations are universal in that they are presented as functions of `div` and `ppt` where `div` is the time required for dividing the random number $r$ by a small prime and `ppt` is the time required for the probabilistic primality test of $r$. Thus, in any situation that `div` and `ppt` can be measured, the optimal combinations can be calculated from these functions. To compute an optimal combination, we first perform a probabilistic analysis of the expected running time of the combination of the trial division and the probabilistic primality test, and then we compute the optimal number of primes used in the trial division that minimizes the expected running time of the combination.

This paper is organized as follows. In Section 2, we describe the outline of finding the optimal combinations of the trial division and the probabilistic primality test. In Section 3, we compute the optimal combination for the primality test. In Sections 4 and 5, we compute the optimal combinations for two different safe primality tests. In Section 6, we present some experimental results that show our probabilistic analyses predict the optimal combinations well. In Section 7, we conclude with some remarks.

## 2 The Outline of Finding Optimal Combinations

We describe the outline of computing the optimal combinations of the trial division and the probabilistic primality test, i.e., finding the optimal number of small primes used in trial division that minimizes the expected running time of this combination.

1. We compute the expected running time $T(k)$ of the combination when we use the $k$ smallest odd primes for the trial division.
2. We show that $\Delta T(k) = T(k) - T(k-1)$ is monotone increasing and thus $T(k)$ is a convex function.
3. We compute the optimal number $k$ minimizing $T(k)$ by computing $k$ satisfying $\Delta T(k) = 0$.
4. Let `div` is the time required for dividing a random number by a small prime and `ppt` is the time required for the probabilistic primality test of the random number. Since the optimal number $k$ is represented as a function of `div` and `ppt`, we first measure `div` and `ppt` and then calculate the optimal number of small primes used in trial division.

We describe how to compute the optimal combination for each case step by step. Since step 4 is trivial, we focus on describing steps 1-3.

## 3 The Case of Prime Numbers

As we described in the previous section, we first compute the expected running time $P_n(k)$ of the primality test on an $n$-bit integer when the $k$ smallest odd primes are used in the trial division, then we show that $\Delta P_n(k)$ is monotone increasing, and finally we compute the optimal number $k$ satisfying $\Delta P_n(k) = 0$.

We denote by $X_i$ the event of dividing an $n$-bit integer by $p_i$ for $1 \leq i \leq k$ in the trial division and by $Y$ the event of performing the probabilistic primality test on an $n$-bit integer. Let $\Pr\{X_i\}$ and $\Pr\{Y\}$ denote the probabilities of $X_i$ and $Y$, respectively. We denote by $\mathtt{div}_n(i)$ the time required to divide an $n$-bit integer by $p_i$ and by $\mathtt{ppt}_n$ the time required to perform the probabilistic primality test on an $n$-bit integer. Then, the expected running time for the primality test on an $n$-bit integer is

$$\begin{aligned} P_n(k) &= \Pr\{X_1\} \cdot \mathtt{div}_n(1) + \cdots + \Pr\{X_k\} \cdot \mathtt{div}_n(k) + \Pr\{Y\} \cdot \mathtt{ppt}_n \\ &= \sum_{i=1}^{k} (\Pr\{X_i\} \cdot \mathtt{div}_n(i)) + \Pr\{Y\} \cdot \mathtt{ppt}_n \end{aligned} \tag{1}$$

We show how to compute $\Pr\{X_i\}$ and $\Pr\{Y\}$. We first compute $\Pr\{X_i\}$, i.e., the probability that we divide an $n$-bit integer $r$ by $p_j$ in the trial division. Since we divide $r$ by $p_i$ if and only if $r$ is not divisible by any primes up to $p_{i-1}$, the probability $\Pr\{X_i\}$ that we divide $r$ by $p_i$ is

$$\Pr\{X_i\} = (1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \cdots (1 - \frac{1}{p_{i-1}}). \tag{2}$$

Consider the probability $\Pr\{Y\}$ that we perform the probabilistic primality test on $r$. Since we perform the probabilistic primality test if and only if $r$ is not divisible by any $k$ smallest primes, $\Pr\{Y\}$ is

$$\Pr\{Y\} = (1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \cdots (1 - \frac{1}{p_k}). \tag{3}$$

We will use $\Phi(i)$ to denote $(1 - 1/p_1)(1 - 1/p_2) \cdots (1 - 1/p_i)$. Then, $\Pr\{X_i\}$ and $\Pr\{Y\}$ become

$$\Pr\{X_i\} = \Phi(i-1), \quad \Pr\{Y\} = \Phi(k). \tag{4}$$

Hence, we get the following equation by replacing $\Pr\{X_i\}$ by $\Phi(i-1)$ and $\Pr\{Y\}$ by $\Phi(k)$ in equation (1).

$$P_n(k) = \sum_{i=1}^{k} (\Phi(i-1) \cdot \mathtt{div}_n(i)) + \Phi(k) \cdot \mathtt{ppt}_n \tag{5}$$

We show that $\Delta P_n(k) = P_n(k) - P_n(k-1)$ is monotone increasing and thus $P_n(k)$ is a convex function. The computation of $\Delta P_n(k)$ is as follows. Since $P_n(k)$ is $\sum_{j=1}^{k} (\Phi(j-1) \cdot \mathtt{div}_n(j)) + \Phi(k) \cdot \mathtt{ppt}_n$ by equation 5,

$$\begin{aligned} P_n(k) - P_n(k-1) &= (\sum_{j=1}^{k} (\Phi(j-1) \cdot \mathtt{div}_n(j)) + \Phi(k) \cdot \mathtt{ppt}_n) \\ &\quad - (\sum_{j=1}^{k-1} (\Phi(j-1) \cdot \mathtt{div}_n(j)) + \Phi(k-1) \cdot \mathtt{ppt}_n) \\ &= \Phi(k-1) \cdot \mathtt{div}_n(k) + (\Phi(k) - \Phi(k-1)) \cdot \mathtt{ppt}_n \\ &= \Phi(k-1) \cdot \mathtt{div}_n(k) - \Phi(k-1) \cdot \frac{1}{p_k} \cdot \mathtt{ppt}_n \\ &= \Phi(k-1) \cdot (\mathtt{div}_n(k) - \frac{\mathtt{ppt}_n}{p_k}) \end{aligned} \tag{6}$$

Thus, $\Delta P_n(k)$ is negative if $\mathtt{div}_n(k) < \mathtt{ppt}_n/p_k$, zero if $\mathtt{div}_n(k) = \mathtt{ppt}_n/p_k$, and positive if $\mathtt{div}_n(k) > \mathtt{ppt}_n/p_k$. Since $\mathtt{div}_n(k) - \mathtt{ppt}_n/p_k$ is negative when $k$ is small and it becomes positive as $k$ grows larger, $\Delta P_n(k)$ is monotone increasing and thus $P_n(k)$ is a convex function.

We compute the optimal number $k$ minimizing $P_n(k)$. Since $\Delta P_n(k) = 0$ when $\mathtt{div}_n(k) = \mathtt{ppt}_n/p_k$, $P_n(k)$ has the smallest value when $\mathtt{div}_n(k) = \mathtt{ppt}_n/p_k$. i.e., $p_k = \mathtt{ppt}_n/\mathtt{div}_n(k)$. Hence, we get the following theorem.

**Theorem 1.** *$P_n(k)$ has the smallest value when $p_k = \mathtt{ppt}_n/\mathtt{div}_n(k)$.*

## 4 The Case of Safe Prime Numbers in OpenSSL

We first present the safe primality test used in OpenSSL. Let $p_1 < p_2 < \cdots < p_k$ denote the $k$ smallest odd primes used in the trial division.

1. The trial division on $r$ and $(r-1)/2$: For each prime $p_i$, $1 \leq i \leq k$, we check if $r \bmod p_i \neq 0$ and $(r-1)/2 \bmod p_i \neq 0$.
2. The probabilistic primality test on $r$.
3. The probabilistic primality test on $(r-1)/2$.

We first compute the expected running time $S_n(k)$ of the above safe primality test. We reuse the notations $X_i$, $Y$, $\mathtt{div}$, and $\mathtt{ppt}$ defined in the previous section. In addition, we denote by $X_i'$ the event of dividing an $(n-1)$-bit integer $(r-1)/2$ by $p_i$ for $1 \leq i \leq k$ in the trial division and by $Y'$ the event of performing the probabilistic primality test on $(r-1)/2$. Then, the expected running time for the safe primality test on an $n$-bit integer is

$$S_n(k) = \sum_{i=1}^{k}(\Pr\{X_i\} \cdot \mathtt{div}_n(i)) + \sum_{i=1}^{k}(\Pr\{X_i'\} \cdot \mathtt{div}_{n-1}(i)) + \Pr\{Y\} \cdot \mathtt{ppt}_n + \Pr\{Y'\} \cdot \mathtt{ppt}_{n-1} \tag{7}$$

We show how to compute $\Pr\{X_i\}$, $\Pr\{X_i'\}$, $\Pr\{Y\}$, and $\Pr\{Y'\}$. We first consider the probability $\Pr\{X_i\}$ that we divide an $n$-bit integer $r$ by $p_i$, $1 \leq i \leq k$, in the trial division. We divide $r$ by $p_i$ if and only if none of $r$ and $(r-1)/2$ is divisible by any primes up to $p_{i-1}$. The probability that $r$ and $(r-1)/2$ is not divisible by a prime $p_i$ is $1 - 2/p_i$ because $r$ and $(r-1)/2$ is not divisible by a prime $p_i$ if and only if $r \bmod \neq 0, 1$ by Theorem 1. Hence, the probability $\Pr\{X_i\}$ is

$$\Pr\{X_i\} = (1 - \frac{2}{p_1})(1 - \frac{2}{p_2}) \cdots (1 - \frac{2}{p_{i-1}}). \tag{8}$$

Consider the probability $\Pr\{X_i'\}$ that we divide an $(n-1)$-bit integer $(r-1)/2$ by $p_i$ in the trial division. We divide $(r-1)/2$ by $p_i$ if and only if none of $r$ and $(r-1)/2$ is divisible any primes up to $p_{i-1}$ and $r$ is not divisible by $p_i$. Thus, the probability $\Pr\{X_i'\}$ is

$$\Pr\{X_i'\} = \Pr\{X_i\} \cdot (1 - \frac{1}{p_i}). \tag{9}$$

Since the probability $\Pr\{Y\}$ that we perform the probabilistic primality test on $r$ is the probability that none of $r$ and $(r-1)/2$ is divisible by any $k$ smallest primes, the probability $\Pr\{Y\}$ is

$$\Pr\{Y\} = (1 - \frac{2}{p_1})(1 - \frac{2}{p_2}) \cdots (1 - \frac{2}{p_k}). \tag{10}$$

Consider the probability $\Pr\{Y'\}$ that we perform the probabilistic primality test on $(r-1)/2$. Let $q_n$ denote the probability that an odd $n$-bit integer $r$ is a prime. According to the prime distribution [2],

$$q_n \approx \frac{1}{2^{n-2}}(\frac{2^n}{n \ln 2} - \frac{2^{n-1}}{(n-1)\ln 2}) \approx \frac{2}{n \ln 2}. \tag{11}$$

Since we perform the probabilistic primality test on $(r-1)/2$ if and only if $r$ is a prime and $(r-1)/2$ is not divisible by any primes up to $p_k$, the probability $\Pr\{Y'\}$ is

$$\Pr\{Y'\} = q_n \cdot (1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \cdots (1 - \frac{1}{p_k}). \tag{12}$$

We use $\Phi(i)$ to denote $(1-1/p_1)(1-1/p_2)\cdots(1-1/p_i)$ as we did and use $\Psi(i)$ to denote $(1-2/p_1)(1-2/p_2)\cdots(1-2/p_i)$. Then, the probabilities $\Pr\{X_i\}$, $\Pr\{X'_i\}$, $\Pr\{Y\}$, and $\Pr\{Y'\}$ become

$$\begin{aligned} &\Pr\{X_i\} = \Psi(i-1), \\ &\Pr\{X'_i\} = \Psi(i-1) \cdot (1-1/p_i), \\ &\Pr\{Y\} \ = \Psi(k), \\ &\Pr\{Y'\} = q_n \cdot \Phi(k). \end{aligned} \tag{13}$$

We get the following equation by replacing the probabilities in equation 7 with the values in the equation above.

$$\begin{aligned} S_n(k) = &\sum_{j=1}^{k} (\Psi(j-1) \cdot \mathtt{div}_n(j)) + \sum_{j=1}^{k} (\Psi(j-1) \cdot (1 - \frac{1}{p_j}) \cdot \mathtt{div}_{n-1}(j)) \\ &+ \Psi(k) \cdot \mathtt{ppt}_n + q_n \cdot \Phi(k) \cdot \mathtt{ppt}_{n-1} \end{aligned} \tag{14}$$

We compute the optimal value $k$ minimizing $S_n(k)$. We assume that $\mathtt{div}_{n-1}(k) \approx \mathtt{div}_n(k)$ and $\mathtt{ppt}_{n-1} \approx \mathtt{ppt}_n$ because $n$ is quite large (512, 1024 or 2048). Then, $S_n(k)$ in equation (14) becomes

$$S_n(k) = \sum_{j=1}^{k} \Psi(j-1)(2 - \frac{1}{p_j})\mathtt{div}_n(j) + (\Psi(k) + q_n\Phi(k))\mathtt{ppt}_n. \tag{15}$$

We compute $p_k$ satisfying $S_n(k) - S_n(k-1) = 0$.

$$\begin{aligned} &S_n(k) - S_n(k-1) \\ &= \sum_{j=1}^{k} \Psi(j-1)(2 - \frac{1}{p_j})\mathtt{div}_n(j) + (\Psi(k) + q_n\Phi(k))\mathtt{ppt}_n \\ &\quad - \ (\sum_{j=1}^{k-1} \Psi(j-1)(2 - \frac{1}{p_j})\mathtt{div}_n(j) + (\Psi(k-1) + q_n\Phi(k-1))\mathtt{ppt}_n) \\ &= \Psi(k-1)(2 - \frac{1}{p_k})\mathtt{div}_n(k) - (\Psi(k-1)\frac{2}{p_k} + q_n\Phi(k-1)\frac{1}{p_k})\mathtt{ppt}_n \\ &= \Psi(k-1)((2 - \frac{1}{p_k})\mathtt{div}_n(k) - (\frac{2}{p_k} + \ q_n \cdot \frac{\Phi(k-1)}{\Psi(k-1)} \cdot \frac{1}{p_k})\mathtt{ppt}_n) \\ &= \Psi(k-1)((2 - \frac{1}{p_k})\mathtt{div}_n(k) - \frac{1}{p_k}(2 + \ q_n \cdot \prod_{i=1}^{k-1} \frac{p_i - 1}{p_i - 2})\mathtt{ppt}_n) \end{aligned} \tag{16}$$

Thus, $S_n(k)$ has the smallest value when $p_k$ satisfies

$$\frac{1}{2p_k - 1} \cdot (2 + q_n \cdot \prod_{i=1}^{k-1} \frac{p_i - 1}{p_i - 2}) = \frac{\mathtt{div}_n(k)}{\mathtt{ppt}_n} \tag{17}$$

Since $q_n \approx 2/(n \ln 2)$ by equation (11), we get the following theorem.

**Theorem 2.** *$S_n(k)$ has the smallest value when $p_k$ satisfies*

$$\frac{1}{2p_k - 1} \cdot (2 + \frac{2}{n \ln 2} \cdot \prod_{i=1}^{k-1} \frac{p_i - 1}{p_i - 2}) = \frac{\mathtt{div}_n(k)}{\mathtt{ppt}_n}.$$

## 5 Another Case of Safe Prime Numbers

We first introduce a safe primality test [9] that is more efficient than the safe primality test in OpenSSL. It seems that this safe primality test had been used by some programs without being published.

1. The trial division on $r$ and $(r-1)/2$: For each prime $p_i$, $1 \leq i \leq k$, we only check if $r \bmod p_i \neq 0, 1$.
2. The probabilistic primality test on $r$.
3. The probabilistic primality test on $(r-1)/2$.

The only difference of the above algorithm differs from the algorithm in the previous section is '$(r-1)/2 \bmod p_i \neq 0$' is replaced by '$r \bmod p_i \neq 1$'. We show the above algorithm is correct by proving that '$(r-1)/2 \bmod p \neq 0$' if and only if '$r \bmod p \neq 1$', i.e., '$(r-1)/2 \bmod p = 0$' if and only if '$r \bmod p = 1$' for any odd prime $p$.

**Lemma 1.** *$(r-1)/2 \bmod p = 0$ if and only if $r \bmod p = 1$ for any odd prime $p$.*

*Proof.* We first prove the forward direction. If $(r-1)/2 \bmod p = 0$, $(r-1)/2$ is divisible by $p$. Since $(r-1)/2$ is divisible by $p$ and $r-1$ is a multiple of $(r-1)/2$, $r-1$ is also divisible by $p$. Thus, $r \bmod p = 1$. We prove the backward direction. If $r \bmod p = 1$, $r-1$ is divisible by $p$. Since $\gcd(2, p) = 1$, $(r-1)/2$ is divisible by $p$, which implies $(r-1)/2 \bmod p = 0$.

We compute the expected running time $N_n(k)$ for the above safe primality test on an $n$-bit integer $r$. The expected running time $N_n(k)$ is the same as the expected running time $S_n(k)$ in the previous section. except that we do not divide $(r-1)/2$ by small primes. Thus, it becomes

$$N_n(k) = \sum_{j=1}^{k} (\Psi(j-1) \cdot \mathtt{div}_n(j)) + \Psi(k) \cdot \mathtt{ppt}_n + q_n \cdot \Phi(k) \cdot \mathtt{ppt}_{n-1}. \quad (18)$$

We compute the optimal value $k$ minimizing $N_n(k)$. Since we assume that $\mathtt{ppt}_n = \mathtt{ppt}_{n-1}$, $N_n(k)$ in equation (18) becomes

$$N_n(k) = \sum_{j=1}^{k} \Psi(j-1)\mathtt{div}_n(j) + (\Psi(k) + q_n\Phi(k))\mathtt{ppt}_n \quad (19)$$

We compute $p_k$ satisfying $N_n(k) - N_n(k-1) = 0$.

$$\begin{aligned}
&N_n(k) - N_n(k-1) \\
&= \sum_{j=1}^{k} \Psi(j-1)\mathtt{div}_n(j) + (\Psi(k) + q_n\Phi(k))\mathtt{ppt}_n \\
&\quad - (\sum_{j=1}^{k-1} \Psi(j-1)\mathtt{div}_n(j) + (\Psi(k-1) + q_n\Phi(k-1))\mathtt{ppt}_n) \\
&= \Psi(k-1)\mathtt{div}_n(k) - (\Psi(k-1)\frac{2}{p_k} + q_n\Phi(k-1)\frac{1}{p_k})\mathtt{ppt}_n \qquad (20) \\
&= \Psi(k-1)(\mathtt{div}_n(k) - (\frac{2}{p_k} + q_n \cdot \frac{\Phi(k-1)}{\Psi(k-1)} \cdot \frac{1}{p_k})\mathtt{ppt}_n) \\
&= \Psi(k-1)(\mathtt{div}_n(k) - \frac{1}{p_k}(2 + q_n \cdot \prod_{i=1}^{k-1} \frac{p_i - 1}{p_i - 2})\mathtt{ppt}_n)
\end{aligned}$$

Thus, $N_n(k)$ has the smallest value when $p_k$ satisfies

$$\frac{1}{p_k} \cdot (2 + q_n \cdot \prod_{i=1}^{k-1} \frac{p_i - 1}{p_i - 2}) = \frac{\mathtt{div}_n(k)}{\mathtt{ppt}_n} \qquad (21)$$

Since $q_n \approx 2/(n \ln 2)$, we get the following theorem.

**Theorem 3.** *$N_n(k)$ has the smallest value when $p_k$ satisfies*

$$\frac{1}{p_k} \cdot (2 + \frac{2}{n \ln 2} \cdot \prod_{i=1}^{k-1} \frac{p_i - 1}{p_i - 2}) = \frac{\mathtt{div}_n(k)}{\mathtt{ppt}_n}.$$

## 6 Experimental Results

We performed the probabilistic analyses using the parameters obtained from an actual machine, a PC equipped with Pentium III 1.6 Ghz CPU and 1 GB

| $n$ | $\mathtt{div}_n(i)$ | $\mathtt{ppt}_n$ | $\mathtt{ppt}_{n-1}$ |
|---|---|---|---|
| 512 bits | 0.011 | 8.328 - 8.554 | 8.246 - 8.391 |
| 1024 bits | 0.012 | 53.639 - 54.433 | 52.820 - 53.145 |
| 2048 bits | 0.017 | 368.503 - 378.741 | 368.706 - 369.331 |

**Fig. 1.** The execution time $\mathtt{div}_n(i)$ of a division and the execution times $\mathtt{ppt}_{n-1}$ and $\mathtt{ppt}_n$ of the probabilistic primality tests measured on a Pentium III 1.6 Ghz with 1 GB main memory

| $n$ | Primality test | OpenSSL safe primality test | Our safe primality test |
|---|---|---|---|
| 512 bits | $p_{135} = 769$ | $p_{138} = 797$ | $p_{249} = 1,583$ |
| 1024 bits | $p_{611} = 4,513$ | $p_{619} = 4,583$ | $p_{1,137} = 9,181$ |
| 2048 bits | $p_{2,444} = 21,799$ | $p_{2,465} = 22,013$ | $p_{4,583} = 44,041$ |

**Fig. 2.** The number of primes optimizing the trial division and the probabilistic primality test on a Pentium III 1.6Ghz with 1GB main memory computed by Theorem 1, Theorem 2, and Theorem 3

| | 50 | 100 | 150 | 250 | 500 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|
| OpenSSL | 0.35 | 0.30 | 0.29 | 0.32 | 0.39 | 0.50 | 0.72 |
| Our algorithm | 0.23 | 0.22 | 0.19 | 0.17 | 0.18 | 0.20 | 0.25 |

**Fig. 3.** The time measured in millisecond required of each combination of trial division and the probabilistic primality test to test the safe primality of a random number when $n = 512$

main memory and compare them with the experimental data. Figure 1 shows the parameters, $\mathtt{div}_n(i)$, $1 \leq i \leq k$, $\mathtt{ppt}_n$, and $\mathtt{ppt}_{n-1}$ measured on the PC when $n$ is 512, 1024, and 2048 bits, respectively. Using the values, we can compute $\mathtt{div}_n(i)/\mathtt{ppt}_n$ and the optimal number of primes to be used in trial division using Theorem 1, Theorem 2, and Theorem 3 (Fig. 2). This figure shows the number of primes that makes the trial division and the probabilistic primality test optimal. When $n = 512$, the OpenSSL safe primality test shows best performance when $k = 138$ and our safe primality test show best performance when $k = 249$.

The current version of OpenSSL uses 2048 smallest primes for the trial division and it takes about 0.7 ms for testing a safe primality of a random number. Reducing the number of primes to 150, it takes 0.29 ms, which is more than 2 times faster than the original version.

## 7 Conclusion

We presented probabilistic analyses on finding the optimal combinations of the trial division and the probabilistic primality test. Using these analyses, we presented three optimal combinations. One is for the primality test and the others are for the safe primality test. The optimal combinations are universal in that they are presented as functions of the execution time of a division ($\mathtt{div}_n$) and the execution time of the probabilistic primality test ($\mathtt{ppt}_n$). The optimal combinations can be used as useful guidelines in designing a primality or a safe primality test. The usefulness of the optimal combinations is more evident when the primality test is implemented on embedded systems or crypto-processors because finding optimal combinations using experiments is very time-consuming and inefficient.

## References


1. W. Bosma and M. P. van der Hulst, Faster primality testing, *CRYPTO'89*, LNCS **435** 652-656 (1990).
2. T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, Introduction to Algorithms, 2nd ed, MIT press (1991)
3. National Institute for Standards and Technology, Digital Signature Standard(DSS), *Federal Register 56* 169 (1991)
4. T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory* **31**(4) 469-472 (1985).
5. B. C. Higgins, The Rabin- Miller Probabilistic Primality Test: Some Results on the Number of Non-Witnesses to Compositeness, citeseer.nj.nec.com/400584.html
6. D.E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms,* Addison-Wesley, U.S.A. (1981).
7. G.L. Miller, Riemann's Hypothesis and Tests for Primality, *Journal of Computer Systems Science* **13**(3) 300-317 (1976).
8. A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, *handbook of Applied Cryptography*, CRC Press, 1997
9. H. Park, An Efficient Implementation of Safe Prime Generation, *International Conference on Ubiquitous Computing*, 241-243, Oct, (2003).
10. H.C. Pocklington, The determination of the prime or composite nature of large numbers by Fermat's theorem, *Proc. of the Cambridge Philosophical Society* **18** 29-30 (1914).
11. M.O. Rabin, Probabilistic Algorithm for Primality Testing, *Journal of Number Theory* **12** 128-138 (1980).
12. R.L. Rivest, A. Shamir and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM* **21**(2) 120-126 (1978).
13. R. Solovay and V. Strassen, A fast Monte-Carlo test for primality, *SIAM Journal on Computing* **6** 84-85 (1977).
14. OpenSSL, http://www.openssl.org.

# Countermeasures for Preventing Comb Method Against SCA Attacks

Mustapha Hedabou, Pierre Pinel, and Lucien Bénéteau

INSA de Toulouse, LESIA,
135, avenue de Rangueil 31077 Toulouse cedex, France
{hedabou, pierre.pinel, lucien.beneteau}@insa-toulouse.fr
http://www.lesia.insa-tlse.fr

**Abstract.** Side Channel Attacks have become a serious threat for cryptographic applications on devices with small resources. Indeed, it turns out that the usual randomization techniques can not prevent the recent DPA attacks (RPA and ZPA). The implementation of elliptic curve cryptosystems (ECC) on such devices must combine an optimized use of space memory with a high level of security and efficiency. In this paper we present an efficient SCA-resistant algorithm based on the fixed-base comb method. We propose to modify the binary representation of the secret scalar in order to obtain a new sequence of non-zero bit-strings. This, combined with the use of Randomized Linearly-transformed coordinates (RLC), will prevent the SCA attacks on the comb method, including RPA and ZPA. Furthermore, our algorithm optimizes the size of the precomputed table; we only store $2^{w-1}$ points instead of $2^w - 1$ for the fixed-base comb method, without affecting in any way the computation time. We also present another countermeasure using a Randomized Initial Point (RIP) to protect the fixed-base comb method against SCA attacks including RPA and ZPA, with an optimized amount of computing time. The cost of this countermeasure does not exceed 2% of the total cost of the fixed-base comb method.

**Keywords:** Elliptic curve, comb method, side channel attacks, scalar multiplication, pre-computed table, memory space.

## 1 Introduction

### 1.1 ECC and Side Channel Attacks

It is well known that elliptic curve cryptosystems (ECC) are suitable for cryptographic applications on devices with small resources, obtaining the same level of security for shorter keys. However, cryptosystems on such devices, including ECC, may be the target of Side Channel Attacks (SCA).

Side Channel Attacks, introduced by Kocher et al [Koc96, KJJ99], exploit some data leaking information such as power consumption and computing time to detect part or whole of the bits of the secret key. We can distinguish two types of SCA attacks: the Simple Power Analysis (SPA) attacks which analyze

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 85–96, 2005.

the information leaking from a single execution of the algorithm, and against which many countermeasures have been proposed [Mon87, LD99, LS01], and the more sophisticated Differential Power Analysis (DPA) which collects information from several executions of the algorithm and interprets it with statistical tools. Randomization techniques applied on the coordinates of the points have been an efficient countermeasure against DPA attacks [Cor99, JT01]; but recently, a new generation of DPA attacks has been proposed: the Refined Power Analysis attack (RPA) [Gou03] and the Zero-value Point Attack (ZPA) [AT03], which exploit the correlation between the power consumption and the data processed by the algorithm for special points that can not be randomized by the usual techniques. An efficient implementation should hence pay a careful attention to these attacks. This paper aims to prevent the fixed-base comb method against all SCA attacks, including RPA and ZPA.

### 1.2 Contribution of This Paper

In this paper we propose a new SCA-resistant scheme based on the fixed-base comb method. The proposed method turns the comb method into a SPA-resistant one by constructing a new sequence of non-zero bit-strings representing the secret scalar; this, combined with Randomized Linearly-transformed Coordinates, will prevent SCA attacks. Furthermore, the proposed scheme is more interesting in terms of used memory space, as we only store $2^{w-1}$ points instead of $2^w - 1$ for the fixed-base method, without increasing the amount of computing time. We also present another efficient countermeasure, based on the use of a Randomized Initial Point (RIP), to prevent the fixed-base comb method against SCA attacks including RPA and ZPA, with negligible computing time with respect to the cost of the original method.

This paper is organized as follows: Section 2 briefly reviews the elliptic curve cryptosystems and describes the fixed-base comb method based on the Lee and Lim technique. In Section 3, we introduce the side channel attacks and the countermeasures against them. In Section 4.1, we present the proposed SCA-resistant scheme based on the fixed-base comb method, and we introduce in Section 4.2 the extended fixed-base comb method with RIP. Finally, we conclude in Section 5.

## 2 Elliptic Curve Cryptosystems

### 2.1 Elliptic Curves

Let $K = \mathbb{F}_q$ be a finite field, where $q > 3$ is a prime integer. The Weierstrass equation of an elliptic curve over $K$ can be written in the following simpler form

$$y^2 = x^3 + ax + b \text{ over } GF(q), \text{ with } a, b \in GF(q) \text{ and } 4a^3 + 27b^2 \neq 0.$$

An elliptic curve over $K$ is the union of the set of the solutions of the Weierstrass equation with a point $\Theta$, called point at infinity. An adding operation can be

defined over the elliptic curve, which turns the set of the points of the curve into a group. The adding operation between two points is defined as follows in affine coordinates. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two points on the elliptic curve, neither being the point at infinity. Over $GF(q)$ the inverse of a point $P_1$ is $-P_1 = (x_1, -y_1)$. If $P_1 \neq -P_2$, then we have $P_1 + P_2 = P_3 = (x_3, y_3)$ with $x_3 = \lambda^2 - x_1 - x_2,\ y_3 = \lambda(x_1 - x_3) - y_1$ and

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \text{ if } P_1 \neq P_2 & \text{(adding)} \\ \frac{3{x_1}^2 + a}{2y_1} \text{ if } P_1 = P_2 & \text{(doubling)} \end{cases}$$

The preceding formulas for adding and doubling of elliptic curve points need a field inversion, which is more expensive than the field multiplication. It may be thus advantageous to represent the points using projective coordinates, thus avoiding the need for an inversion. We give in this section the adding and doubling formulae in Jacobian coordinates [CMO98], but all the following can be applied to other types of projective coordinates.

In Jacobian projective coordinates, the projective point $(X, Y, Z)$, $Z \neq 0$ corresponds to the affine point $(\frac{X}{Z^2}, \frac{Y}{Z^3})$. The projective equation is

$$Y^2 = X^3 + aXZ^4 + bZ^6 \text{ over } GF(p),$$

The adding and doubling formulae in Jacobian projective coordinates can be represented as follows

**Point Adding Formula in Jacobian Coordinates**:
$X_3 = T,\ Y_3 = -8Y_1^4 + M(S - T),\ Z_3 = 2Y_1Z_1,$
$S = 4X_1Y_1^2,\ M = 3X_1^2 + aZ_1^4,\ T = -2S + M^2.$

**Point Doubling Formula in Jacobian Coordinates**:
$X_3 = -H^3 - 2U_1H^2 + R^2,\ Y_3 = -S_1H^3 + R(U_1H^2 - X_3),\ Z_3 = Z_1Z_2H,$
$U_1 = X_1Z_2^2, U_2 = X_2Z_1^2,\ S_1 = Y_1Z_2^3,\ S_2 = Y_2Z_1^3,\ H = U_2 - U_1,\ R = S_2 - S_1.$

These additions formulae require 10 and 16 field multiplications for adding and doubling operations respectively.

The randomized Jacobian coordinates method transforms the base point $(x, y, 1)$ in projective coordinates into $(r^2x, r^3y, r)$ using a random number $r$. The properties of Jacobian projective coordinates imply that the relation $(x, y, 1) = (r^2x, r^3y, r)$ holds for every non-zero $r$.

### 2.2 Elliptic Curves Scalar Multiplication

The computation of the scalar multiplication $kP$, where $P$ is an elliptic curve point and $k$ is an integer, is the most important part of the elliptic curve implementation process. There are many algorithms for computing scalar multiplication; the standard one is the Binary method.

**Algorithm 1: Binary Method**
*Input:* $P,\ k = (k_{l-1}, ...., k_0)_2$.
*Output:* $Q = kP$.
*1.* $Q = P$.
*2. For* $i = l - 2$ *downto 0 do*
*2.1 If* $k_i = 0$ *then* $Q \leftarrow 2Q$
*2.2 else* $Q \leftarrow 2Q + P$.
*3. Return* $Q$.

In the following we describe the fixed-base comb method [BHLM01] based on the Lim and Lee [LL94] technique.

Let $(k_{l-1}, \cdots, k_1, k_0)$ be the binary representation of an integer with $k_i \in \{0, 1\}$, and let $w$ be an integer such as $w \geq 2$; we set $d = \lceil \frac{l}{w} \rceil$. $P$ being an elliptic curve point, for all $(b_{w-1}, \cdots, b_1, b_0) \in {\mathbb{Z}_2}^w$, we define

$$[b_0, b_1, \cdots, b_{w-1}]P = b_0P + b_1 2^d P + b_2 2^{2d} P + \cdots + b_{w-1} 2^{(w-1)d} P.$$

The comb method considers that $k$ is represented by a matrix of $w$ rows and $d$ columns, and processes $k$ columnwise.

**Algorithm 2: Fixed-Base Comb Method**
*Input: a positive integer* $k = (k_{l-1}, \cdots, k_1, k_0)$*, an elliptic curve point* $P$ *and a window*
*width* $w$ *such as* $w \geq 2$.
*Output:* $kP$.
*1.* $d = \lceil \frac{l}{w} \rceil$.
*2. Precomputation: compute* $[b_{w-1}, \cdots, b_1, b_0]P$ *for all* $(b_{w-1}, \cdots, b_1, b_0) \in {\mathbb{Z}_2}^w$.
*3. By padding* $k$ *on the left with 0's if necessary, write* $k = K^{w-1} \parallel \cdots \parallel K^1 \parallel K^0$
*where each* $K^j$ *is a bit-string of length* $d$*. Let* $K_i^j$ *denote the* $i$*-th bit of* $K^j$.
*4.* $Q \leftarrow [K_{d-1}^{w-1}, \cdots, K_{d-1}^1, K_{d-1}^0]P$.
*5. For* $i$ *from* $d - 2$ *down to* $0$ *do*
    *5.1* $Q \leftarrow 2Q$
    *5.2* $Q \leftarrow Q + [K_i^{w-1}, \cdots, K_i^1, K_i^0]P$.
*6. Return* $Q$.

## 3 Side Channel Attacks on ECC

In this section we review the side channel attacks. We will present the Simple Power Analysis (SPA), the Differential Power Analysis (DPA), and the recent DPA attacks, especially the Refined Power Analysis (RPA) introduced by Goubin [Gou03] and the Zero-value Point Attack proposed by Akishita and Takagi [AT03], which manage to break elliptic curve cryptosystems even if the previously known countermeasures are used.

### 3.1 SPA and Countermeasures

The Binary method computes an adding and doubling operation if the bit $k_i = 1$, and only a doubling if $k_i = 0$. By observing the power consumption of the device,

an SPA attacker can detect the value of the secret bit. To prevent SPA attacks, many countermeasures have been proposed; the standard approach is to use fixed pattern algorithms [Cor99, Mon87, LD99] that compute an adding and doubling operation for each bit of the secret key.

Another type of countermeasures include indistinguishable addition formulae [JQ01, LS01], but they can not be applied on general elliptic curves. To prevent SPA attacks against an algorithm using pre-computed points, Möller [Möl01] uses a new representation without zero digits for the secret scalar, which ensures a fixed pattern computation for the algorithm.

### 3.2 DPA and Countermeasures

DPA is a more sophisticated side channel attack introduced by Kocher et al [KJJ99]; they interpret with statistical tools the results collected from several executions of the algorithm. In order to resist to DPA, the randomization of parameters seems to be an efficient technique. Coron [Cor99] proposed to randomize the secret scalar $k$ by replacing it by $k+r\#E$, where $r$ is a 20-bit random integer and $\#E$ is the number of points of the curve; the security depends on the size of $r$. Coron proposed also to blind the base point $P$ by replacing it by $P+R$ where $R$ is a random point. The efficiency of these measures has been questioned by Okeya and Sakurai [OS02], whose attack exploits the correlation between the power consumption and the weight of the data loaded from a precomputed table.

Another approach for preventing DPA is to randomize the base point representation; this can be done by using the Coron [Cor99] or the Joye and Tymen [JT01] techniques. The first method transforms an affine point $P = (x, y)$ in randomized Jacobian coordinates $P = (r^2x, r^3y, r)$ for a random non-zero integer $r$, thus the power consumption is also randomized during the algorithm execution. Joye and Tymen use a random curve belonging to an isomorphism class of the elliptic curve; a point $P = (x, y)$ of an elliptic curve $E$ is transformed into $P' = (r^2x, r^3y)$ which is a point of the corresponding isomorphic curve $E'$ of $E$.

### 3.3 RPA, ZPA, and Geiselmann–Steinwandt's Attacks

The refined-power analysis (RPA) proposed by Goubin [Gou03] uses special points to deduce the bits of the secret key. The fundamental remark of Goubin is that randomizing points with a 0-coordinate ($(x, 0)$ or $(0, y)$) yields points that possess still a 0-coordinate. Supposing that the bits $k_{l-1}, \cdots, k_{j+1}$ of the secret scalar $k$ are known by the attacker, and that he wants to guess the value of the next bit $k_j$, he just needs to choose the point $P = (c^{-1} mod \#E)(0, y)$ with $c = 2^j + \sum_{i=j+1}^{l-1} 2^i k_i$. If, in the process of the computation of $kP$, the scalar multiplication computes the point $cP = (0, y)$, the power consumption for the next step is significantly distinct. Thus, the attacker can know whether $cP$ has been computed or not, and hence if $k_j$ was 1 or 0. Iterating this process, all bits of the secret key can be determined.

Akishita–Takagi [AT03] generalized Goubin's idea to elliptic curves without points with a 0-coordinate. Their attack focuses on the auxiliary registers which

might contain a zero value, when the adding and doubling operations are performed by the scalar multiplication. The ZPA attack is in particular efficient on several standard curves with no 0-coordinate point.

Another attack exploiting the points with a 0-coordinate was presented by Geiselmann and Steinwandt [GS04] to break the SPA-resistant width-$w$ method [Möl01]. The authors noticed that the smartcard's power consumption depends on the Hamming weight of the data transported over the bus. By observing this consumption during the calculation of $kP$, an attacker can know whether a point $P$ with a 0-coordinate is fetched from the precomputed table or not, and thus can directly detect the digits $k_i$ such as $k_i = 1$. By repeating this process for the points $P_b = b^{-1}P$, where $b \in \{2^w, 2, \cdots, 2^w - 1\}$, all digits $k_i = b$ of the secret scalar $k$ will be revealed.

## 4 Proposed Countermeasures

The execution of a SPA attack on the fixed-base comb method can allow to deduce the bits of the secret scalar. This is because the comb method performs only a doubling operation if the bit-string $[K_i^{w-1}, \cdots, K_i^1, K_i^0]$ is equal to zero, and an adding and doubling operation in the other case; thus, the analysis of power consumption during the execution of the algorithm can reveal whether the bit-string $[K_i^{w-1}, \cdots, K_i^1, K_i^0]$ is zero or not. Since the probability to have a zero bit-string ($[K_i^{w-1}, \cdots, K_i^1, K_i^0] = (0, \cdots, 0)$) is less important than the probability to have a single zero bit ($k_i = 0$), the comb method offers a better resistance against SPA attacks than the binary method, but it is not totally secure against them.

Geiselmann and Steinwandt's attack will be efficient against the fixed-base comb method, since it uses a precomputed table, even if the usual randomization techniques are used. The comb method is also not secure against the more powerful RPA and ZPA attacks.

In this section, we propose two countermeasures that aim to secure the comb method against SCA attacks. First, we propose to convert the comb method into a SPA-resistant scheme by changing the representation of the scalar $k$; the use of Randomized Linearly-transformed Coordinates [IIT04] with the obtained scheme achieves the security against SCA attacks. The second countermeasure renders the fixed-base comb method SCA-resistant by using a Randomized Initial Point.

### 4.1 Changing the Representation of $k$ and Using RLCs

Our first aim is to generate a new representation of $k$ as a sequence of bit-strings different from zero, so as to thwart the SPA attack. For this purpose, we modify the binary representation of $k$ by eliminating all its zero bits and using only bits equal to 1 or $-1$. We then combine the SPA-resistant algorithm with Randomized Linearly-transformed Coordinates [IIT04].

**A New Representation for $k$.** Our algorithm replaces every zero bit of the scalar $k$ by 1 or $-1$, depending of its neighbour bits. Assuming that $k$ is odd, let $k_i$ be the first bit equal to 0. We then replace $k_i$ by $k_i + 1 = 1$ and $k_{i-1}$ by $k_{i-1} - 2 = -k_{i-1} = -1$, which does not change the value of $k$. We can iterate this process, given that each time we come to a $k_i$ equal to 0, $k_{i-1}$ is equal to 1. Finally, every bit of the bit-strings will be different from zero and the multiplication $kP$ will be SPA-resistant.

But we have also to make the generation of our new representation SPA-resistant. In the present state, a bit of $k$ is either touched if it is a zero bit or kept unchanged otherwise; hence, a SPA attack on this algorithm can occur. To deal with this threat, we modify our method to ensure that each bit is touched, independently of its value (0 or 1).

If $k$ is even, we make $k' = k+1$, and we compute $k'P$. The result of the scalar multiplication $kP$ is then recovered by performing the substraction $k'P-P$. This too might give way to a SPA attack, due to the difference in the treatment of even and odd scalars. So we convert as well the odd scalars $k$ to $k' = k+2$, and recover in this case $kP$ by performing the substraction $k'P - 2P$. Finally, we arrive at the following algorithm:

**Algorithm 3: Modified Binary Representation**
*Input: An integer* $k = (k_{l-1}, \cdots, k_1, k_0)_2$.
*Output: Modified bits* $(k_{l-1}, \cdots, k_0)$ *of* $k$, $k_i \in \{-1, 1\}$.
*1. If* $k \mod 2 = 0$ *then* $k \leftarrow k + 1$ *else* $k \leftarrow k + 2$.
*2. For* $i = 1$ *to* $l - 1$ *do*
*2.1* $b[0] \leftarrow k_i$, $b[1] \leftarrow k_{i-1}$, $b[2] \leftarrow -k_{i-1}$
*2.2* $k_i \leftarrow b[1 - k_i]$, $k_{i-1} \leftarrow b[2 - k_i]$.
*3. Return* $(k_{l-1}, ...., k_0)$.

It is clear that all the new bit-strings $[K_i^{w-1}, \cdots, K_i^1, K_i^0]$, for $i = 0, \cdots, d-2$, obtained from the new digits of the scalar $k$ output by Algorithm 3, are different from zero. Thus, Algorithm 2 combined with the use of this new sequence of bits-strings constitutes a SPA-resistant comb method.

**The Size of the Table.** The fixed-base comb method [BHLM01] precomputes the points $[b_{w-1}, \cdots, b_1, b_0]P$ for all $(b_{w-1}, \cdots, b_1, b_0) \in {\mathbb{Z}_2}^w$. Thus, the number of points stored in the precomputed table is $2^w - 1$.

On the other hand, the set of points stored in the precomputed table of the proposed scheme is $\{[b_{w-1}, \cdots, b_1, b_0]P$, for all $b_i = \pm 1\}$, which may be represented as $\{-Q, Q\}$, where $Q = \{[b_{w-1}, \cdots, b_1, b_0]P$, with $b_0 = 1\}$. We need hence only to store in the precomputed table the points $[b_{w-1}, \cdots, b_1, 1]P$ with $b_i = \pm 1\}$, and the number of the points stored in the precomputation phase of the proposed scheme is $2^{w-1}$, which is about the half of what is required by the fixed-base comb method.

**Security Considerations.** This section discusses the security of the proposed scheme against the SPA, DPA and second-order DPA, Geiselmann and Steinwandt, RPA and ZPA attacks.

• SPA: as explained before, our method builds a new sequence of bit-strings which form the new scalar's representation, in which all the bit-strings are different from zero. At each step, the main phase of the multiplication algorithm performs then exactly an adding and doubling operation, and the elliptic curve scalar multiplication behaves in a fixed pattern. Consequently, the execution of a SPA attack can not reveal any information on the bits of the secret scalar.

• DPA and second-order DPA: the use of projective randomization methods, such as randomized projective coordinates[Cor99] or random isomorphic curves [JT01], prevents DPA attacks. Okeya and Sakurai's second-order DPA attack [OS02] may be applied against the proposed algorithm, since it uses a precomputed table. For each bit-string, we access the table to get a point $[K_i^{w-1}, \cdots, K_i^1, K_i^0]P$ to be added to $Q$. An attacker could thus manage to detect whether or not a bit-string $[K_i^{w-1}, \cdots, K_i^1, K_i^0]$ is equal to $[K_j^{w-1}, \cdots, K_j^1, K_j^0]$, by monitoring some information related to the power consumption. To prevent this attack, we propose to change the randomization of each precomputed point after getting it from the table. Thus, even if we have got the same point for different bit-strings, the new point randomization implies that we load a different data.

• Geiselmann and Steinwandt, RPA and ZPA attacks: to prevent these attacks, we propose to use the Randomized Linearly-transformed Coordinates (RLC) introduced by Itoh and al [IIT04]. This technique converts a point $(x, y, z)$ into a randomized point $(x', y', z')$ such as

$$x' = \lambda_x(\lambda)(x - \mu_x) + \mu_x,\ y' = \lambda_y(\lambda)(y - \mu_y) + \mu_y,\ z' = \lambda_z(\lambda)(z - \mu_z) + \mu_z$$

where $\lambda_x, \lambda_y, \lambda_z$ are functions of $\lambda$ and $\mu_x, \mu_y, \mu_z$.

The RLC technique with $\mu_x, \mu_y \neq 0$ allows to randomize also the points with a 0-coordinate and all the intermediate values, and thus makes the proposed algorithm secure against RPA, ZPA and Geiselmann-Steinwandt's attacks.

**Computation Cost.** The multiplication algorithm performs now an adding (A) and doubling (D) operation at each step, so the cost of the main computation phase is $[d-1](A+D)$.

Now, we evaluate the cost of the precomputation phase. In this phase, we generate the sequence of points $[b_{w-1}, \cdots, b_1, 1]P$, for all $(b_{w-1}, \cdots, b_1, 1) \in \mathbf{Z}_2^w$, such as

$$[b_{w-1}, \cdots, b_1, 1]P = b_{w-1}2^{(w-1)d}P + \cdots + b_2 2^{2d}P + b_1 2^d + P.$$

To perform the precomputing phase, we first compute $2^dP,\ 2^{2d}P, \cdots, 2^{(w-1)d}P$, which will cost $((w-1)d)$ doubling operations. The second step consists in computing all possible combinations $b_{w-1}2^{(w-1)d}P + \cdots + b_2 2^{2d}P + b_1 2^d + P$, where $b_i \in \{-1, 1\}$, for $i = 2, \cdots, w-1$. The cost of this second step is at most

$2^w - w$ adding operations for $w = 2,\ 3,\ 4,\ 5$ (which are the optimum choices for $w$ in elliptic curve cryptosystems). The total cost of the precomputing phase is thus

$$[(w-1)d]D + [2^w - w]A$$

where $A$ and $D$ denote and adding an doubling point operation, and the total cost of the proposed method including efforts for preventing SPA attacks is

$$[wd-1]D + [2^w - w + d - 1]A.$$

We recall that the precomputation phase of the fixed-base comb method computes $[b_{w-1}, \cdots, b_1, b_0]P$ for all $(b_{w-1}, \cdots, b_1, b_0) \in \mathbb{Z}_2^w$. The cost of this phase is $2^w-(w+1)A+(w-1)dD$, since we have to compute $2^dP, 2^{2d}P, \cdots, 2^{(w-1)d}P$ and all possible combinations $\sum_{i=r}^{i=s} b_1 2^{id}P$, with $b_i \in \{0,1\}$, and $0 \leq r < s \leq w-1$. As the main phase computes an adding and doubling point at each step, the total cost of fixed-base comb method is

$$[wd-1]D + [2^w - w + d - 2]A.$$

The following table compares the efficiency of the proposed method with that of the fixed-base comb method, including only efforts for preventing SPA attacks and using randomized Jacobian coordinates. $S$ will denote the number of points stored in the precomputation phase, and $T$ the number of field multiplications; $k$ is a scalar with length 163 ($log_2(k) = 163$).

| Method | $w=2$ | | $w=3$ | | $w=4$ | | $w=5$ | |
|---|---|---|---|---|---|---|---|---|
| | S | T | S | T | S | T | S | T |
| Fixed-base comb method | 3 | 3010 | 7 | 2808 | 15 | 2710 | 31 | 2780 |
| Proposed method | 2 | 3026 | 4 | 2824 | 8 | 2726 | 16 | 2796 |

In terms of memory consumption, it is clear that the proposed algorithm is more interesting than the fixed-base comb method. Furthermore, since the number of points stored by the proposed algorithm is about the half of what is required by the fixed-base comb method, the cost of the efforts dedicated to prevent recent DPA attacks (second-order DPA, RPA ...) for the proposed method will not be more important.

### 4.2 Fixed-Base Comb Method with Randomized Initial Point (RIP)

In this section, we propose to use another countermeasure to prevent SPA and DPA attacks, based on the Randomized Initial Point method [IIT04, MMM04]. This method introduces a random point $R$, computes $kP+R$, and substracts $R$ to recover $kP$. The following algorithm implements the proposed countermeasure.

**Algorithm 4: Extended Fixed-Base Comb Method with RIP**
*Input: a positive integer* $k = (k_{l-1}, \cdots, k_1, k_0)$*, an elliptic curve point* $P$ *and a window width* $w$ *such as* $w \geq 2$.
*Output:* $kP$.
1. $d = \lceil \frac{l}{w} \rceil$.
*2.* $R \leftarrow Randompoint()$.
*3. Precomputation: compute* $[b_{w-1}, \cdots, b_1, b_0]P - R$ *for all* $(b_{w-1}, \cdots, b_1, b_0) \in {\mathbb{Z}_2}^w$.
*4. By padding* $k$ *on the left with 0's if necessary, write* $k = K^{w-1} \parallel \cdots \parallel K^1 \parallel K^0$*, where each* $K^j$ *is a bit-string of length* $d$*. Let* $K_i^j$ *denote the* $i$*-th bit of* $K_i^j$.
*4.* $Q \leftarrow ([K_{d-1}^{w-1}, \cdots, K_{d-1}^1, K_{d-1}^0]P - R) + 2R$.
*5. For* $i$ *from* $d-2$ *down to* $0$ *do*
    *5.1* $Q \leftarrow 2Q$
    *5.2* $Q \leftarrow Q + [K_i^{w-1}, \cdots, K_i^1, K_i^0]P - R$.
*6. Return* $Q - R$.

$Randompoint()$ is a function that generates a random point $R$ on the curve. The simplest way to obtain a random point $R$ is to generate a random $x$-coordinate and to compute the corresponding $y$-coordinate if it exists, but this process is probabilistic and may require many computations. The optimized way is to randomize a fixed stored elliptic curve point $Q$ by using Randomized Projective Coordinates [Cor99].

In our extended fixed-base comb method, the point $\Theta$ stemming from the multiplication of $P$ by a bit-string equal to zero is replaced by a random point $R$, which implies that the algorithm computes the scalar multiplication with a uniform behaviour, performing exactly an adding and doubling operation at each step. Consequently, the execution of a SPA can not reveal any bit of the secret key. Since $R$ is chosen randomly by some way mentioned above, all intermediate values will also be randomized; hence, the algorithm will also resist to DPA, Geiselmann and Steinwandt's attack, RPA and ZPA. But even if all intermediate values are randomized, the algorithm is still vulnerable to a second-order DPA. To prevent this attack, we have to rerandomize each precomputed point after getting it from the table.

**Computation Cost.** To perform the precomputation phase, we first compute $(P - R)$, $2^dP$, $2^{2d}P, \cdots$, $2^{(w-1)d}P$, and then compute all possible combinations $b_{w-1}2^{(w-1)d}P + \cdots + b_2 2^{2d}P + b_1 2^d + (P - R)$, where $b_i \in \{-1, 1\}$, for $i = 2, \cdots, w-1$, in the same manner as in Section 4.1.4. The cost of this precomputation phase is $[(w-1)d]D + [2^w - w + 1]A$. The cost of the main phase is $dD + [d+1]A$, and the total cost including efforts for preventing SPA, DPA, RPA and ZPA is thus

$$[wd]D + [2^w - w + d + 2]A.$$

As explained before, to prevent the second-order DPA, we have to randomize each precomputed point after getting it from the table, which takes $5dM$, where

$M$ denotes a field multiplication. The following table gives in percentage the increase in computation time of the proposed method with respect to the original method, using Jacobian coordinates.

| $log_2(k)$ | $w = 2$ | $w = 3$ | $w = 4$ | $w = 5$ |
|---|---|---|---|---|
| 163 | 1.43% | 1.74% | 1.92% | 1.93% |
| 210 | 1.08% | 1.35% | 1.48% | 1.52% |

## 5 Conclusion

In this paper, we have presented new SCA-resistant countermeasures based on the fixed-base comb method for the scalar multiplication. The first method proposed converts the comb-method to an SPA-resistant scheme by changing the representation of the scalar. The obtained scheme is combined with Randomized Linearly-transformed Coordinates to achieve resistance against SCA attacks, with optimized performances. Indeed, the proposed scheme requires to store only $2^{w-1}$ points in a precomputed table instead of $2^w - 1$ for the fixed-base comb method, without increasing the amount of the computation time. We have also presented an extended fixed-base comb method with a RIP, which is resistant against the SPA, DPA, RPA and ZPA attacks. The cost of the proposed countermeasures does not exceed 2% with respect to the cost of the fixed-base comb method.

## References

[AT03] T. Akishita, T. Takagi. *Zero-value point attacks on elliptic curve crytosystems.* In: ISC 2003, vol. 2851, LNCS, pp. 218–233, Springer-Verlag, 2003.

[BHLM01] M. Brown, D. Hankerson, J. Lopez, A. Menezes. *Software implementation of the NIST elliptic curves over prime fields.* In: Progress in Cryptology CT-RSA 2001, D. Naccache, editor, vol 2020, LNCS, pp. 250–265, 2001.

[CMO98] H. Cohen, A. Miyaji, T. Ono. *Efficient elliptic curve exponentiation using mixed coordinates.* In: Advances in cryptology-Asiacrypt'98, K. Ohta and D. Pei, editors, vol 1514, LNCS, pp. 51–65, 1998.

[Cor99] J.S. Coron. *Resistance against differential power analysis for elliptic curve cryptosystems.* In: Cryptography Hardware and Embedded Systems-CHES'99, C.K. Koç and C. Paar, editors, vol 1717, LNCS, pp. 292–302, 1999.

[GS04] W. Geiselmann, R. Steinwandt. *Power attacks on a side-channel resistant elliptic curve implementation,* Information Processing Letters, vol 91, pp. 29–32, 2004.

[Gou03] L. Goubin. *A refined power-analysis attack on elliptic curve cryptosystems.* In: Public Key Cryptography International Workshop-PKC 2003, vol 2567, LNCS, pp. 199–210, 2003.

[IIT04] K. Itoh, T. Izu, M. Takenaka. *Efficient countermeasures against power analysis for elliptic curve cryptosystems.* In: Proceedings of CARDIS-WCC 2004.

[JQ01] M. Joye, J.J. Quisquater. *Hessian elliptic curves and side-channel attacks.* In: Cryptography Hardware and Embedded Systems-CHES'01, C.K. Koç, D. Naccache and C. Paar, editors, vol 2162, LNCS, pp. 412–420, 2001.

[JT01] M. Joye, C. Tymen. *Protections against differential analysis for elliptic curve cryptography: an algebraic approach.* In: Cryptography Hardware and Embedded Systems-CHES'01, C. Koç, D. Naccache and C. Paar, editors, vol 2162, LNCS, pp. 386–400, 2001.

[KJJ99] P. Kocher, J. Jaffe, B. Jun. *Differential power analysis.* In: Advances in Cryptology-CRYPTO'99, M. Wiener, editor, vol 1666, LNCS, pp. 388–397, 1999.

[Koc96] P. Kocher. *Timing attacks on implementations of Diffie-Hellman, RSA, DSA and other systems.* In: Advances in Cryptology-CRYPTO'96, N. Koblitz, editor, vol 1109, LNCS, pp. 104–113, 1996.

[LD99] J. Lopez, R. Dahab. *Fast multiplication on elliptic curves over $GF(2^m)$ without precomputation.* In: Cryptography Hardware and Embedded Systems CHES'99, C. Koç and C. Paar, editors, vol 1717, LNCS, pp. 316–327, 1999.

[LL94] C. Lim, P. Lee. *More flexible exponentiation with precomputation.* In: Advances in Cryptology-CRYPTO'94, vol 839, LNCS, pp. 95–107, 1994.

[LS01] P.V. Liardet, N. Smart. *Preventing SPA/DPA in ECC systems using the Jacobi form.* In Cryptography Hardware and Embedded Systems-CHES'01, C. Koç, D. Naccache and C. Paar, editors, vol 2162, LNCS, pp. 401–411, 2001.

[MMM04] H. Mamiya, A. Miyaji, H. Morimoto. *Efficient Countermeasures against RPA, DPA, and SPA.* In Cryptography Hardware and Embedded Systems-CHES'04, M. Joye, J.J. Quisquater, editors, vol 3156, LNCS, pp. 343–356, 2004.

[Möl01] B. Möller. *Securing elliptic curve point multiplication against side-channel attacks.* In: Information Security, G.I. Davida and Y. Frankel, editors, vol 2200, LNCS, pp. 324–334, 2001.

[Mon87] P.L. Montgomery. *Speeding up the Pollard and elliptic curve methods of factorization,* Mathematics of Computation, 48(177), pp. 243–264, January 1987.

[OS02] K. Okeya, K. Sakurai. *A Second-Order DPA attacks breaks a window-method based countermeasure against side channel attacks,* Information Security Conference (ISC 2002), LNCS 2433, pp. 389–401, 2002.

# An Email Worm Vaccine Architecture

Stelios Sidiroglou, John Ioannidis, Angelos D. Keromytis, and Salvatore J. Stolfo

Department of Computer Science, Columbia University
{stelios, ji, angelos, sal}@cs.columbia.edu

**Abstract.** We present an architecture for detecting "zero-day" worms and viruses in incoming email. Our main idea is to intercept every incoming message, pre-scan it for potentially dangerous attachments, and only deliver messages that are deemed safe. Unlike traditional scanning techniques that rely on some form of pattern matching (signatures), we use behavior-based anomaly detection. Under our approach, we "open" all suspicious attachments inside an instrumented virtual machine looking for dangerous actions, such as writing to the Windows registry, and flag suspicious messages. The attachment processing can be offloaded to a cluster of ancillary machines (as many as are needed to keep up with a site's email load), thus not imposing any computational load on the mail server. Messages flagged are put in a "quarantine" area for further, more labor-intensive processing. Our implementation shows that we can use a large number of malware-checking VMs operating in parallel to cope with high loads. Finally, we show that we are able to detect the actions of all malicious software we tested, while keeping the false positive rate to under 5%.

## 1 Introduction

Recent incidents have demonstrated the ability of email-based worms and viruses ("malware") to infect large numbers of hosts very rapidly [1, 2]. Email malware propagates as executable attachments that users are tricked into opening, thus causing the malignant code to run and propagate, usually by sending copies of itself to all the entries in the user's address file. While email attachments are not the only vector by which malware propagates, they pose a substantial threat that merits special treatment, especially since attachments have the advantage (from the defender's perspective) that they can be caught before they hit the user's machine. There are numerous approaches to defending against malicious software, the usual example being the various antivirus packages.

Virus scanners are predominately signature-based, identifying security threats by scanning files for certain byte sequences that match already known patterns of malicious code. This translates to a constant need for maintaining an up-to-date signature database. The problem is further exacerbated by the lag in the cycle of detecting a new attack and the deployment of the corresponding signature, especially when humans are involved in the process. Many modern email-borne viruses do not rely on software bugs; rather, they rely on humans to click on the attachments, thus activating them.

The need for frequent updates and the inherent delay between the creation of malicious software, and the detection and deployment of signatures or patches relegate signature-based techniques to a secondary role in the active security of systems.

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 97–108, 2005.

Behavior-based mechanisms characterize software based on the perceived effects that the program has on the examined system, instead of relying on distinct signatures of that software. The obvious benefit of this approach is that it can detect new attacks (no prior knowledge or signatures required) as long as there is some differentiation between the behavior of a malicious and normal piece of software. The majority of these behavior-based systems rely on anomaly detection algorithms for their classification and thus detection of malignant code. Anomaly-detection algorithms work by constructing models of normal behavior and subsequently checking observed behavior against these models for any statistically significant variations that may hint at malicious behavior. The success of an anomaly detection algorithm depends on the choice of an accurate behavior model. Current host-based IDS systems employ anomaly detection algorithms that are based on network activity, system call, and file system monitoring. The reason behind the absence of reliable host-based IDS that are based on the forementioned models has primarily to do with the overbearing computational overhead associated with extracting behavior models from irregular and high-volume events. In particular, analyzing all system calls in a system imposes a considerable overhead due to the sheer volume of events; correlating this with the highly irregular nature of system calls in general imposes a considerable computational overhead with a high false positive rate as a further disadvantage.

Our approach combines the ability of a host-based IDS to detect previously unseen malware with the concept of a mail-server based filtering solution. To wit, we scan each incoming mail message at the mail server for potentially dangerous attachments. Every such attachment is sent to one of a set of protected environments running various mail readers (MUAs — Mail User Agents) along with a host-based IDS. In our particular instance, the IDS looks for registry accesses which the MUA is not likely to perform. Using the Windows registry allows for a more accurate correlation of events given that it is an integral component of the Windows operating system and helps reduce the false positive rate associated with detecting malicious behavior. The protected environment opens each executable attachment, runs it, and if the IDS detects suspicious behavior, it notifies the mailserver to discard the corresponding email message. The entire environment is running under VMware so that no clean-up needs to be performed; the VM is discarded and a new one is spawned for each new check.

The advantage of such an approach is that adding compute power (faster and/or more machines) to the checking components allows a site to customize the resources needed for defense to its needs. Different environments can be set up running different MUAs, selected based on the local user population. Such an approach also does not preclude using traditional techniques such as pattern-matching to catch known viruses.

Our implementation shows that we can use a large number of malware-checking VMs operating in parallel to cope with high loads. The average time for downloading the message, detecting the attack and updating the MTA message queue was 28 seconds. Finally, we show that we are able to detect the actions of all malicious software we tested, while keeping the false positive rate to under 5%. Combining additional detectors will enable the use of data correlation algorithms that can be used, in turn, to reduce the false positive rate.

## 2 System Architecture

Our architecture makes use of several components that have been developed for other purposes. Its novelty lies in the combination of all the components in detecting and deterring zero-day email worms from propagating using an entirely automated process. As illustrated in Figure 1, our system consists of three main components:

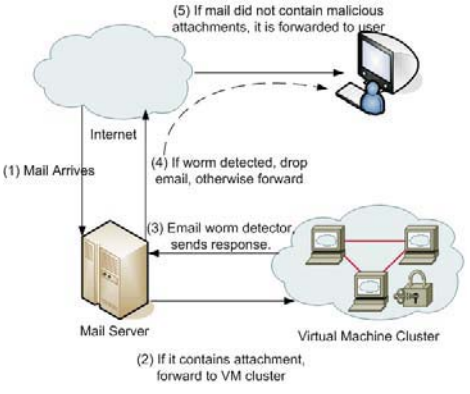



**Fig. 1.** System Architecture

- A virtual machine cluster, which houses protected environments that run instances of different Mail User Agents (MUAs) and operating systems.
- A host-based IDS that is responsible for detecting anomalous behavior.
- An email-worm-vaccine aware Mail Transport Agent (MTA) that classifies and manages potentially malicious email messages.

In the following sections we discuss the individual components in detail.

### 2.1 Virtual Machine

Host-based Intrusion Detection Systems (IDS) must, by definition, run the potentially malicious application on the host machine. Allowing the attack to run locally renders that particular machine useless for further use. For this reason, the malicious software is tested on an isolated, controlled environment that will provide the level of protection required. An ideal candidate is a virtual machine environment that can be effectively flushed after each use without further impact to the underlying system. Specifically, we use VMware images that contain the base system that we use across the VM-cluster, which has the advantage of providing an identical test case for use with the host-based IDS. An additional benefit of using a centralized VM-based architecture is that we avoid the need to deploy IDS and mail filtering software on large numbers of desktops.

### 2.2 Host-Based Intrusion Detection

In order to be able to detect zero-day email worms, we need a non signature-based approach. For this purpose, we employ a behavior based mechanism as the anomaly detection component of our architecture.

Behavior-based mechanisms extract and characterize software based on the perceived effects that the program has on the examined system in lieu of relying on distinct signatures of that software. The obvious benefit of this approach is that it can detect new attacks (no prior knowledge or signatures required) as long as there is some differentiation between the behavior of a malicious and the behavior of a normal piece of software. The majority of these behavior-based systems rely on anomaly detection algorithms for their classification and thus detection of malignant code. Anomaly detection algorithms work by constructing models of normal behavior and subsequently checking observed behavior against these models for any variations that may hint at malicious behavior. As mentioned earlier, the success of an anomaly- detection algorithm is contingent upon the choice of behavior model. Current host-based IDS systems employ anomaly detection algorithms that are based on network activity, system call and file system monitoring. The reason behind the absence of reliable host-based IDS that are based on the aforementioned models has primarily to do with the high computational overhead associated with extracting behavior models from irregular and high-volume events. In particular, analyzing all system calls in a system imposes an considerable overhead due to the sheer volume of events; correlating this with the highly irregular nature of system calls in general imposes a considerable computational overhead with the a high false positive rate as appendage.

### 2.3 MTA

Another critical component of our architecture is an email-worm-vaccine aware Mail Transfer Agent (MTA). The purpose of this augmented MTA's components are:

- Classification and filtering of potentially malicious email
- Communication with the host-based IDS cluster
- Maintenance of message queue

The MTA, as a first line of defense, will be responsible for imposing message classification and filtering. A tightly-coupled learning component will facilitate the decision process by receiving feedback from the host-based IDS. The filtering component of the MTA will conceptually reside in front of the classification component. Filtering will be the primary means by which to avoid denial-of-service attacks on the underlying system. For example, in the case of a mass email-worm outbreak, once the IDS component identifies a message as containing a malicious payload all subsequent email containing identical payloads will be sent directly to the quarantine component, bypassing the rest of the system. This case becomes much more difficult to solve for polymorphic and metamorphic email-worms; Spinellis [3] shows that it is an NP-hard problem. The only viable plan of action in the presence of a high-volume polymorphic outbreak would be to filter all incoming email that fit the high-level characteristics (having an attachment or originating from a particular source) by either pushing them directly to the quarantine or replying with a 451 ("transient error, try again later") message.

Classification of messages would be performed on the basis of a set of heuristics such as the presence of attachments or embedded URLs. Once a message has been classified as suspicious, it is sent to the host-based IDS VM cluster. At that point, the messages are placed in temporary queues waiting for a decision from the IDS.

### 2.4 Mail User Agent

The final component of the system architecture is the Mail User Agent (MUA). The primary purpose of the MUA is the retrieval and execution of potentially malicious email. The MUA, in turn, simulates the behavior of a naïve user by opening all email attachments and clicking on all available URLs. The reason that we use an MUA instead of simply downloading the email directly is so that we can expose any vulnerabilities that are directly related to using the particular MUA.

## 3 Implementation

Our prototype implementation, shown in Figure 2(left) consists of four components: RAD [4], VMware [5], Postfix [6] and Microsoft Outlook [7]. These components interact to provide a secure environment, detect anomalous behavior, manage email queues and simulate naive user behavior respectively. In Section 4 we evaluate the performance of our approach. Here, we introduce the components and discuss the implementation.

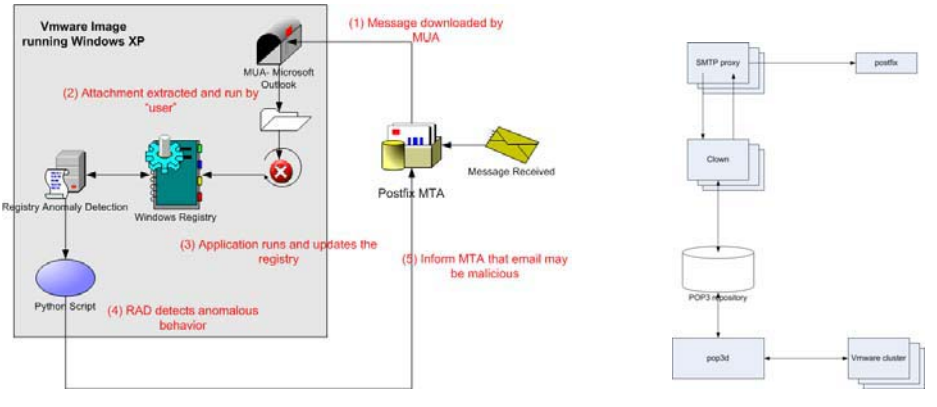



**Fig. 2.** System Implementation (left), with MTA details (right)

### 3.1 RAD

In order to detect anomalous behavior, namely email worms, we employ a RAD (Registry Anomaly Detection) [4], which monitors in real-time accesses to the Windows registry and detects malicious behavior.

The Windows Registry is an integral component of the Windows operating system, frequently used by a majority of programs. These characteristics elevate the Registry to prime candidate position as source of audit data. RAD attaches a sensor on the Registry and applies the acquired information to an anomaly detector that can correlate activity that corresponds to malicious software.

The main advantage of using RAD is its ability to accurately detect anomalous behavior with a low computational overhead. The low overhead makes it a viable solution for real-time detection of malicious software.

RAD constructs a data model from five features extracted directly from the registry sensor. These features are: the name of the process accessing the registry, the type of query sent to the registry, the key that is being accessed, the response from the registry, and the value of the key that is being accessed.

Using the features thus monitored from the registry accesses, RAD builds a model from normal (non-attack) data. This model is then used to classify registry accesses as either normal or malicious.

### 3.2 VMware

VMware allows multiple virtual machines, each running its own operating system, to co-exist on a single real machine. Potentially dangerous applications can thus be isolated from each other by running them in separate virtual machines. We prepare a single VMware image that contains an already trained model for our host-based IDS and the applications that we are testing, namely, standard Microsoft products (Office, Outlook, Outlook express, Messenger) and various other popular applications.

The image is used for a single detection session; testing a single email attachment at a time. For this purpose we set the VMware disk mode to "non-persistent, so that any changes made to "disk" are lost when the virtual machine is terminated. Having the disk in nonpersistent mode allows for one additional advantage, the use of the repeatable resume feature. Repeatable resume allows for a virtual machine to quickly start from a resumed state bypassing the need to reboot the operating system any time a new virtual machine environment is needed.

### 3.3 MTA

We based our implementation on the `smtp.proxy` open-source package as a front-end for any MTA. Figure 2(right) shows the components of this implementation. `Smtp.proxy` is a relatively simple piece of code that listens on the SMTP port (port 25), waiting for incoming SMTP connections. When a connection arrives, the proxy contacts the real MTA (in our case, Postfix [6]) and goes through the initial HELO/MAIL/RCPT phase with both sides. Thus, our proxy does not have to know any special site-specific restrictions on acceptable domains, anti-spam measures, and so on, that the Postfix administrator may have set up. Configuration details such as preventing open-relays or maximizing concurrency are beyond the scope of this paper — ours is a proof-of-concept, not a highly-optimized implementation. When the remote MTA sends the DATA command, followed by the body of the email message, the proxy saves it in a uniquely-named temporary file, and invokes a script which we wrote, *clown*, after it has received the entire message, but before it responds to the DATA command of the remote MTA.

A copy of *clown* is forked for every message received; it therefore keeps a tally of how many copies of itself are currently running, waiting for the cleanup VMs to return. If a limit, chosen so that the queue of unprocessed messages does not grow steadily, is exceeded, *clown* returns a 451 ("transient error, try again later") message, which causes *smtp.proxy* to pass that on to the remote MTA so that the mail message can be processed later. The local copy is then removed.

Once *clown* receives control, it runs the file with the contents of the email message through a MIME normalizer (a separate big problem in itself, and outside the scope of this paper); it then passes a copy of the message on to one of the cleanup virtual machines and waits for the VM to finish processing. The copy passed to the VM includes an extra header with the IP address and port to contact (*e.g.,* `X-Clown: 128.59.16.20:12588`). The VM will respond with an indication as to whether the message is acceptable or not. If the message is deemed safe, *clown* will simply return with a 0 exit code, at which point *smtp.proxy* will pass the file on to the real MTA for eventual delivery. Otherwise, a 554 ("permanent error") response will be given to the proxy, which will pass it on to the remote MTA. The copy of the message is discarded, *clown* exits, and another queued message is processed.

We had a choice between a pull-model and a push-model for passing the messages on to the VM cluster. We opted for the pull-model, as it made implementation easier. To wit, *clown* deposits every message in a POP3 repository, which, for the particular POP3 server we use, happens to be the Unix mail file format. As each VM becomes available, it pulls the topmost (oldest) message from the POP3 server, processes it, and then connects to the TCP port specified in the `X-Clown:` header.

To ward against VM cluster failures or excessive load, each blocked *clown* process times out after a preset amount of time. If this timeout occurs, the corresponding message is removed from the POP3 server (if it is still there), and a 451 error code is sent to the remote MTA to indicate a transient error, so that the latter can attempt to re-send the message at a later time.

### 3.4 MUA

The Mail User Agent is the software that the user usually interacts with when dealing with email. In our architecture, the MUA is responsible for simulating the behavior of a naïve user, opening every attachment and clicking on every link. Specifically, we use the popular Microsoft Outlook MUA [7] and the EZdetach [8] plug-in. EZdetach can extract and save Outlook attachments from messages, as well as run custom scripts on these attachments.

Outlook connects to the email-worm MTA through a secure IMAP connection and downloads suspicious messages from the server. As soon as a message is downloaded, attachments are extracted and run with administrator privileges. If these attachments contain malicious logic, RAD will detect anomalous behavior and notify *clown.*

## 4 Evaluation and Results

In this section we discuss the preliminary results of our proof-of-concept implementation. The results presented in this section are a coarse indication of overall system

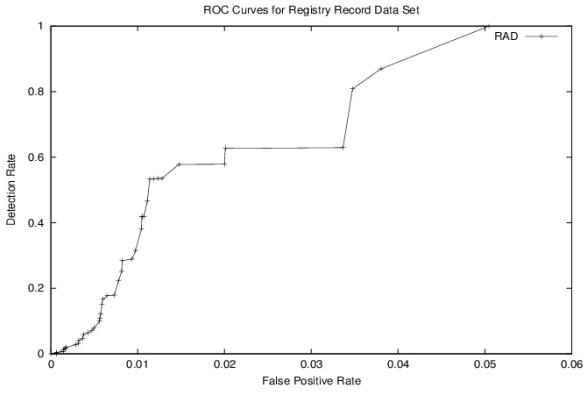



**Fig. 3.** Figure showing results of varying the threshold on the data set

performance. The optimization of the system for deployment in large-scale organizations is the subject of future work.

### 4.1 RAD

In order to evaluate the RAD system, we used the publicly available data set that can be found at `http://www.cs.columbia.edu/ids/rad`. The training data we used were collected over two days of "standard" user usage. Standard usage is defined as logging in, surfing the web, reading email, using a word processor, then logging off.

This simulated use of Windows produced a training data set (clean of attacks) consisting of 500,000 records. The attack data set (mixture of regular usage and embedded attacks) consisted of 300,000 records. The attack data set includes an array of publicly available attacks such as *Back Orifice*, *aimrecover*, *browlist*, *l0phtcrack*, *runattack*, *whackmole* and *setuptrojan*.

The natural way to evaluate the performance of an IDS is to compute the detection rate and the false positive rate. Detection rate is defined as the percentage of malicious programs that are correctly identified as anomalous by the system. The false positive rate is, in turn, defined as the percentage of normal programs diagnosed as malignant. As illustrated in Figure 3, in order to achieve $100\%$ detection rate, a false positive rate of $5\%$ needs to be tolerated.

### 4.2 Timing

To test the efficacy of our system, we evaluated the performance of our system, in terms of detection latency, against real-world exploits. The tests were conducted on a PC with a 2 GHz Intel P4 processor and 1 GB of RAM, running windows XP Professional as the host operating system and Windows XP Professional as the guest OS under VMWare Workstation version 4.0.5.

The average time for downloading the message, detecting the attack and updating the MTA message queue was 28 seconds. Downloading the message and detecting using

RAD all happen in sub-second times. The additional latency is imposed by the Microsoft Outlook MUA, as this is the minimum checking period allowed.

We can avoid this performance overhead by using a lighter-weight MUA (we implemented a simple IMAP client written in Python) but we would like to maintain the ability of detecting client-specific attacks. A hybrid approach can be used in cases where the extra overhead is not acceptable. In this scenario, the light-weight client would be used to check for suspicious attachments as a first step, and if none are found, continue with using a more widely used MUA.

Of further interest to the overall system performance is the cost of instantiating pristine VMWare images. To avoid the cost of moving around very large VMWare images which are in the range of 3GB, we set VMWare disks in non persistent mode and use the "repeatable resume" feature to quickly restart from a ready state. Restarting VMWare when using these features takes approximately 4 seconds.

**Table 1.** Daily average email statistics collected from the Columbia University Computer Science Department

| Message Avg | Virus Avg | Spam Avg | Delivered Avg |
|---|---|---|---|
| 84966 | 4922 | 29726 | 50317 |

In order to get a rough estimate on the scalability of our system, we collected statistics from the mail server at Columbia University's Computer Science department. Table 1 illustrates the daily average email characteristics for the time period of December 25th 2004 to January 24th 2005 as collected from the Sophos PureMessage email management solution. From the 50000 email messages received, approximately 8% contain attachments [9]. This translates to 4025 email attachments that need to be processed by our system. Given that processing time per email attachment is approximately 30 seconds, the system can process 3000 email attachments per day per VM. As VMWare ESX server can scale up to 80 powered-on virtual machines, our organization can rely on a single machine to handle daily email processing requirements. Obviously, these back-of-the-envelope calculations do not take into account the arrival rates and expected delay per message but they serve as a rough estimate of what resources are required to deal with an enterprise environment.

## 5 Discussion

The two major goals of our architecture are scalability and reliability. Scalability will enable the use of the email worm detection architecture in a large-scale enterprise environment. To achieve this requirement, we need to minimize the rate of false positives in the host-based IDS, and speed up the detection stage on the virtual machine. Reducing the rate of false positives can be achieved by combining the RAD system with additional detectors such as the Windows Event Log data. This combination will allow for the use of data correlation algorithms that can be used, in turn, to produce more accurate behavior models. Reducing the time needed to detect malicious activity can be achieved by retrofitting MUAs to minimize the delay of checking and downloading messages.

Reliability will help our architecture in dealing with more complex issues such as targeted attacks against the system and encrypted email. One of the fundamental assumptions that our system makes is that the virtual machine can mimic the exact behavior of an operating system. If a worm can detect the presence of a virtual machine, it could potentially vary its behavior avoiding detection. The virtual machine that we choose for deployment in our system should successfully conceal its presence to the guest operating system as much as possible. In the absence of obvious clues from the VM, there are other techniques that an attacker can use (although not as reliable) to detect the presence of a virtual machine such as timing attacks *etc*. For this purpose, we can insert logic that identifies this sort of attempts.

The advent of end-to-end encryption mandates that our architecture should include a solution to address this problem. Storing all user keys on the mail server is not the best solution to this problem. However many organizations already require all email to be decryptable by them for legal reasons (*e.g.,* SEC regulations that cover all financial institutions in the US). Providing hooks to the MUAs in the virtual machine is one possible solution. This problem remains open for future consideration.

## 6 Related Work

Computer viruses have been studied extensively over the last several years. Cohen was the first to define and describe computer viruses in their present form. In [10], he gave a theoretical basis for the spread of computer viruses. The strong analogy between biological and computer viruses led Kephart *et al.* [11] to investigate the propogation of computer viruses based on epidemiological models. They extend the standard epidemiological model by placing it on a directed graph, and use a combination of analysis and simulation to study its behavior. They conclude that if the rate at which defense mechanisms detect and remove viruses is sufficiently high, relative to the rate at which viruses spread, they can prevent widespread virus propagation. [12] describes a filesystem layer designed specifically for efficient virus scanning and removal.

Also by Zou *et al.* [13], is the work that present an email worm model that takes into account the behavior of email users, specifically, email checking frequency and the probability of opening an email attachment. They observe that the node degrees, as a logical network defined by email addresses, have heavy-tailed distributions. Their results indicate that email worms spread more quickly on a power law topology but are also easier to contain through immunization. In [14], the authors analyze network traffic traces collected for college campus environments and present an analysis of two major email-worm outbreaks, SoBig and MyDoom. Their work focuses on the effects of mass mailing worms on a single subnet. They show that both worms analyzed exhibit noticeable abnormalities in the traffic of the infected hosts.

The author of [15] proposes an automated email virus detection and control scheme using the attachment chain tracing (ACT) technique. This technique is based on epidemiological models that are used in infections disease analysis and control. The author shows how these techniques can be used for detecting and immunizing an email virus.

One approach for detecting new email viruses was described in [16], which keeps track of email attachments as they exchanged between users through a set of collaborating email servers that forward a subset of their data to a central data warehouse and correlation server. Only attachments with a high frequency of appearance are deemed suspicious; furthermore, the email exchange patterns among users are used to create models of normal behavior. Deviation from such behavior (*e.g,* a user sending a particular attachment to a large number of other users at the same site, to which she has never sent email before) raises an alarm. Information about dangerous attachments can be sent to the email servers, which then filter these out. One interesting result is that their system only needs to be deployed to a small number of email servers, such that it can examine a miniscule amount of email traffic (relative to all email exchanged to the Internet) — they claim 0.1% — before they can determine virus outbreaks and be able to build good user behavior models.

MEF [17] is a UNIX mail filter that detects known and uknown malicious windows executables. By employing data-mining techniques on a database of known malicious executables, a generalized model is extracted that can, in turn, be used to detect future instances.

The work presented by Zou *et al.* [18] is probably the most closely related work. The authors present a feedback email worm defense system that uses a multi-step system for detecting new attacks. They also discuss the idea of using a honeypot system to detect outgoing traffic. Unfortunately, they provide no implementation details and do not address any of the apparent systems issues.

## 7 Conclusion

We have a described a novel approach for scanning incoming email messages for *zero-day* worms and viruses. Briefly, we intercept all incoming messages, pre-scan them for suspicious content, and only deliver messages that are deemed safe. Instead of relying on a traditional signature-based approach, we employ a behavior approach where we actually "open" attachments inside an instrumented virtual machine looking for anomalous behavior.

We have implemented this architecture in a proof-of-concept implementation where we observe the behavior of different application on the Windows registry in real-time. We show that we are able to detect the actions of all malicious software we tested at a false positive rate of 5%. Furthermore, we show that our implementation can be offloaded to any number of ancillary machines thus minimizing the computational overhead on the mail server.

## Acknowledgements

We wish to thank Viktor Dukhovni for his invaluable help with Postfix.

## References


1. US-CERT Incident Note IN-2003-03: Sobig Worm. http://www.cert.org/incident_notes/IN-2003-03.html (2003)
2. US-CERT Technical Cyber Security Alert TA04-028A: MyDoom Virus. http://www.us-cert.gov/cas/techalerts/TA04-028A.html (2004)
3. Spinellis, D.: Reliable identification of bounded-length viruses is NP-complete. IEEE Transactions on Information Theory **49** (2003) 280–284
4. Apap, F., Honig, A., Hershkop, S., Eskin, E., Stolfo, S.J.: Detecting Malicious Software by Monitoring Anomalous Windows Registry Accesses. In: Proceedings of the $5^{th}$ International Symposium on Recent Advances in Intrusion Detection (RAID). (2002)
5. VMware. http://www.vmware.com (2004)
6. Postfix. http://www.postfix.org (2004)
7. Microsoft Outlook 2003. http://office.microsoft.com/en-us/FX010857931033.aspx (2004)
8. EZdetach. http://www.techhit.com/ezdetach/ (2004)
9. Stolfo, S.J., Li, W.J., Hershkop, S., Wang, K., Hu, C.W., Nimeskern, O.: Detecting Viral Propagations Using Email Behavior Profiles. In: ACM TOIT 2005. (2005)
10. Cohen, F.: Computer Viruses: Theory and Practice. Computers & Security **6** (1987) 22–35
11. Kephart, J.O.: A Biologically Inspired Immune System for Computers. In: Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, MIT Press (1994) 130–139
12. Miretskiy, Y., Das, A., Wright, C.P., Zadok, E.: Avfs: An On-Access Anti-Virus File System. In: Proceedings of the $13^{th}$ USENIX Security Symposium. (2004) 73–88
13. Zou, C.C., Towsley, D., Gong, W.: Email Worm Modeling and Defense. In: Proceedings of the $3^{rd}$ International Conference on Computer Communications and Networks (ICCCN). (2004)
14. Wong, C., Bielski, S., McCune, J.M., Wang, C.: A Study of Mass-Mailing Worms. In: Proceedings of the ACM Workshop on Rapid Malcode (WORM). (2004) 1–10
15. Xiong, J.: ACT: Attachment Chain Tracing Scheme for Email Virus Detection and Control. In: Proceedings of the ACM Workshop on Rapid Malcode (WORM). (2004) 11–22
16. Bhattacharyya, M., Schultz, M.G., Eskin, E., Hershkop, S., Stolfo, S.J.: MET: An Experimental System for Malicious Email Tracking. In: Proceedings of the New Security Paradigms Workshop (NSPW). (2002) 1–12
17. Schultz, M.G., Eskin, E., Zadok, E., Bhattacharyya, M., Stolfo, S.J.: Mef: Malicious email filter - a unix mail filter that detects malicious windows executables. In: Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference. (2001)
18. Zou, C.C., Gong, W., Towsley, D.: Feedback Email Worm Defense System for Enterprise Networks. Technical Report TR-04-CSE-05, Univ. of Massachussetts, ECE Department (2004)

# Enforcing the Principle of Least Privilege with a State-Based Privilege Control Model*

Bin Liang[1,2], Heng Liu[2], Wenchang Shi[3], and Yanjun Wu[3]

[1] Department of Computer Science & Technology, Tsinghua University, Beijing 100084, China
[2] Venus Info Tech Inc., Beijing 100081, China
{Liangbin, Liuheng}@venustech.com.cn
[3] Institute of Software, Chinese Academy of Sciences, Beijing 100080, China
{Wenchang, Yanjun03}@ios.cn

**Abstract.** In order to provide effective support to the principle of least privilege, considering the limitation of traditional privilege mechanisms, this paper proposes a new privilege control model called State-Based Privilege Control (SBPC) and presents the design and implementation of a prototype system for SBPC called Controlled Privilege Framework (CPF) on the Linux operating system platform. SBPC decomposes the time space of a process' lifetime into a series of privilege states according to activities of the process and its need for special permissions. The privilege state is closely related to the application logic of a process. It is the privilege state transfer event that stimulates a process to transfer from one privilege state into another one. For a specified process, there is a specific set of privileges corresponding to every privilege state of the process. With the implementation of CPF, experiment results show that fine-grain and automatic privilege control can be exercised transparently to traditional applications, threats of intrusion to a system can be reduced greatly, and support to the principle of least privilege can therefore be achieved effectively.

## 1 Introduction

A privilege is a special right that a process must possess to perform some security-relevant functions. Activities of a process to perform such kind of functions may have grave impact on various aspects of the security of a system. Control and management of these activities are of great significance to the security of a whole system. Abuse of a privilege may lead to very serious security problem. To obtain a privilege is usually the first try of a malicious user before committing an intrusion. With regard to a privileged process, an operating

* Supported by the National Natural Science Foundation of China under Grant No.60373054 and No.60073022; the National 863 High-tech Program of China under Grant No.2002AA141080; the Science and Technology Program of Haidian District under Grant No.K20044803.

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 109–120, 2005.

system trusts the application logic of the process, namely, the operating system trusts that the process can perform and only perform privileged activities in accordance with its application logic. The trust is based on the confidence in the integrity of the privileged process. However, the integrity of a privileged process may be compromised severely due to various causes, such as vulnerabilities in a system, malicious codes, buffer overflow, etc. To strengthen the control over privileged activities hence reduce the threats of intrusion, the privilege mechanism in an operating system should effectively enforce the principle of least privilege [1].

Focusing on the effective enforcement of the principle of least privilege in an operating system, this paper proposes a new privilege control model called State-Based Privilege Control (SBPC), which is application-logic-oriented and exercises privilege control in terms of privilege-state of process. A prototype system for SBPC called Controlled Privilege Framework (CPF) is designed and implemented on the Linux operating system platform. Experiment results show that CPF can provide fine-grain and automatic privilege control on an operating system and consequently can effectively support the principle of least privilege as well as can reduce the threats of intrusion to a system.

The outline of the rest of the paper is as follows: Section 2 presents our proposal for a State-Based Privilege Control (SBPC) Model. Section 3 discusses the design and implementation of SBPC via a prototype system called Controlled Privilege Framework (CPF). In Section 4, we briefly discuss the related work and in Section 5 we conclude.

## 2 The State-Based Privilege Control Model

This section states the authors' point of view on the underline principles for the design of privilege control mechanisms, puts forward the State-Based Privilege Control (SBPC) Model and presents a formal specification for the model.

### 2.1 Underline Principles

In the authors' point of view, to effectively support the principle of least privilege, the design of privilege control mechanisms should consider the following underline principles.

1. Privilege control should focus on the program logic of a process.
2. The lifetime of a process may be divided into several periods based on privilege-related attributes. A process may have different privileges in different period.
3. The privilege attribute of a user should only be used as the global constrain to process privileges.
4. With respect to some privileges, parameters of privileged activities may be used to provide finer-grain privilege control.

If a privilege mechanism is developed on an existing system, the new mechanism should be compatible with the old one (if any) and the use of them should be transparent to applications.

### 2.2 Description of the Model

The State-Based Privilege Control (SBPC) model uses Process Privilege State (PPS) to describe a process' behavior. SBPC decomposes the time space of a process' lifetime into a series of PPSs according activities of the process and its need for special permissions. The PPS is closely related to the application logic, or program logic, of a process. During the lifetime of a process, its PPS is being changed dynamically. It is the Privilege State Transfer Event (PSTE) that stimulates a process to transfer from one PPS into another one. For a specified process, there is a specific set of privileges corresponding to every privilege state of the process.

To illustrate the basic concepts in SBPC, a wu-ftpd daemon process running on a Linux operating system is quoted as an example. As shown in Fig. 1, the lifetime of the process is divided into four PPSs.

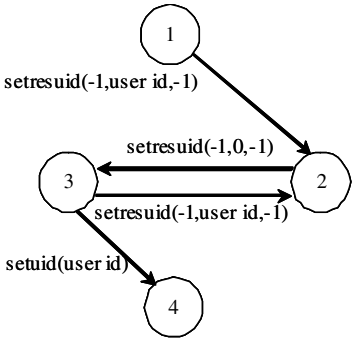



**Fig. 1.** Privilege state diagram of wu-ftpd

1. The wu-ftpd daemon starts running as root, waiting for a user to login by authentication. At this stage, all ID attributes are zero (root). After a user login, the daemon forks a service process and sets its effective and filesystem UIDs to those of the user. The process then transfer to PPS2.
2. In this state, effective and filesystem UIDs of the daemon are set to that of the authenticated user, real and saved UIDs are still zero. For the rest of this session, it is in an unprivileged state, unless it needs to perform some privileged operations.
3. When the daemon needs some privileges, it set its effective and filesystem UIDs to zero and transfers to PPS3. After privilege operation, it goes back to PPS2.

4. During operation, the authenticated user needs to execute some external commands, e.g. tar, compress, etc.. To confine the rights, the daemon transfers from PSS2 to PPS3 and then to PPS4 by forking a new process and setting its UID attribute to that of the user. After execution, the new process will terminate itself.

In PPS1, the daemon is assigned the privileges to bind ftp port and set process UIDs arbitrarily. The daemon doesnt perform any privileged operations in PPS2, but CPF restricts the parameters of the set*uid operations and the valid target PPS. In PPS3, the daemon is assigned the privileges to change root directory, to override discretionary access control restrictions, and to change the owner of arbitrary files, etc. Some system calls, e.g. execve, kill, etc., are identified as security-sensitive calls to wu-ftpd. The privilege to invoke the execve system call is assigned to the daemon in PPS4, but command-files can be executed are limited via privilege parameters.

To make the transfer of PPSs and the grant of privileges work, SBPC needs to keep track of the detail of all PPSs and privilege. It defines a Program Privilege Table (PPT) as a depository to preserve information of all PPSs and their corresponding privileges of all privileged programs. Relevant parameters should be provided for some specific privileged operations. They are also maintained in the PPT.

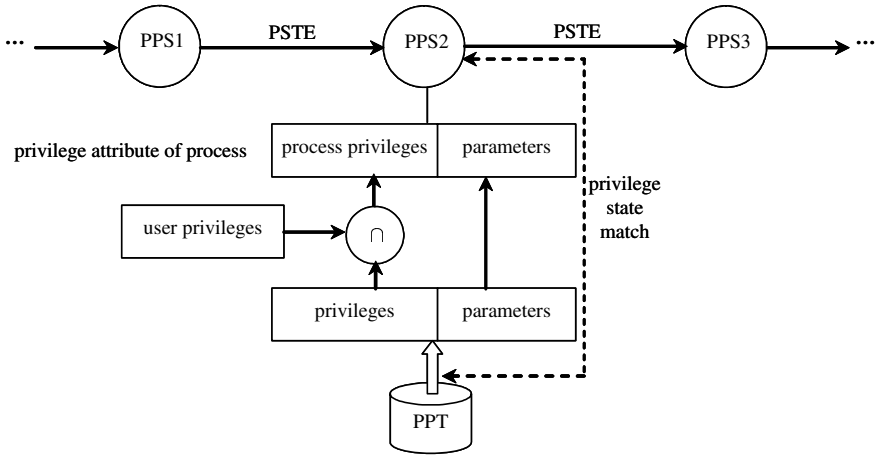



**Fig. 2.** SBPC architecture

As Figure 2 depicts, when a PSTE occurs, the system tries to look up the new PPS in the program entries of the PPT. If there is a match, the intersection of the privileges in the matched entry of the PPT and the privileges of the process current owner is assigned to the process. Additionally, to the privileges with parameter characteristics, the permitted operation parameters in the PPT are also assigned to the process. If there is no match, the privileges of the process will be set to null.

It is possible to change the current privileges of a process in SBPC, however, the result privileges must be kept as a subset of the initial privileges of the current PPS. It should be noted that to change a privilege should itself be taken as a privileged operation. It is only the process with the pertinent privilege, e.g. CAP_SETPCAP in Linux, that can perform such operation.

To change the current privileges of a process is a dangerous operation. It may incur some serious security problems [2]. In the formal specification of SBPC given next, two different secure system states are defined according to whether it is permitted to change the current privileges of a process or not.

### 2.3 Formal Specification of the Model

The state machine theory is applied to devise the formal specification of SBPC in a way similar to that in defining the BLP model [3]. A SBPC system includes an initial state $Z_0$ and a sequence of triples each of which is defined as *(request, decision, state)*. If the initial state of a system and every state in the sequence are secure, the system is a secure system.

The formal definitions of a system state and a secure system state are presented as follows. Coming first is the definition of elements of the model.

**Definition 1.** *The SBPC model consists of the following components:*

- *U: the set of users.*
- *Prog: the set of system programs.*
- *Proc: the set of system processes.*
- *PState: the set of all possible privilege states.*
- *Priv: the set of system privileges.*
- *PParam: the set of all possible parameters of privileged operations.*

**Definition 2.** *A system state v is described as a six-tuple (PPT, CP, UP, PS, CU, PP), where:*

- $PPT \subseteq \mathbf{P}(Prog \times PState \times \mathbf{P}(Priv \times \mathbf{P}(PParam)))$*, is a program privilege table, which is a relation of program, privilege state, privileges and corresponding permitted parameters of privileged operations.* $\mathbf{P}(X)$ *denotes the power set of X.*
- $CP \subseteq \mathbf{P}(Proc \times \mathbf{P}(Priv \times \mathbf{P}(PParam)))$*, is a current privilege information table of processes, which is a many-to-many relation of processes, current privileges and corresponding permitted parameters.*
- $UP : U \rightarrow \mathbf{P}(Priv)$*, is a function that maps each user* $u_i$ *to its privilege set* $UP(u_i)$*.*
- $PS : Proc \rightarrow PState$*, is a function that maps each process* $p_i$ *to its current privilege state* $PS(p_i)$*.*
- $PU : Proc \rightarrow U$*, is a function that maps each process* $p_i$ *to its current user* $PU(p_i)$*.*
- $PP : Proc \rightarrow Prog$*, is a function that maps each process* $p_i$ *to its corresponding program* $PP(p_i)$*.*

A system may be working in two different secure states. One is called stably secure state and the other liberally secure state. A stably secure state prevent the current privileges of a process from being changed, they must always be identical to the initial privileges of the PPS. A liberally secure state allows a process to change its current privileges on condition that the current privileges do not exceed the initial privileges of the PPS.

**Definition 3 (Stably Secure).** *A system state* $v = (ppt, cp, up, ps, cu, pp)$ *is stably secure iff:*
$\forall (p, x) \in cp \Rightarrow \exists a = (pp(p), ps(p), y) ppt$ *so that* $x = (y \cap (up(pu(p)) \times \mathbf{P}(PParam)))$

**Definition 4 (Liberally Secure).** *A system state* $v =$*(ppt, cp, up, ps, cu, pp) is liberally secure iff:*
$\forall (p, x) \in cp \Rightarrow \exists a = (pp(p), ps(p), y) ppt$ *so that* $x \in (y \cap (up(pu(p)) \times \mathbf{P}(PParam)))$

## 3 Development of a Prototype System

This section discusses the methods to build a prototype system for SBPC called Controlled Privilege Framework (CPF) on the Linux operating system platform. CPF is designed and implemented as a Loadable Kernel Module (LKM) based on the Linux Security Module (LSM) framework [4]. It works under the rules defined by the stably secure state, i.e. a process can by no means change its current privileges. This section also makes an experiment with the wu-ftpd daemon process to test the functionality of CPF.

### 3.1 CPF Privileges

Invoking system calls is the only way for a process to access resources. It is a good choice to identify and partition system privileges via system call security analysis. So, an analysis method similar to that used in the REMUS [5] project is utilized to identify and control those system calls that may allow full control over a system or may lead to a denial of service attack. On the other hand, the method Linux uses to partition POSIX.1e capabilities is of great worth. It covers a majority of the security-relevant system calls. To provide backward compatibility and make the functionality of the privilege mechanism transparent to traditional applications, CPF supports the functionality of the existing POSIX.1e capabilities. Both the advantage of the system call analysis method and that of the method Linux uses to partition POSIX.1e capabilities are taken in the design of the CPF privilege mechanism.

In CPF, system privileges are partitioned into 71 CPF privileges. They can be grouped into nine categories. 1: Equivalents of original Linux capabilities. 2: Refined Linux capabilities. 3: Invoking privileged system calls, e.g. _sysctl. 4: Access to security policy setting or configuration files, i.e. security database. 5: Access to system files. 6: Audit. 7: Mandatory Access Control (MAC) privilege, e.g. write-down, set security label, etc. 8: CPF control, e.g. enable, disable,

reload, etc. 9: Invoking security-sensitive system calls, e.g. execve, kill, etc., under dangerous circumstance, e.g. daemon processes providing remote service.

The setting of privileges in CPF has the following characteristics:

1. Fine-grain partitioning of system privileges. For example, the original Linux capability CAP_SYS_ADMIN is divided into many CPF privileges.
2. The setting of CPF privileges is backward compatible with the setting of Linux capabilities. Every Linux capability has one or more corresponding CPF privileges. operations.
3. Privilege parameters are applied to 20 CPF privileges. The relation between privileges and parameters is explicit and can be configured for specific privileged processes.
4. Invoking a security-sensitive system call is regarded as a privileged operation. During this operation, some system processes must interact with some remote clients continuously, e.g. http and ftp servers. To these processes, invoking some "normal" system calls may incur security problems. For example, the execution of an interactive shell may provide a full control environment over the system for an attacker. In CPF, a specific system call can be identified as a security-sensitive system call for a process individually. As a result, only when the process holds the corresponding CPF privilege can it invoke an identified system call. So far, this kind of system calls includes *execve*, *kill*, *open*, *set*id*, *setgroups*, etc.

### 3.2 Privilege States

Among traditional Linux process attributes, the user and group IDs of a process are used to identify the current user and group of the process. To the Linux capability mechanism, the effective UID (*euid*) and filesystem UID (*fsuid*) of a process are the foundation of privilege control. Numerous important applications, such as named and sendmail, rely on both the capability logic and the ID attributes of a process. Additionally, real and saved UIDs/GIDs are also significant to Linux privilege control [6]. Thus, using other attributes to identify PPS in CPF is inappropriate.

In CPF, all user and group IDs (*uid*, *euid*, *suid*, *fsuid*, *gid*, *egid*, *sgid*, and *fsgid*) of a process are used to identify a PPS. Each PPS is also given a unique state number. The user and group ID attributes together with the state number distinctly determine a PPS. Obviously, it is possible for several different PPSs to have identical user and group ID attributes.

A set composed of the state numbers of all valid target states are assigned to every PPS. When a process is about to transfer from one PPS to another, the target PPS is to be determined not only by the new ID attributes of the process but also by the set related to valid target states. The state number of the new PPS must be included in the set.

Activities of invoking the *execve* or *set*id* (*setuid*, *setreuid*, *setresuid*, *setfsuid*, *setgid*, *setregid*, *setresgid*, and *setfsgid*) system calls make up the PSTEs in CPF. Invoking one of these system calls may lead to a privilege state transfer in Linux.

### 3.3 Program Privilege Table

The CPF system uses a binary format privileged-program configuration file, i.e. "/etc/cpf/.program", to store information concerning privileges and permitted parameters for all possible privilege states of all privileged programs. When loading the CPF module, the system reads the configuration file into the kernel space to construct a hash table indexed by the inode numbers of privileged-program files. This hash table is the system PPT.

### 3.4 System Architecture

The system architecture of CPF is sketched in Figure 3. As shown in Fig. 3, CPF's main functionality includes policy parsing, policy loading, privilege computing and privilege checking.

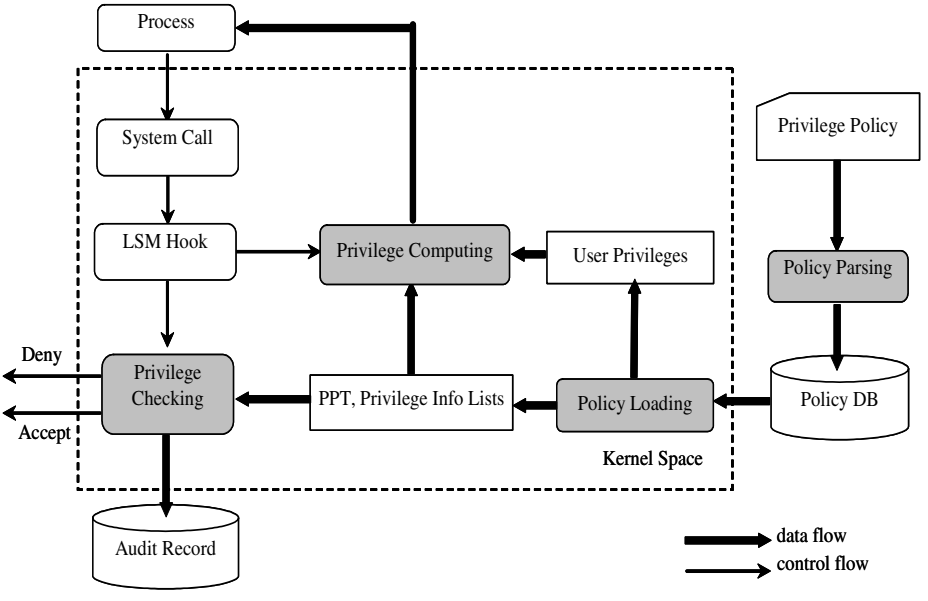



**Fig. 3.** CPF architecture

**Policy Parsing.** A CPF policy defines the rules to control privileged activities of processes. It carries privilege-relevant information about privileged programs and users. Security policies are application oriented. There should be an easy way for users to define their policies after a system is deployed. CPF supports plain-text policy definition. A policy specification language, called Privilege Specification Language (PSL), is designed for users to specify CPF policies. Policies written in PSL are stored in several plain-text format policy configuration files. A policy parser is implemented to translate these PSL files into binary format CPF policy database files, e.g. "/etc/cpf/.program", etc.

**Policy Loading.** A policy loader is responsible for loading CPF policy database files into the pertinent hash tables in the kernel space. When the CPF module is being loaded, the policy loader will carry out its job. Four hash tables are used to store user privileges, privilege-relevant information of privileged programs (PPT), the list of security policy configuration files, and the list of system files respectively.

**Privilege Computing.** When one of the *execve* or *set*id* system calls is invoked, CPF will re-compute the privileges of the current process. A LSM hook, *security_bprm_compute_creds*, is inserted into the code of the *execve* system call. When the current process executes a program, CPF intercepts the *execve* system call and computes the privileges for the new process. Invoking *set*id* may lead to a process privilege state transfer. Similarly, after *set*id* is invoked, CPF will re-compute privileges via corresponding LSM hooks, e.g. *security_task_post_setuid*, etc.

During the computing, CPF tries to look up the new PPS in the corresponding entries of the PPT. If there is a match, the intersection of the privileges in the matched entry of the PPT and the privileges of the process' current effective user is assigned to the process. Additionally, a pointer is maintained for retrieving the permitted parameters of privileged operations. If there is no match, the privileges of the process will be set to null.

To bind security attributes to kernel objects, LSM provides for opaque *security* fields that are attached to various internal kernel objects. Privileges of a process and other privilege-related information are saved in the *security* field added into process kernel data structure (*task_struct*) by LSM.

**Privilege Checking.** Before a privileged operation can be performed, the corresponding system call is intercepted via a LSM hook for privilege check. Only with the corresponding CPF privilege, can the current process perform a privileged operation. If the privilege is of parameter characteristics, CPF additionally need to check whether operation parameters are in the valid domain or not.

CPF privilege check is involved in many LSM hooks. For example, before a process actually writes a security policy file, the *security_file_permission* hook is used to validate the privileges of the process. When a privilege being checked is a refined Linux capability, the system needs to know which corresponding CPF privilege should be examined. Sometimes, interface arguments of the corresponding LSM hook, i.e. *security_capable*, can't provide enough information. On this occasion, according to the arrangement of the system stack, CPF directly reads register data from the system stack to get necessary information, such as the number of system call, arguments, etc.

Privilege checking is one of the main sources from which audit messages come. The results of privilege check, i.e. whether the requests for privileged operations are permitted or denied, are recorded into the audit trails.

### 3.5 Performance

Because the proportion that privilege operates in all operation of the system is very small, if the performance test aims at the whole system or a specific

application, performance influence that CPF produces will be difficult to observe. Consequently, the CPF performance test focuses on testing the kernel time overhead of typical related system calls.

Table 1 reports the average execution time on a same hardware platform (a 1.7 GHz Intel Pentium 4 with 256 Mb of RAM) for three typical system calls: gethostname, sethostname, and open. They correspond to non-privileged operations, normal privileged operations, and privileged operations with parameter characteristic respectively. By reading the real-time clock register of the microprocessor before and after execution of the system call, the CPU time of million times execution is measured.

**Table 1.** Average CPU time of million times execution for 3 typical system calls (in seconds)

| System call | Standard kernel | Loaded CPF | CPF overhead (%) |
|---|---|---|---|
| *gethostname* | 1.370 | 1.373 | 0.22% |
| *sethostname* | 1.098 | 1.365 | 24.32% |
| *open* | 6.375 | 15.135 | 137.41% |

As shown in Table 1, the CPF overhead of non-privileged operations (*gethostname*) is very small (0.22%), can be nearly ignored. A system call with very simple logic, *sethostname*, is selected as an extreme test case to improve the observability of the CPF overhead of privileged operations. Even so, the overhead is still only 24.32%. Because more time may be needed to check privileged parameters, the overhead of privileged operations with parameter characteristic is comparatively great. When open system call is controlled as a security-sensitive call and be performed privilege parameters checking, the overhead archives 137.41%. However, because of the proportion of privileged operations with parameter characteristic is so small, this overhead is still acceptable. In summary, CPF has a little effect on whole performance of the system.

### 3.6 An Experiment on the Prototype System

Wu-ftpd is very popular among FTP server software. There have been a number of attack cases against wu-ftpd, e.g. [7]. These attacks can lead to very serious security problems. For that reason, an experiment with the wu-ftpd daemon process running on the Linux operating system platform is made to illustrate how to use CPF to enforce sophisticated check to defend malicious attacks.

As Figure 1 depicts and Section 2.2 describes, the lifetime of the wu-ftpd daemon process is divided into four PPSs. According to demands on special permissions necessary for the wu-ftpd daemon process to carry out its work, appropriate CPF privileges are assigned to each PPS respectively. See Section 2.2 for details.

Based on the above setting, privileges of the wu-ftpd daemon are restricted within reasonable limits in finer granularity dynamically. A Denial of Service

(DoS) attack test, which utilizes an off-by-one bug in wu-ftpd [7], is conducted to show the functionality of CPF.

When the wu-ftpd daemon is running on a traditional Linux system of kernel 2.6.0 without CPF, a remote attacker can break into the system, filches all root privileges and invokes the reboot system call to shutdown the system successfully. On the same system but with CPF running, the remote attacker can't get necessary privileges to set UID attributes arbitrarily and can't shutdown system. The attack fails.

These experiment results show that the threats of intrusion are greatly reduced thanks to the functionality of CPF.

## 4 Related Work

Trusted XENIX [8] is an important product in the field of secure operating system. In Trusted XENIX, two types of privilege mechanisms are provided: Special user and group identity mechanism and Generalized Privilege Mechanism (GPM). Trusted XENIX privilege mechanism is a comparatively excellent one. Privileged processes can be restricted to performing specific privileged operations based on their application logics. However, Trusted XENIX privilege mechanism is dependent on DAC partially. User can affect the configuration of DAC. This may result in illegal privileged operations. To existing privileged programs, the modification of their source code is necessary to acquire and drop GPM privileges dynamically.

The Linux Intrusion Detection System (LIDS) [9] aims to extend the concept of capability presented in the basic Linux system by defining fine-grain file access capabilities for each process. A relation between process privileges and program logics is constructed indirectly by disabling some important capabilities globally and enable them for specific programs exceptively. This idea is very artful and simple. It also has comparatively high usability. However, LIDS can't support the principle of least privilege thoroughly and can only be as an expedient method.

REMUS [5] is a sandboxing system. Its system architecture is similar to ours, the main differences are that CPF can provide sub-process confinement based on privilege state and mainly focuses on privilege control. The method of system call security analyses in REMUS can be used for reference for CPF.

BlueBox [10] is a policy-driven, host-based intrusion detection system developed by IBM Watson research lab. In BlueBox, process capabilities are specified as a set of rules (policies) for regulating access to system resources on a per executable basis. The language for expressing the rules is intuitive and sufficiently expressive to effectively capture security boundaries. In terms of BlueBox, CPF is also a policy-driven access control system and PSL can be regarded as a policy language too. Moreover, BlueBox incorporates process state into rules to protect process against a much larger number of potential attacks. This is also similar to the concept of privileged state of CPF.

## 5 Conclusion

This paper proposes a new privilege control model called SBPC and implements a prototype system for SBPC called CPF on the Linux operating system platform. SBPC implements fine-grain privilege control with the concept of privilege state. It decomposes the lifetime of a process into privilege states according to activities of the process and its need for privileges. The privilege state is closely related to the application logic of a process. CPF is developed as a LKM based on the LSM framework. It is backward compatible with the Linux capability mechanism and transparent to traditional applications. A DoS attack experiment with the wu-ftpd daemon is conducted to test the functionality of CPF. Experiment results show that fine-grain and automatic privilege control can be exercised transparently to traditional application, threats of intrusion to a system can be reduced greatly, and support to the principle of least privilege can therefore be achieved effectively in a Linux system with CPF.

Partitioning privilege states is the key point to CPF. The practical identifiers of a PPS may vary on different operating systems. Generalized privilege state identification mechanism will be the next topic of our research.

## References

1. Saltzer J, Schroeder M. The Protection of Information in Computer Systems. In Proceedings of the IEEE 63(9), Sept. 1975, 1278–1308
2. Sendmail Inc. Sendmail Workaround for Linux Capabilities Bug. http://www.sendmail.org/sendmail.8.10.1.LINUX-SECURITY.txt
3. Bell D, LaPadual L J. Secure Computer System: Unified Exposition and MULTICS Interpretation. MTR-2997 Rev.1, The MITRE Corporation, Bedford, MA, USA, Mar 1976.
4. Wright C, Cowan C, Morris J, Smalley S, Kroah-Hartman G. Linux Security Modules: General Security Support for the Linux Kernel. Usenix Security Symp., Usenix Assoc., 2002, 17–31
5. Benaschi M, Gabrielli E, Mancini LV. REMUS: A security-enhanced operating system. ACM Transaction on information and System Security, 2002, 5(1): 36–61
6. Chen H, Wagner D, Dean D. Setuid Demystified. In Proceedings of the 11th USENIX Security Symposium, San Francisco, CA, 2002.
7. Purczynski W, Niewiadomski J. Wu-ftpd remote vulnerability. July 2003. http://www.isec.pl/vulnerabilities/isec-0011-wu-ftpd.txt
8. National Computer Security Center. Final Evaluation Report TIS Trusted XENIX version 4.0. 1994.
9. Xie Huagang, The Linux Intrusion Detection Project. http://www.lids.org.
10. Chari SN, Cheng PC. BlueBox: A Policy-Driven, Host-Based Intrusion Detection System. ACM Transaction on information and System Security, 2003, 6(2): 173–200

# Security On-demand Architecture with Multiple Modules Support*

Yanjun Wu[1,2], Wenchang Shi, Hongliang Liang, Qinghua Shang, Chunyang Yuan, and Bin Liang

[1] Institute of Software, Chinese Academy of Sciences, No.1#712, POBox.1871, Beijing, PR. China
yanjun03@ios.cn,
[2] Graduate School of Chinese Academy of Sciences, Beijing PR. China

**Abstract.** It's very important for a general-purpose operating system to have a security-tunable feature to meet different security requirements. This can be achieved by supporting diverse security modules, invoking them on demand. However, the security architectures of existing projects on Linux kernels do not support this feature or have some drawbacks in their supporting. Thus we introduce a layered architecture which consists of original kernel layer, module coordination layer and module decision layer. The architecture supports multiple modules register, resolves policy-conflicts of modules by changing their invoking order, and allow user to customize the security by enabling or disabling modules during runtime. The detailed structure and implementation in Linux based system, SECIMOS is described. The caching issue and performance are also discussed. Our practice showed the architecture helps us achieve flexible adaptation in different environments.

## 1 Introduction

Computing systems are becoming more complex, which makes the security requirements more complicated. For example, with the recent fast development of mobile technology, the computation environments are also becoming *uncertain*. This uncertainty means a host can be put into different LANs, collaborate with different hosts, or connect Internet in different ways. Thus different security policies are expected to efficiently meet the different requirements in the various environments. As the result, a security-critical system should have not only the complete security policies, but also the ability to change these policies as the environment changes, such as choosing one policy instead of another, or tuning the whole system from strict security to flexible availability.

* Supported by the National Natural Science Foundation of China under Grant No.60373054 and No.60073022; the National 863 High-tech Program of China under Grant No.2002AA141080; Graduate Student Innovation Grant of Chinese Academy of Sciences.

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 121–131, 2005.

In the literature of OS security, many approaches have been proposed to achieve the goal, and most of them focus on how to specify diverse security policies and enforce them dynamically. In fact the security architecture is also important since it's the infrastructure to abstract the policy elements, and enforce the security mechanism. To introduce the relationship between architecture and security, we first clarify three concepts that will be used in the paper: policy, model and module.

- Policy is a set of requirements to specify rules or regulations the system needs. It might be ambiguous since it can be expressed either formally or informally.
- Model is used to formally describe a particular security policy or a set of security policies. It focuses on characteristics of policies by abstracting away details, and then states explicitly in a formal language with proofs to ensure correctness and consistency.
- Module is an aggregation of routines or algorithms that implement specific security requirements or regulations. A security module could be developed independently from others security modules and its platform. Given definite interfaces, it should be pluggable to the supporting kernel and provide the kernel with certain security functions or mechanisms.

Briefly speaking, security model is the formal expression of security policy, and security module is implementation of a security model or a set of security policies. The relationship can be illustrated as Figure1.

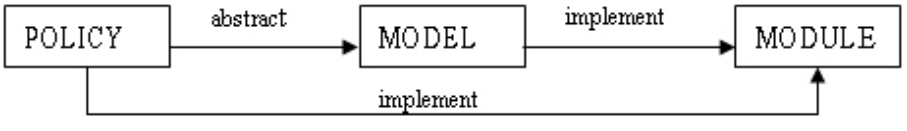



**Fig. 1.** Relationships between policy, model and module

So if the OS architecture supports multiple security modules, we can easily achieve diverse security policies support. However, supporting multiple security modules in existing mainstream OS is difficult. First we cannot change the original kernel too heavily in order to keep it mainstream and general purposed. Second, we must ensure different modules can work together consistently. Furthermore, we should provide a mechanism to enable or disable a certain module as we desire. Finally the performance penalty should not be too much.

The following discussion of paper is based on Linux, which is an open source and mainstream operating system. The rest of the paper is organized as follows. Section 2 describes the related works. Section 3 introduces the SECIMOS architecture and its detailed components. Section 4 discusses caching mechanism, and presents performance result. A brief conclusion and on going work is summarized in Section 5.

## 2 Related Works

This section gives a brief view of popular projects and analyzes their merits and drawbacks in security architecture.

### 2.1 LSM

Linux Security Modules (LSM) [1] is a lightweight, general-purpose access control framework for the Linux kernel that enables many different access control mechanisms to be implemented as modules. It adds a field for every security-critical kernel data structure to store security-related information, and provides low-level hooks to mediate access to kernel objects.

LSM itself is not a security module and does not provide any security mechanism, but it provides a strong foundation for Linux kernel to support various security mechanisms. Now LSM has been merged into 2.6 kernel version. Many projects that used to patch kernel now migrate to LSM, such as Capability, DTE, LIDS, and SELinux we will mention later.

### 2.2 RSBAC

RSBAC (Rule Set Based Access Control) [2] uses GFAC (Generalized Framework for Access Control) [3] under Linux. Its architecture can be divided into AEF (Access Enforcement Facility), ADF (Access Decision Facility), and ACI (Access Control Information). The ADF part provides a module registration facility to support multiple access control models which can be added and removed at runtime in form of loadable kernel module. A subject initializes an access request for an object through a system call, AEF collects the ACI of subject and object, and then forwards the request to ADF, ADF makes the decision according to ACI and the rules of each model one by one. If at least one model returns no grant decision, the request is denied. When AEF gets a grant decision from ADF, and the system call is successfully completed, AEF will notify ADF to update inner states of ADF.

However, RSBAC's architecture has several drawbacks. First, its AEF is embedded and hard-coded in the system call, thus it strongly relies on the kernel source and has to update its patches when kernel is upgraded; once RSBAC has been compiled into the kernel, the execution of AEF cannot been bypassed even there is no model in ADF (i.e. there is waste of executions when user disable all modules). Second, the invoking order of models in ADF is fixed, eg. MAC is always before ACL and CAP. But actually in some specific system calls such as setuid(), it's better to call CAP first, since most of requests will be filtered earlier. Third, the final decision is simply the "AND" result of all models' decisions. If one model returns NOT_GRANTED, the request is denied. This may not be the result that user wants in some privileged operations, eg. reboot or shutdown. Finally, the rights revocation and caching mechanism are not considered in RSBAC architecture.

### 2.3 SELinux

SELinux(Security Enhanced Linux) [4] is based on Flask[5] architecture. The Flask architecture also separates enforcement from policy decision, but it employs unified SID (Security Identifier) to represent both subjects and objects and uses client/server concept like a microkernel; the policy decision part is Security Server (SS), while the clients are composed of Object Manager (OM) and processes who send requests. Before the request is sent to SS, OM maps the subject and object to their SIDs by which SS can make decision.

SELinux implements TE (Type Enforcement), RBAC (Role-Based Access Control) and experimental MLS (Multi-Level Security) in SS. But they are mixed together, and can not be separated clearly, since the aim of SELinux is to enforce wide range of policies by the combination of TE and RBAC. However for some models such as information flow, it may be inadequate at least difficult to directly describe and enforce the policy with TE and RBAC.

By far LSM version of SELinux can "stack" simple modules such as Capability and OWL (Open Wall Linux security module) [6]. However the "stack" mechanism provided by LSM requires one module to be primary module and the primary module must clearly know the secondary modules it wants to stack, and explicitly calls the secondary module when needed. Thus the primary module has full power to decide other modules' fate.

There are many other projects related to Linux kernel security architecture, especially system call interposition based architecture such as REMUS[7], Janus[8]. But system call interception is not race-free, may require code duplication, and may not adequately express the full context needed to make security policy decisions, as pointed out by [1].

By observing the above projects, we believe a better architecture should have the following features:

- Modularity. System can register or unregister security modules to support different security mechanism. These modules should have minimal dependency on kernel version development.
- Harmonization. There should be a coordinator to handle policy conflicts of modules, and make the desired decision.
- Tunability. System can be tuned by enable or disable a module. That means user can *customize* or *tailor* the security according to their needs, such as disable some modules when performance is more crucial than security.

## 3 SECIMOS Architecture

This section will describe our SECIMOS (SECurity In Mind OS) architecture and its detailed structure. SECIMOS is based on Linux 2.6.4 kernel, and aims to provide flexible and dynamic support of diverse security policies. The overview of SECIMOS architecture is as figure 2. It consists of three parts which we called layers here: Original Kernel Layer (OKL), Module Coordination Layer (MCL), and Module Decision Layer (MDL).

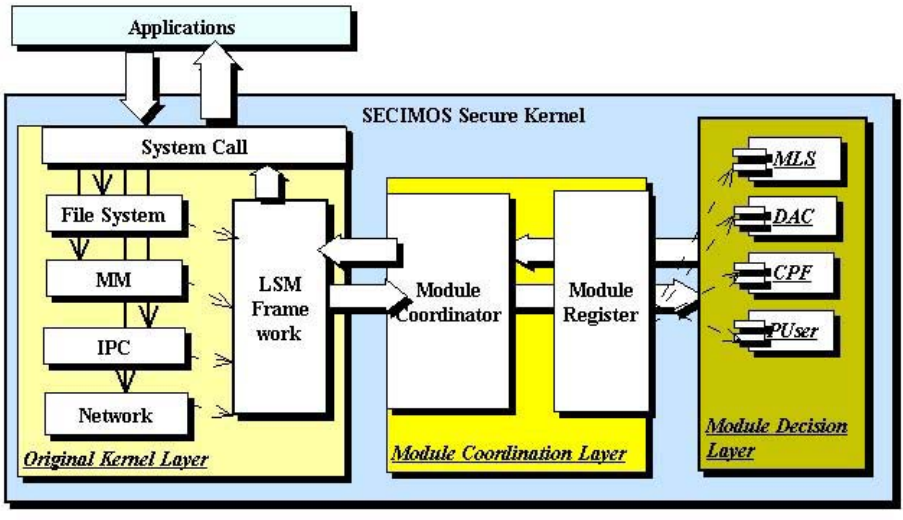



**Fig. 2.** SECIMOS architecture overview

### 3.1 Original Kernel Layer

Original Kernel Layer is the collection of kernel components with a security enforcement interface. As a general-purposed and mainstream operating system, original Linux kernel has the following characteristics:

- Complex inner structures. As a monotonic kernel, it contains file system, process schedule, memory management, IPC, network stack, etc., with cross-references between each other.
- Simple access control interface. System call is the only interface for the interactions between user applications and kernel. But many different system calls finally invoke the same kernel procedure to access the kernel objects. So although there are nearly 300 system calls provided by Linux kernel, the complexity of interface needed for access control will not be that much.

Here we adopt LSM as the security enforcement interface for its generality and being accepted by official kernel. LSM helps us collect the checkpoints of access control and hide the complexity of kernel internals.

### 3.2 Module Coordination Layer

Module Coordination Layer is the key component of SECIMOS architecture. From original kernel layer it is a normal module registered to LSM framework, while from module decision layer it provides an interface for different modules to register. There are two major components in this layer, Module Coordinator and Module Register, as illustrated by figure 2.

**Module Coordinator.** A computer system may need different security policies in different environments. If all the security polices are implemented by several modules, and polices that one module presents conflict with policies the other module presents, system must decide which module should be prior. For example, Tom makes his own file /home/tom/topsecret not readable for others by setting its DAC mask as "rw− − − − − − −". But if the system also has a privilege user, such as a backup manager who needs to read all the files, then the DAC mechanism conflicts with the Privileged User mechanism. If we simply deny the backup managers reading request, the file /home/tom/topsecret will not be backuped. This may not be what the system wants. But if we allow the request permanently, it is danger for a confidential or privacy purposed system. This is what Module Coordinator tries to resolve.

Let us generalize the problem and consider the simplest situation. Supposing there are only two modules in system, the conflicting conditions between them can be summarized as following four cases, as illustrated by table 1.

**Table 1.** four cases of two modules confliction

| Case No. | Module 1 Decision | Module 2 Decision | Possible Final Decision |
|---|---|---|---|
| 1 | Allow | Deny | Allow |
| 2 | Deny | Allow | Allow |
| 3 | Deny | Allow | Deny |
| 4 | Allow | Deny | Deny |

In Table 1, each possible final decision may be what we want in a certain situation. If the result can be simply customized according to users needs, it would be very meaningful for a security system to achieve necessary availability. Module Coordinator does this through two steps.

First we identify each module as one of the four types: Null, Grant, Constraint, Grant-Constraint.

- Null module. The Null module means the module does not make decision, such as audit or a keeping track module.
- Grant module. The Grant module grants the access right even it is not allowed by the modules invoked later. If it does not grant the right, it simply leave the request to the consequent modules.
- Constraint module. Most of access control modules are Constraint modules. If the Constraint module denies a request, the denial result will be immediately returned to the enforcement part. But if the module allows the request, it leaves the finial result to the consequent modules.
- Grant-Constraint module. This module has the full power to decide the final decision. Whether it allow or deny a request, the decision will be immediately returned as the final decision.

The invoking order is also important. Supposing in Table 1, Module 1 is a Grant module and Module 2 is a Constraint module, different invoking order will result in different decision. The comparison is illustrated by Table 2.

**Table 2.** Different results for different invoking orders (Module 1 is a Grant module, Module 2 is a Constraint module.)

Invoke Module 1 before Module 2

| Module 1 | Module 2 | Final Decision |
|---|---|---|
| Allow | Deny | Allow |
| Deny | Allow | Allow |

Invoke Module 2 before Module 1

| Module 2 | Module 1 | Final Decision |
|---|---|---|
| Allow | Deny | Allow |
| Deny | Allow | Deny |

So secondly, we assign each module a value that indicates its invoking order. The module with small order value will be invoked before the module with larger one.

Now lets see how we solve the problem described at the beginning of this subsection. First we identify the DAC module as a Constraint module, and Privilege User module as a Grant module. If we prefer DAC module in a normal time, we assign a smaller order to DAC module and invoke it before Privilege User module. If the backup time comes, we assign Privilege User module a smaller order value than that of DAC to grant backup right. When the backup manager finishes his job, the order value could be adjusted again. This is can also be described by Table 2 if we assume Module 1 as Privilege User module and Module 2 as DAC module.

As figure 3 shows, the Module Coordinator Component maintain a list that chains all the modules according to their invoking order. In data structure of node, order is invoking order of module, type is what type the module is, Null, Grant, Constraint or Grant-Constraint. enable field is used to indicate whether the module is currently enabled or disabled.

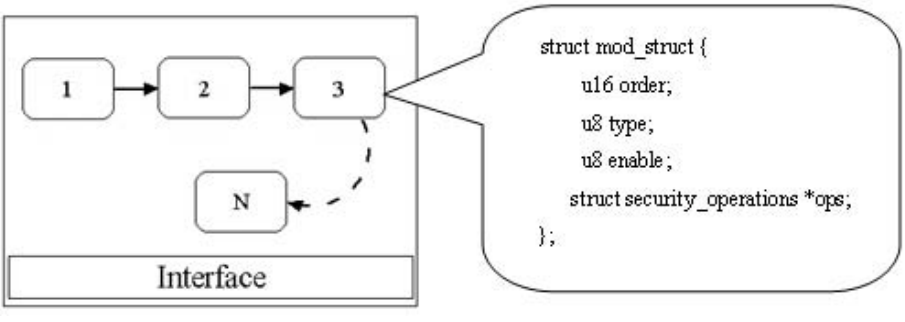



**Fig. 3.** Module Coordinator and data structure of module node

Module Coordinator also provides an interface for security administrator to tune the registered modules. The tuning operation includes the following:

- change or list invoking order of a module,
- change or list the type of a module,

– runtime enable ordisable a module,
– list the current status (enabled or disabled) of a module.

**Module Register.** As another component in Module Coordination Layer, Module Register provides a register interface for various modules. The register interface is like LSM framework, except it requires a module to provide three additional parameters needed by Module Coordinator component: invoking order, module type and default status (enabled or disabled). This is simply implemented by the kernel function *module_param*.

If a module provides an order value that already owned by another registered module, the Module Register will refuse the registration until a proper, unused order value is provided. When a module is registered, a new *mod_struct* node is allocated and inserted into the proper position of module list.

### 3.3 Module Decision Layer

This layer is collection of different modules. Currently SECIMOS implements four modules: MLS, DAC, PUser, CPF. We will give brief introduction of them.

MLS is a module that implements BLP[9] model based Multi-Level Security. It is used for strong confidential security policy and supports 256 sensitivity level and 64 categories.

DAC module is different from traditional DAC mechanism of UNIX system. It extends the access rights from rwx to 12 kinds. The work is based on the EA/ACL project [10], but we implement it as a loadable module.

PUser stands for Privilege User, which can be regarded as a simplified RBAC[11]. There are 10 privilege users, such as security administrator, system operator, audit administrator, backup manager, etc. We implement it as a Grant module, so that a normal user can be assigned a privilege role and perform some special task.

CPF is our extension for current capabilities of Linux kernel. Since some capabilities the kernel provides are too coarse, for example CAP_NET_ADMIN, CAP_SYS_ADMIN. CPF divides them into several capabilities. This is similar as BlueBox [12], but we provide a more easy way of configuration.

## 4 Discussions

### 4.1 Tunability

In the real world, different users may have different security requirements in different time or different situations. A PC user without network connection may need little security mechanisms, while a military system requires strict hierarchy policy(eg. MLS) to prevent information leakage.

One of SECIMOS architecture's goal is to provide system with a tunability feature, by which user can tune the whole system from strict security to high flexibility, and vice versa. This is achieved by enabling or disabling certain modules through Module Coordinator Interface.

SELinux also provide tunable features by changing and reloading security policies. But user must have detailed knowledge about how to write a correct policy. Although SECIMOS achieves it in a coarse-grained way, it hides the complexity of security policies and alow user to manipulate the system at the module level.

### 4.2 Caching Mechanism

Performance is also a pain in many security systems besides usability. One solution is due to the developers if they could optimize their code and minimize the overhead of security mechanism. This is beyond the scope of our paper. We focus on the other popular way which introduces the caching mechanism to cache the decision result and reduce the number of computing decision.

The AVC (Access Vector Cache) used by SELinux is a famous and successful example of employing caching mechanism. A record of AVC is a (SSID, OSID, Perms) triple, which stores what permissions a subject with SSID has on the object with OSID. When a request comes, the enforcing part of Flask will first look up AVC. If the cache hits, then it gives the decision. Otherwise, the Security Server is consulted, and AVC is updated.

The paper [5] does not give the detailed comparison of how much performance the AVC achieved. But we believe that the caching mechanism is not always good choice for all the security modules, especially for simple modules. If the time spent on looking up cache table is longer than directly computing the decision, then the cache is not a better choice. Particularly, cache mechanism introduces complexity into security architecture. System needs a notification mechanism to invalidate the cache record when security policy changes, thus makes the right revocation more complicated.

In SECIMOS no caching mechanism is used. One reason is till now security policies each module enforces are not complex. But another reason is more important. Since SECIMOS architecture aims to support multiple modules, and the modules are independent from each other. There is no uniform representation for these different modules, thus its hard to predict cache format that a certain module needs. But the module itself can implement cache mechanism internally if it needs.

### 4.3 Performance

Table 3 shows the lmbench [13] result of SECIMOS modules under Linux 2.6.4 kernel, Intel Pentium 2.4G Hz, 512M DDR. The second row with header Normal 2.6.4 Kernel is the result of official Linux 2.6.4 kernel. The third row with header SECIMOS_ALL is the result of SECIMOS with all modules enabled. The rest rows are the result with individual module enabled while others disabled.

From Table 3 we can see the whole SECIMOS increases more than 200% performance penalty to the normal kernel when performing stat and open+close system call. This is partly because MLS, DAC, PUser use extended attribute (EA) to store the security properties of objects, which introduces additional

**Table 3.** Performance result of whole SECIMOS and individual module

| | Null call | null I/O | Stat | open close | select TCP | sig inst | sig hndl | fork proc | exec proc | Sh proc |
|---|---|---|---|---|---|---|---|---|---|---|
| Normal 2.6.4 Kernel | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| SECIMOS ALL | 1 | 1.25 | 3.148515 | 2.76 | 1.083 | 1 | 1.04 | 1.02 | 1.2 | 1.16 |
| MLS only | 1 | 1.13 | 1.331683 | 1.28 | 1.078 | 1 | 1.04 | 1.02 | 1.1 | 1.02 |
| DAC only | 1 | 1.08 | 1.836634 | 1.03 | 1.017 | 1 | 1.01 | 1.01 | 1 | 1.01 |
| CPF only | 1 | 1.21 | 1.079208 | 1.1 | 1.006 | 1 | 1.01 | 1.02 | 1 | 1.01 |
| PUser only | 1 | 1.08 | 2.50495 | 2.53 | 1.007 | 1 | 1.01 | 1 | 1 | 1.07 |

slow I/O operations. But it should not be that significant, the deeper reason remains as our future work. Other performance effects are relatively smaller.

## 5 Summary

From years of security practices and experiences, we recognize the usability of a policy-fixed security system is very limited. Customs usually have different requirements in different time or different situation. So the system should provide different security mechanisms to meet these needs, with minor change or a few tuning actions. But till now, we did not find a proper architecture that meets the goal. SECIMOS architecture is our first attempt to build an environment and requirement adaptive security system. It reached the goal by providing the following features:

- Dynamically register modules that can present different security policies
- Runtime enable and disable a module
- Runtime change a modules type and invoking order to resolve conflicts
- Achieve the above without patching kernel

There is still much work to do on SECIMOS architecture. First, some complex security module needs to keep the track of system state. If the module is disabled and enabled occasionally during run time, the state it keeps may be inconsistent and no longer correct. SECIMOS provided a method for a module to keep state but not make decision when being disabled, but this method is not mature yet. Second, the performance penalty is still heavy when all the modules are enabled. The reason needs to be uncovered and the proper solution should be provided.

## References

1. C. Wright, C. Cowan, J. Morris, S. Smalley, and G. Kroah-Hartman. Linux security modules: General security support for the linux kernel. In Proceedings of the 11th USENIX Security Symposium
2. Rule Set Based Access Control. http://www.rsbac.org

3. M.D.Abrams, L.J.LaPadula, K.W.Eggers, et al. Generalized Framework for Access Control: An Informal Description. Proceedings of the 13th National Computer Security Conference, Oct 1990, pp.135-143.
4. NSA Security Enhanced Linux. http://www.nsa.gov/selinux
5. R.Spencer, S.Smalley, P.Loscocco, et al. The Flask Security Architecture: System Support for Diverse Security Policies. Proceedings of the 8th USENIX Security Symposium, Washington, DC, USA, Aug 1999, pp.123-139.
6. Openwall Project. http://www.openwall.com
7. MASSIMO BERNASCHI, et al. REMUS: A Security-Enhanced Operating System.
8. T.Garfinkel and D. Wagner. Janus: A practical tool for application sandboxing.
9. Bell D, LaPadual L J. Secure Computer System: Unified Exposition and MULTICS Interpretation. MTR-2997 Rev.1, The MITRE Corporation, Bedford, MA, USA, Mar 1976.
10. Linux EA/ACL project. http://acl.bestbits.at
11. Sandhu R, Coyne E J, Feinstein H L,Youman C E. Role-based access control model. IEEE Computer, 1996, 29(2): 38 47
12. SURESH N. CHARI and PAU-CHEN CHENG. BlueBoX: A Policy-Driven, Host-Based Intrusion Detection System.
13. Larry McVoy and Carl Staelin. lmbench: Portable tools for performance analysis.InProc. Winter 1996 USENIX, San Diego, CA, pp.279-284. January 1996.

# Measuring Resistance to Social Engineering

Hågen Hasle, Yngve Kristiansen, Ketil Kintel, and Einar Snekkenes

Gjøvik University College,
Box 191, 2802 Gjøvik, Norway
einar.snekkenes@hig.no
www.nislab.no

**Abstract.** Social engineering (SE) is the name used for a bag of tricks used by adversaries to manipulate victims to make them say or do something they otherwise wouldn't have. Typically this includes making the victims disclose passwords, or give the adversary illegitimate access to buildings or privileged information. The book *Art of Deception: Controlling the Human Element of Security* by Kevin Mitnick gives several examples of potential attacks. Clearly, countermeasures are needed. Countermeasures may include special hardware, software, improved user interfaces, routines, procedures and staff training. However, in order to assess the effectiveness of these countermeasures, we need a SE resistance metric. This paper defines such a metric. We have also implemented software to obtain metric test data. A real life SE experiment involving 120 participants has been completed. The experiment suggests that SE may indeed represent an Achilles heel.

**Keywords:** Information security, Social engineering, Vulnerability analysis, Security metrics, Security testing.

## 1 Introduction

Social engineering (SE) can be regarded as "people hacking". It's hacker jargon for soliciting unwitting participation (by persuading or deceiving targets to volunteer information or assistance) from a person inside an organization, rather than breaking into the system independently[Vig]. SE was made famous by Kevin Mitnick [MS03].

SE may exploit human traits and weaknesses such as a desire to be perceived as a 'nice' person, greed, laziness etc.

### 1.1 Problem Description

In order to assess the risk associated with SE, we need to measure the resistance to social engineering (SER). Similarly, in order to assess the effectiveness of SE countermeasures, we may want to measure SER both before and after a countermeasure has been implemented. This will give us a better understanding of the relative SER contributions of the countermeasures investigated.

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 132–143, 2005.

### 1.2 Claimed Contributions

We provide the definition of a social engineering resistance (SER) metric, including a discussion on ethics. It is shown that that automated SER data collection is feasible using standard WEB and email technology. The paper also includes SER data from a real life experiment having 120 participants.

## 2 Related Work

The motivation for our interest in SE tests is:

- How can we assess the security robustness of an organization? We are primarily interested in using the results of the tests as a decision criteria when selecting or designing security awareness training programs.
- Given the choice of several SE counter measures (e.g. training programs, procedures, software/hardware support etc.), we need a way of making ranking these options based on their effectiveness.

Consequently, we need a way of assessing the effectiveness of SE countermeasures.

Many papers have been written on the topic of SE techniques and their associated countermeasures [And93, Sch00, HLK01, Gor95, Pou00, Win97]. Barrett [Bar03] claims that NLP (Neuro Linguistic Programming) is an effective approach to SE. Winkler presents a case study of an actual industrial espionage attack against a large US corporation [Win96]. Examples of SE in the context of internet worms can be found in [KE03] [Ber03]. Weaver et al has been developed a taxonomy for internet worms [WPSC03]. The paper identifies SE as one of the key techniques for worm activation. Rubin [Rub01] addresses the issue of SE in a remote electronic voting context.

There seems to be a widely held belief that SE tests are useful. For example:

- R. Henning [Hen99] suggests that SE tests should be carried out on a regular basis, as a way of assessing the performance relative to a security service level agreement.
- Integralis (`www.integralis.com/downloads/quarto/s3report_1.pdf`) claims that this type of 'human' vulnerability testing provides crucial information about security weaknesses and that a few simple tests can provide substantial direction for the developers of training programs.
- Candice Neethling (`www.securityforums.com/forum/viewtopic.php?p=10398`) claims that

  > Social-engineering tests are effective means of determining the current levels of user awareness. Such tests are also a good way to highlight to users the potential pitfalls resulting from a lack of awareness and application of security in everyday operations.

SE tests can be carried out using phone, email, face-to-face conversations, games, web pages etc.. There are many organizations offering SE tests including

- HCS systems: `www.hcssystems.com/consulting/security-review.pdf`
- Integralis: `www.integralis.com/downloads/quarto/s3report_1.pdf`
- AEA technology: `www.aeat.com/consulting/itrmst5.htm`
- Marsys: `www.marsys.com/pdf/MARSYS_Security.pdf`
- Infotex: `www.infotex.com/solutions/penetration_testing.htm`

Rienze [Rie99] describes a SE experiment carried out at John Hopkins University.

Barrett [Bar03] describes SE tests including 'Phone up help desk with cover story' and 'Head hacking'. Head hacking follows the simple three-phase approach. First, identify and locate some potential targets; next, get to know the target and their weaknesses; and finally exploit those weaknesses. However, this method is not described in any detail — that is, it is not at all obvious that the two white hats reading the paper and carrying out an attack according to the method described will obtain the same results. The tests are based on communication by speech.

The most striking difference between the tests carried out by Barrett [Bar03] and us is that Barrett appears to use person-to-person interactions when performing the tests and he has a strong focus on the performance of individuals.

Unfortunately, neither detailed results nor the details of the test method appears to be available from the literature or the WEB.

Our approach is characterized by a high degree of automation — using web and email, giving raise to scalable experiments. We use collective statistics — focusing on the overall performance of the organization rather than on specific individuals.

When designing SE tests we need to consider the following: The outcome of a SE attack depends on

- the adversary's (both white and black hats) ability to carry out the attack.
- the victim's ability to detect the attack.
- other factors outside the control of both adversary and victim.

Techniques to assess the capability of an adversary include standard lie detection procedures and the like. McClure et al. [MAMG02] describes a system for deception detection. This system detects microdynamic cues and may be used to assess (parts of) white hat attack capabilities.

Although a SE test (and attack) makes use of techniques other than lying — lying often plays a key role in the 'attack'. The capability to both detect and construct lies can be assessed using games such as 'Truth or Lie' (iteslj.org/games/9919.html).

Vaughn et al. [VHS] provides an overview of current (as of 2001) work on security metrics. Their paper also offers a metric taxonomy and several metric design guidelines. Shirley Payne [Pay01] describes good security metrics as SMART: "specific, measurable, attainable, repeatable, and time-dependent". She goes on to describe two different approaches to how security metrics can be generated. We have chosen the top-down approach in our own work. Our goal is clear; we want to investigate companies' resilience against SE.

## 3 Defining a Social Engineering Resistance Metric

The goal of our work is to define a practical method for measuring the ability of organizations to protect (information) assets against attacks making use of SE techniques.

Our metric can be described as follows: First we define an experiment for individual users. For each individual participating in the experiment we have two outcomes: disclosure of sensitive data and no disclosure of sensitive data. We then repeat this experiment with several participants. Our metric corresponds to the probability that an adversary selecting one user at random, will fail to obtain secret information. This clearly requires our test population to be a representative subset of the employees.

The rest of this section describes the metric in more details and explains the motivation for the choices made.

### 3.1 Direct or Indirect Metric?

A direct metric makes measurements of the phenomenon of interest. An indirect metric takes measurement data from some phenomenon with a strong correlation to the phenomenon of interest. In practice, there will be degrees of indirectness. Several degrees of directness can be identified:

- Expose the users to a 'representative' SE attack, and record the outcome and possibly the detailed actions of the participants.
- Ask what the users would have done in a hypothetical situation, either through an interview or a questionnaire.

Usually, indirect metrics are more likely to suffer from poor validity than the more direct metrics. As a compromise on practicality we decided to design a metric where we expose the users to a specially designed SE attack.

### 3.2 How Many Categories?

Our goal is to have a general metric; it should not be tied to a specific experiment. SE is a broad field; there are many different ways an adversary can succeed. We want our metric to encompass numerous sorts of SE tricks we had to generalize our measurement unit. We concluded that our measurement unit could only have two categories: disclosure of sensitive data and no disclosure of sensitive data.

### 3.3 Individuals or a Group?

Our approach is to check robustness in breadth. That is, we make sure that our experiment is scalable by having a low 'per user' cost. Given a fixed budget for the SER metric data collection, and having carefully considered ethical issues, we have decided to collect 'shallow' data from each individual. We believe our decisions are supporting the principle of 'Defence in depth without defence in breadth is not effective'.

When measuring a single person we only have two categories. They have a clear order between them, and the scale is an ordinal one. In our metric of an entire organization, the measurement unit we use has a ratio scale with a total order.

### 3.4 The SER Metric

**Precondition.** The test population must not be provided with information that may result in distortion or invalidation of the test data.

**Selection of Test Participants.** The subset to be tested must be a random and representative subset of the population.

**Experiment.** Each participating individual is exposed to an electronic SE attack. The attack must be designed such that for each individual, he either discloses or he doesn't disclose his system login password.

**Data Collection.** Before the experiment starts, we record the following:

- number of individuals selected or invited to take part in the experiment ($N$).
- number of active participants ($A$) . An active participant is defined as a participant that is known to have received the initial stimulus of the experiment. When using email, this corresponds to reading the email.

During the experiment we record usernames and passwords obtained from the users through the SE experiment. We check username/ password pairs against all local username/password databases. We let $P$ denote the total number of valid password/username pairs obtained during the experiment.

**Metric Computation.** When designing the metric we need to consider how decisions of non-active participants can influence the outcome of our experiment. We have several alternatives including.

- All non-active participants (i.e. $N - A$)would have disclosed valid username/ password pairs — the pessimistic assumption, corresponding to $SER_{LOW}$.
- Non-active participants have a password disclosure willingness similar to that of the active participants. This gives a robustness of $(1 - P/A) * 100\%$.
- No non-active participants would have disclosed valid username/password pairs — the optimistic assumption corresponding to $SER_{HIGH}$ below.
- Make some other assumption with respect to the distribution of username/ password disclosure.
- Present the results in a way that highlights the uncertainty associated with non-active participants.

We have decided to do the latter. The idea is to present the robustness data as an interval, where the lower (upper) bound corresponds to the pessimistic (optimistic) assumption.

$$SER = < SER_{LOW}, SER_{HIGH} >$$

For each of the passwords and usernames recorded, check if they correspond to valid password/username pairs on any of the information systems. Count the number of such valid pairs ($P$) where

$$SER_{LOW} = (1 - (P + N - A)/N) * 100\%$$
$$SER_{HIGH} = (1 - P/N) * 100\%$$

Since $P \leq A \leq N$, it can be shown that $SER_{LOW} \leq 1 - P/A \leq SER_{HIGH}$.

An example: Assume we have 100 participants ($N = 100$), where only 50 are known to have received the initial experiment stimulus ($A = 50$), and 10 disclosed a valid username password pair ($P = 10$). Then $SER =< 1 - (10 + 100 - 50)/100, 1 - 10/100 >=< 40\%, 90\% >$. That is, the SER is in the interval between 40% and 90%.

One of the referees noted that the gap between $SER_{LOW}$ and $SER_{HIGH}$ can be very large when $A$ is small. Recall that a small $A$ signifies that the number of confirmed active participants is low. We may approach this issue in several different ways e.g. by making use of additional 'knowledge' such as the assumptions indicated above. In organizations where one has standardized on the use of a single email client/web browser, it would be more easy to collect 'correct' reception confirmation data (i.e. the value of $A$). Also, if the duration of the experiment is sufficiently long, one would expect $A$ to approach $N$, and consequently the difference between $SER_{LOW}$ and $SER_{High}$ would be relatively small.

### 3.5 Evaluating the Metric

Using the taxonomy of [VHS], our metric can be characterized as follows:

- Objective. Provided the attack/rules of engagement are precisely defined.
- Quantitative. Response from each participant is one of 'disclose', 'not disclose' or 'no response'.
- Dynamic. Responses from individuals are expected to change e.g. as the results of security awareness campaigns.
- Absolute. The metric can be interpreted on its 'own'.

## 4 Ethical Considerations of Conducting SER Experiments

SE is about deceiving other people. With respect to ethics of SE tests, there are several issues to consider. For example, `www.isecom.org/osstmm/rules.shtml` gives a set 'rules of engagement'. Barrett[Bar03] suggests that SE tests must be well controlled.

### 4.1 Ethical Standards

The Helsinki declaration[Ass] is a statement of ethical principles to provide guidance to physicians and other participants in medical research involving human subjects. The declaration has been adopted by the World Medical Association (WMA). We believe that this declaration is important although our research is not medical, because our research involves human subjects. The declaration states in 21 that the right of research subjects to safeguard their integrity must always be respected. Every precaution should be taken to respect the privacy of the subject, the confidentiality of the patient's information and to minimize the impact of the study on the subject's physical and mental integrity and on the personality of the subject. Similar demands are listed in a set of guidelines published by the "The National Committee for Research Ethics in the Social Sciences and the Humanities" in Norway [fREitSStH].

### 4.2 An Ethical Dilemma

Use of password, associated technical access control mechanism and routines and procedures are elements employers make use of to protect their assets. In some cases, these mechanisms are used to protect information sensitive both from an ethical and legal point of view. How are we to check that this information is sufficiently well protected and that staff is handling passwords in accordance with specified procedures? Would it be unethical to check if the medical staff is sufficiently robust against unauthorized attempts to obtain sensitive patient information?

Clearly, both legal (e.g. entrapment) and ethical issues should be considered. The answer to the question may depend on how the tests are performed and how the test results are presented and disclosed. Some may argue that it is acceptable to provide 'league tables' of medical institutions with respect to handling of patient records. One may also argue that it would be unreasonable to disclose similar league tables on individual members of medical staff.

### 4.3 Our Solution

Our solution to this ethical dilemma was first of all to seek the consent of the management of the organizations we performed our experiment in. If the employees in an organization is under (ethical and/or legal) obligation to protect information, then it may be argued it would be reasonable to carry out compliance tests.

Secondly, we have tried to minimize the impact our experiments have on the participants. Everyone is anonymous and we informed everyone after the experiment was over. We provided management with information which was suitable for communicating to the experiment participants including the purpose of the experiment, what we did to protect their anonymity and privacy, and whom they could contact if they had any complaints.

Thirdly, every day people face a real threat that someone may try to use similar methods to harm them. One might argue that our experiment is merely

a measurement of how people behave in a real-life situation. The benefits our experiment may have on the participants should not be overlooked either. They have probably learned from this experience, and will be better suited to face the real dangers when they have to.

## 5 Using the SER Metric

We performed our SER experiment in an organization with about 2000 computer users. The experiment was carried out in cooperation with the organization's IT department, and with the explicit authorization of the CEO. The experiments lasted for three days, from Tuesday 02.12 to Thursday 04.12. When the results from the experiments were ready everything was made anonymous.

### 5.1 Selecting the Test Population

We obtained a list of all computer users at the organization were we conducting the experiment at. We removed everyone who could know about our experiment from the list. The employees in the organization naturally belong to two separate subgroups, where the difference is the nature of their work. We randomly selected two groups of 30 users from each user category, a total of 4 group (120 users). The groups where disjoint and selected in such a way that they where representative of the users with respect to both age and gender.

### 5.2 Experiment I: A Survey

The goal was to let users think they participate in a real survey and that they leave their username and password for authentication in case they win first prize.

In order to make the impression of a genuine request, graphical elements had been downloaded from the institutions main page and was included both in the email and the survey site. In addition, we used SSL to protect the information in transit and to make the overall feel more genuine.

Each individual target user was assigned a unique ID. Every request to the pages were tagged with this ID using several methods. First, the ID was embedded into the URL of requests. To track visits outside the experiment, i.e. direct site visits, the ID was also embedded into a Cookie in the user's browser.

Both the mail and the site contained a lot of invisible graphical elements that had no uses other than transferring the ID to and from the site. Elements like these are called web-bugs [Smi99], and are heavily utilized for tracking by the marketing industry, including companies in Norway [Aft][Net]. As such, the technique is not new. By utilizing web bugs [Smi99], we were able to track if and when a user read the e-mail, clicked on the link in the e-mail and entered the web pages outside of the e-mail. We also logged the users' actions and input on the web pages.

The individual emails were sent to sixty participants during one working day.

### 5.3 Experiment II: A Mail with a Login-Box

In this experiment, each user was assigned a unique ID. The experiment was implemented using a web-bugged email with a blank content. In addition to track reading, the web bug also triggered a standard Internet Explorer login box. To better simulate a real life situation, the web bug was provided with some logic that prevented the login box from being displayed more than once for each user.

The individual emails in this experiment were sent to the groups, one group from each of the categories (sixty participants), during one working day.

## 6 Experiment Result

During the three-day period that our experiment lasted, we got a total of fifteen passwords. In summary, we had 120 participants ($N = 120$), 59 active participants ($A = 59$) and we obtained 15 seemingly valid password/username pairs ($P = 15$). This gives SER of $< 37\%, 88\% >$

We registered activity on 59 participants, (1 was registered as visiting the survey without the system first detecting that the email was read, he was not included), 31 for the survey and 28 for the login-box.

**Table 1.** Social engineering experiment data

| Experiment | Passwords obtained | Active participants | Users in group |
|---|---|---|---|
| Category 1 | | | |
| Survey | 5 | 22 | 30 |
| Login Box | 5 | 19 | 30 |
| Category 2 | | | |
| Survey | 3 | 9 | 30 |
| Login Box | 2 | 9 | 30 |
| Sum | 15 | 59 | 120 |

The 7 (5+2) users that entered their password had been nagged by the login box several times before they did so. On average, they pressed the cancel button 4.1 times before entering their password. Only one user entered the password right away. We can also see that the users re-read the empty email several times after entering the password, on average 8.6 times. No such similar trends were detected for the other experiment.

It is interesting to note that the disclosure per active participant is surprisingly consistent for all four groups (24%, 26%, 33%, 22%). The relatively low percentage of active participants in the category 2 groups can partially be explained by a high number of out-of-office activities by these users in this period.

### 6.1 Discussion

Generally, much fewer people read the mail than we had hoped. Although this can partly be explained by the Eudora-error we explain below, other factors must also be taken into consideration. One possible reason was that the experiment was conducted at a time where many of the users were very busy, and they might not have the time to read mail.

Also, as far as we can tell by talking to company officials, no one bothered to inform the IT department or the administration of the mail they received.

There were virtually no differences between the results from the two experiments. Our results indicate that the experiments are closely related. This is good; it gives us a more absolute metric and increases the relevance of each experiment. We also note that it is difficult to conduct technical experiments in an environment that is very heterogeneous. A metric like this gives a better representation of reality if we had had better control of the environment. In a company with a higher standardization on which email client they use, this would probably be possible.

When we added the second experiment with the login-box we did so without any extensive testing. We tested our software using a Microsoft Outlook mail client with a successful result. What we failed to foresee was the multitude of email programs that were used; at least both web-mail, Outlook and Eudora. In Eudora the login-box didn't show, so the experiment failed for those who used Eudora. The survey experiment mail was shown properly, but Eudora stopped the web-bug so we couldn't measure how many who read the mail.

## 7 Suggested Further Work

Our definition of the SER metric is very simple. In our definition of SER, we only considered two outcomes: 'Resist attack' and 'Give inn to attack'. Alternative definitions of SER should be investigated. Since a SE attack typically goes through several stages, it may be of interest to enhance the SER metric to include data on progress relative to these stages.

Test procedure/routines for countermeasure effectiveness.(need $n$ groups, $n >= 2$) We also need to consider effects the experiment execution has on subject resistance if we are to use a before/after approach on the same individuals.

Other issues of interest includes effectiveness results on different countermeasures and a more detailed analysis of SER with respect to validity and reliability.

Interviewing experiment participants and let them explain why they were deceived might give added insight. It would be interesting to see how our experiments correlate with the results from a self-assessment survey. Based on known psychological experiments [AW02] we would expect the correlation to be weak.

## 8 Conclusions

Our experiment shows that it is relatively cheap and easy to mount a large scale SE attack (or experiment) with a high success rate. Consequently it seems fair

to say that SE represents a realistic and serious threat. Our experience from carrying out the experiment and analyzing the experiment data suggests that security metrics work in organizations will benefit strongly from an incident report system.

A heterogeneous environment makes 'reliable' data collection more difficult. From a security point of view, a heterogeneous environment offers both users, the IT department and adversaries additional challenges.

We have defined what we believe is a good and simple metric that scales well and is highly usable and easy to assess.

## References


[Aft] Aftenposten. Dataforeningen raser mot nettovervåking. www.aftenposten.no/nyheter/nett/article.jhtml?articleID=663692.

[And93] Ross Anderson. Why cryptosystems fail. In *Proceedings of the 1st Conference on Computer and Communications Security*, 1993.

[Ass] World Medical Association. World medical association declaration of Helsinki. www.wma.net/e/policy/b3.htm.

[AW02] Martha Augostinos and Ian Walker. *Social cogntition. an integrated Introduction.* SAGE publications Ltd, 6 Bonhill Street, London, Reprinted 2002.

[Bar03] Neil Barrett. Penetration testing and social engineering: hacking the weakest link. *Information Security Technical Report*, 8(4):56–64, 2003.

[Ber03] Hal Berghel. Digital village — malware month. *Communications of the ACM*, 46(12), December 2003.

[fREitSStH] The National Committee for Research Ethics in the Social Sciences and the Humanities. Guidelines for research ethics in the social sciences, law and the humanities. www.etikkom.no/retningslinjer/NESHretningslinjer/NESHretningslinjer/Eng%elsk.

[Gor95] S. Gordon. Social engineering: Techniques and prevention. In *Proceedings of the 12th World Conference on Computer Security, Audit & Control, Westminster, U.K.*, pages 445–451, October 1995.

[Hen99] Ronda R. Henning. Security service level agreements: Quantifiable security for the enterprise? In *Proceedings of the 1999 workshop on New security paradigms Caledon Hills, Ontario, Canada*, pages 54–60, 1999. ISBN: 1-58113-149-6 doi.acm.org/10.1145/335169.335194.

[HLK01] Brian Hatch, James Lee, and George Kurtz. *Hacking Linux exposed: Linux security secrets & solutions.* Osborne/McGraw-Hill, 2001. ISBN: 0-07-212773-2.

[KE03] Darrell Kienzle and Matthew C. Elder. Recent worms: A survey and trends. In *Proceedings of the 2003 ACM workshop on Rapid Malcode. Washington, DC, USA*, pages 1–10, 2003. ISBN: 1-58113-785-0.

[MAMG02] J. McClure, W. I. Ames, T. F. McGraw, and J. L. Gouin. A system and method for enhanced psychophysiological detection of deception. In *Proceedings of the 36th Annual 2002 International Carnahan Conference on Security Technology*, pages 50–59, 2002.

[MS03] K. D. Mitnick and W. L. Simon. *The Art of Deception: Controlling the Human Element of Security*. John Wiley & Sons, 2003.

[Net] Aftenposten Nettutgave. Nettstedet vet at du er der. `www.aftenposten.no/nyheter/iriks/article.jhtml?articleID=662796`.

[Pay01] Shirley C. Payne. A guide to security metrics. `rr.sans.org/audit/metrics.php`, July 2001.

[Pou00] K. Poulsen. Mitnick to lawmakers: People, phones and weakest links, 2000. Available from `www.politechbot.com/p-00969.html`.

[Rie99] Greg Rienzi. All university computer users need to protect passwords. The Gazette Online — The newspaper of the Johns Hopkins University, October 1999. Volume 29 No. 7 `www.jhu.edu/~gazette/1999/oct1199/11warns.html`.

[Rub01] Aviel D. Rubin. Security considerations for remote electronic voting. In *29th Research Conference on Communication, Information and Internet Policy (TPRC2001)*, 2001.

[Sch00] B. Schneier. *Secrets and Lies: Digital Security in a Networked World*. John Wiley & Sons, 2000.

[Smi99] R. M. Smith. The web bug faq. Electronic Frontier Foundation, 1999.

[VHS] Rayford Vaughn, Ronda Henning, and Ambareen Siraj. Information assurance measures and metrics — state of practice and proposed taxonomy. `www.cs.jmu.edu/users/prietorx/HICSS36/Minitrack14/FullPapers/InfoAssura%nceMesureMetricsFinalVaughn.pdf`. A revised version will be presented at the Thirty-Sixth Hawaii International Conference on System Sciences (HICSS-36) to be held January 6–9, 2003.

[Vig] Vigilante.com. Vigilante home. `Vigilante.com`.

[Win96] Ira Winkler. Case study of industrial espionage through social engineering. In *Proceedings of 19th National Information Systems Security Conference*, 1996. `citeseer.ist.psu.edu/320204.html`.

[Win97] I. Winkler. *Corporate Espionage: what it is, why it is happening your company, what you must do about it*. Prima Publishing, CA, 1997. ISBN: 0761508406.

[WPSC03] Nicholas Weaver, Vern Paxson, Stuart Staniford, and Robert Cunningham. A taxonomy of computer worms. In *Proceedings of the 2003 ACM workshop on Rapid Malcode*, pages 12–18. ACM Press, 2003. ISBN: 1-58113-785-0 `doi.acm.org/10.1145/948187.948190`.

# Conformance Checking of RBAC Policy and Its Implementation

Frode Hansen and Vladimir Oleshchuk

Agder University College, Department of Information and Communication Technology, Grooseveien 36, N-4876 Grimstad, Norway
{Frode.Hansen, Vladimir.Oleshchuk}@hia.no

**Abstract.** The purpose a security policy is to specify rules to govern access to system resources preferably without considering implementation details. Both policy and its implementation might be altered, and after introducing changes, it is not obvious that they are consistent. Therefore, we need to validate conformance between policy and its implementation. In this paper we describe an approach based on finite-model checking to verify that a RBAC implementation conforms to a security policy. We make use of the model-checking system SPIN, and show how to express RBAC policy constraints by means of LTL and how to model an RBAC implementation in SPIN's internal modeling language PROMELA.

## 1 Introduction

The aim of access control is to control the access to system resources available for users according to a security policy. The purpose of a security policy is to specify general rules to govern access to system resources preferably without considering implementation details. A security policy implementation is achieved through internal system mechanisms configured to enforce security policy requirements in the course of system utilization. Security policy should encompass both an organization's internal requirements to protect their information assets, and external requirements set by law enforcement boards, such as national/international directives on data and privacy protection. The decision making process of establishing and changing such requirements make security policies more static compared with their implementations. Security policy implementations are, to a greater extent, organization and system dependent, and would therefore be modified more often to reflect the dynamics of organizational and system changes. However, after introducing changes it is not obvious that policy and implementation is consistent with one another. For example, whenever a company establishes new job functions within the organizational structure, it would result in a constitution of new roles, reflecting the job functions, and users are assigned accordingly. This change should be implemented in conformance to the existing security policy. Moreover, the new roles might be in conflict with other, already implemented roles or they may require special security clearance from users if they are to be assigned to them. Thus new security requirements

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 144–155, 2005.

may appear to be established in the security policy. Furthermore, the security policy may also be modified as result of revision of the security requirements. For example, it can be done as reaction on exposure of unauthorized access, finding error in policy definition or new regulations specified by law enforcement boards. In that context, we consider a security policy as a requirement specification that the system behavior should conform to. Another reason to handle security policy separately is that some security requirements are common for all organizations in the same domain, for example for medical hospitals in the same country.

In this paper we describe an approach based on finite-model checking to verify whether an access control implementation conforms to its security policy. We suppose that access control is implemented within Role-Based Access Control (RBAC) [1] framework and the security policy describes constraints that the RBAC based implementation should satisfy. In our approach we use finite-model checking system SPIN [2]. We represent security policy as a set of claims that can be seen as a description of system behaviors that are both permitted and prohibited. We show how to express RBAC policy constraints by means of Linear Temporal Logic (LTL), and how to model RBAC implementation in SPIN's internal modeling language PROMELA. The paper is organized as follows. In Section 2 we briefly introduce the RBAC model. In Section 3 we consider various constraints in RBAC that can be used for security policy enforcement. Section 4 gives presentation of the model-checking system SPIN, its internal language PROMELA and LTL. In Section 5 we describe the conformance checking environment and express security constraints in LTL. Section 6 gives a demonstration of our idea through an case study, and Section 7 concludes our work.

## 2 Role-Based Access Control

In RBAC policies [1], a user's access to resources is based on organizational tasks and responsibilities that the user takes on in an organization. In RBAC users are associated with roles (e.g. Professor, Associate Professor, Student), and permissions are assigned to roles. The general model of RBAC is shown

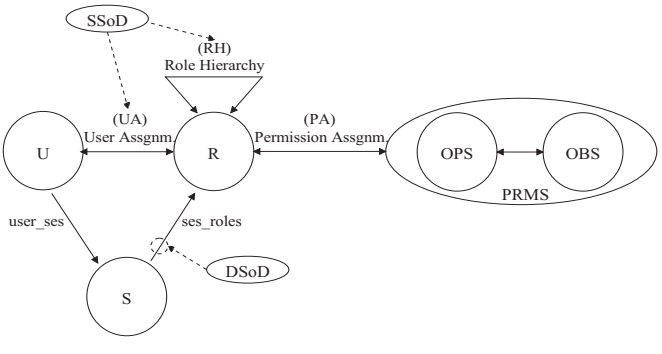



**Fig. 1.** Role-Based Access Control Model

in Figure 1 where core RBAC is the base model for any RBAC system and consists of five basic data element sets: Users (U), Roles (R), Objects (Obs), Operations (Ops) and Permissions (Prms). Users are defined as human beings, machines, networks and autonomous agents. A role represents a job function or organizational responsibilities, and permissions are an approval to execute operations on one or more RBAC objects. In addition, the model include sets of Sessions (S), where each session is a mapping between a user (user_ses) and an activated subset of roles (ses_roles) that are assigned to the user. On the sets U, R, Prms and S several functions and relations are defined. In addition, hierarchical RBAC adds the concept of role hierarchies where roles inherit permission from its junior roles. Generally RBAC can be formally defined as follows:

**Definition 1.** *RBAC model consists of the following components [1].*

- U, R, Prms *and* S*, represent the finite set of users, roles, permissions and sessions, respectively;*
- $\mathrm{RH} \subseteq \mathrm{R} \times \mathrm{R}$*, is a partial order on roles, called dominance relation, written as* $\succeq$*, and defines a hierarchy of roles, that is,* $\mathrm{r_1} \succeq \mathrm{r_2}$*, where* $(\mathrm{r_1}, \mathrm{r_2}) \in \mathrm{R}$*;*
- $\mathrm{UA} \subseteq \mathrm{U} \times \mathrm{R}$ *is the relation that associates users with roles;*
- $\mathtt{assign\_roles(u:U)} \rightarrow 2^{\mathrm{R}}$*, is the mapping of a user onto a set of roles s/he has assigned. Formally:* $\mathtt{assign\_roles(u)} = \{\mathrm{r} \in \mathrm{R} \mid (\mathrm{u,r}) \in \mathrm{UA}\}$*;*
- $\mathtt{assign\_users(r:R)} \rightarrow 2^{\mathrm{U}}$ *is the mapping of a role onto a set of users that has this role assigned. Formally:* $\mathtt{assign\_users(r)} = \{\mathrm{u} \in \mathrm{U} \mid (\mathrm{u,r}) \in \mathrm{UA}\}$*;*
- $\mathrm{PA} \subseteq \mathrm{R} \times \mathrm{Prms}$ *is the relation that associates a permission to a role;*
- $\mathtt{assign\_perms(r:R)} \rightarrow 2^{\mathrm{Prms}}$ *is the mapping of a role onto a set of permissions. Formally:* $\mathtt{assign\_perms(r)} = \{\mathrm{p} \in \mathrm{Prms} \mid (\mathrm{r,p}) \in \mathrm{PA}\}$*;*
- $\mathtt{user\_ses(u:U)} \rightarrow 2^{\mathrm{S}}$ *is the mapping of a user onto a set of sessions;*
- $\mathtt{ses\_roles(s:S)} \rightarrow 2^{\mathrm{R}}$ *is the mapping of each session to a set of roles;*
- $\mathtt{avail\_session\_perms(s:S)} \rightarrow 2^{\mathrm{Prms}}$*, the permissions available in a session, that is,* $\cup_{\mathrm{r} \in \mathtt{ses\_roles(s)}} \mathtt{assign\_perms(r)}$*.*

## 3 Security Constraints

RBAC policy *constraints* play an important part in the RBAC framework and provide additional flexibility since constraints allow the system to control user behavior according to specified requirements [3]. Constraints express restrictions on the various RBAC components, such as *prerequisite-*, *cardinality-* and *Separation of Duty* (*SoD*) constraints. *Prerequisite* constraints can be applied to a user-role assignment that requires the assignment of another role to the user before the user-role assignment may succeed. *Cardinality* constraints are specified in order to enforce a numerical restriction on RBAC components, such as limiting the number of users assigned to a specific role.

Over the years, researchers have focused on enforcing variations of SoD constraints on RBAC components. SoD has long been recognized as an important security principle and is regarded to be beneficial to prevent fraud and errors,

by distributing responsibility and authority for an action or task among several users in a policy domain, and there have been proposed a range of taxonomies for it in the literature [4–7]. Common for all proposed taxonomies, are that they identify *Static SoD* (SSoD), and *Dynamic SoD* (DSoD), as two main categories of SoDs (see Fig. 1). In its simplest form, SSoD is enforced on the assignment of users to roles where a user is never allowed to be assigned to roles that share an SSoD relation. Simple DSoD allows user-role assignments that do not create *conflict-of-interest* as long as they are activated separately in distinct sessions.

Simon and Zurko [4], and Gligor *et al.* [6] present variations of SoD in role-based environments and show that it is possible to define SoD constraints on users, roles, objects and operations and combinations of these. In addition to the broad categories of simple SSoD and simple DSoD, they identify: *Object-based SoD* (*ObjSoD*), which prevents a user from performing more than one operation on an object; *Operational SoD* (*OpSoD*), that prohibits a user acting on its own, from performing all the operations necessary to complete a critical business function; and lastly, *History-based SoD*, which is a combination of *ObjSoD* and *OpSoD* that excludes a user from performing all the operations specified in a business task on the same object or a collection of objects. Additional taxonomies for *conflict-of-interest* constraints, were described by Nyanchama and Osborn [7]. In addition to those already presented above, they considered: *User-user constraints*, which are constraints to be enforced on two or more users that should never share user-role associations; *Permission-permission constraints*, that prevent assignments of two or more conflicting permissions to the same role; and *Static user-role constraints*, which encompass conflict-of-interest issues that exist between users and roles, where a user should never be assigned to a particular role, as long as the user e.g. lacks clearance or competence.

Recently, researchers have focused on integrating context as a natural part of the access control policy [8–11] to allow environmental factors (e.g. location, time, system state, etc.) influence access decisions carried out by the system. Bertino *et. al.* presented in [10] a Temporal Role-based Access Control (TRBAC) model for specifying temporal constraints and dependencies on enabling and disabling of roles. Joshi *et. al.* [11] extended and generalized the TRBAC model to incorporate temporal constraints on role *activation*, the *periodicity* and *duration* in which roles can be enabled, *user-role* assignments, *role-permission* assignments, as well as adding constraints on the enabling and disabling of constraints and SoD relations. In [8, 9], Hansen and Oleshchuk extend the RBAC model by specifying spatial restrictions on permissions assigned to roles, where a role may have different permissions enabled, dependent on the location. Spatial constraints on permissions assigned to a role can be beneficial when specifying the access control policy in mobile environments where the location in which a user access services from is a key component. They also, identify Spatial SSoD and Spatial DSoD, where roles are mutually exclusive reliant on the location.

## 4 Model-Checking in SPIN

To study conformance between security policy and its implementation in RBAC framework we make use of the model-checking system SPIN [2]. Generally, verification models in SPIN are focused on proving correctness of process interactions, and they attempt to abstract as much as possible from internal sequential computations. A model to be analyzed in SPIN must be specified in internal specification language called PROMELA. Each model described in PROMELA is a large finite-state machine and can be verified with SPIN under different types of assumptions about the environment (e.g., message loss, message duplications etc.). Given a system model specified in PROMELA, SPIN can either perform random simulations of the system's execution or can perform an efficient on-the-fly verification of the system's correctness properties. The verifier can also be used to verify the correctness of system invariants and correctness properties expressed in Linear Temporal Logic (LTL) formulae. In LTL, in addition to Boolean operators, we can use temporal operators such as ALWAYS (denoted by $\Box$) and EVENTUALLY (denoted by $\Diamond$) (see [2] for more details). LTL allows us to express temporal properties we expect the system behavior will conform to during the system lifetime. Properties expressed in the formal LTL notation both gives an unambiguous presentation of possible system behavior and also possibility to verify whether the system model conform to the requirements.

In context of this paper we design a model based on RBAC such that the model, described in PROMELA, contains only details related to RBAC functionality. We abstract the model from implementation details to get a feasible model, of a reasonable size that can be analyzed by SPIN. LTL is used to express constraints or security requirements based on security policy. Since any implementation should conform to the security policy it implements, the model in PROMELA should conform to LTL expressions representing security policy constraints. SPIN has in-build LTL-translator that transforms LTL expressions into PROMELA never-claims. Then by in running verification mode we can verify whether any possible behaviors of the model satisfy LTL expressions by showing that there are no model behaviors that are represented in never-claims. More details will be given in the following sections.

## 5 Security Policy and Its Implementation

The objective of RBAC framework is to implement security according to an organizational security policy. Since both security policy and its RBAC implementation can be changed separately, by different bodies, the security responsible should be able to validate conformance between RBAC implementation and security policy when this is necessary. This can be a resource consuming task, and a manual inspection would most likely not comprehend all the aspects of a complex security policy. Moreover, conformance analysis can be computational demanding task; therefore we need a computer-based tool to simplify it. Typical security requirements for RBAC based model can be formulated in plain English, however

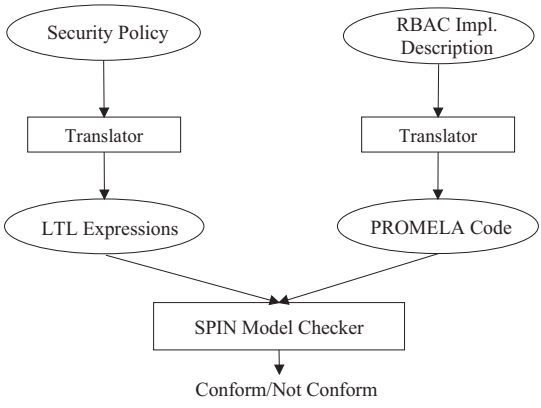



**Fig. 2.** Conformance checking of Security Policy and RBAC Implementation

in order to verify whether RBAC-based implementation conforms with security requirements we have to express them in some formally defined language with appropriate expressive power. The diagram in Fig. 2 shows the overview over the main steps in system that verifies whether security policy constraints conform to its implementation. The system is based on software verification system SPIN and to be able to use SPIN as a conformance checker, we translate the security policy (constraints) into LTL expressions and model RBAC implementation in PROMELA. The model checker use the model of the RBAC implementation as input to be validated up against the security constraints represented by LTL expressions.

### 5.1 Specification of Security Constraints with LTL

To be able to use LTL to formally specify security constraints on system behavior we utilize variables and constants from the system model and define predicates we may use to express both expected and unexpected behavior of the system model. In the following examples we demonstrate how some security constraints from Sect. 3 can be expressed in plain English and, formally in LTL language. We use notation similar to the one described in Sect. 2, with minor adjustments.

*Example of a Pre-requisite constraint.*

- In English: A user assigned to role `accounting manager` should already be assigned to role `accountant`.
- In LTL: $\forall$`u`$\in$`U`, $\Box$ `(account_manager`$\in$`assign_roles(u))` $\rightarrow$ `(accountant` $\in$`assign_roles(u))`. Thus, if a user `u` gets assigned to the `account_manager` role, this user must as well be assigned to the `accountant` role.

*Example of a User-user constraint.*

- In English: Alice and Bob should not share role assignments.
- In LTL: $\Box$`(assign_roles(Alice)` $\cap$ `assign_roles(Bob)` == $\emptyset$`)`. The intersection of the role sets assigned to Alice and Bob should be empty.

*Example of a SSoD constraint.*

- In English: A user should never be assigned to both roles `financial manager` and `auditor`.
- In LTL: $\Box$(!(financial_manager $\in$ assign_roles(u) && auditor $\in$ assign_roles(u))), where, ! denotes negation. Thus if the role financial_manager exists in u's role set, then the auditor role does not, and vice versa.

DSoD is enforced on the activation of roles at system run-time and restricts the availability of the roles that can be performed simultaneously within a single session. DSoD is defined in [1] as a pair, (rs, n) $\in$ DSoD, where DSoD is defined as a set {(rs,n) | rs $\in$ R and n is a non negative number, n $\geq$ 2} and (rs, n) $\in$ DSoD means that no user may activate n or more roles from rs in any single session. However, in RBAC, users are allowed to have several sessions activated simultaneously, thus it is possible for a user to activate dynamically conflicting roles in separate sessions. Therefore, we must be able to constrain the activation of roles a cross multiple user sessions.

**Definition 2.** *DSoD, for one user, with multiple sessions.*
$\forall$(rs, n) $\in$ DSoD, $\left| \bigcup_{s \in \text{user\_ses(u)}} \text{ses\_roles(s)} \cap \text{rs} \right| \leq$ (n - 1).

Enforcing multiple session DSoD, prevents a user from activating n-1 conflicting roles simultaneously in distinct sessions and can be specified in LTL language as follows in the example below.

*Example of DSoD for one user in multiple sessions*

- In English: A user should never be able to activate roles initiator and authorizer simultaneously cross a user's sessions.
- In LTL: $\forall u \in U, \forall s \in S, \Box$ (!(initiator $\in \bigcup_{s \in \text{user\_ses(u)}}$ ses_roles(s) && authorizer $\in \bigcup_{s \in \text{user\_ses(u)}}$ ses_roles(s))).

As described in Sect. 3, Object-based DSoD (ObjDSoD) prevents a user from performing more than one operation on an object and can be defined as follows.

**Definition 3.** *Object-based DSoD.*
$\forall u \in U, \forall$(OPS,OBS,n) $\in$ ObjDSoD, $\forall ob \in$ OBS, $\Rightarrow$
$\left| \bigcup_{r \in \text{allowed\_roles(u)}} \text{allowed\_ops(r, ob)} \cap \text{OPS} \right| \leq$ (n - 1), *where*
allowed_roles(u) = $\bigcup_{s \in \text{user\_ses(u)}}$ ses_roles(s).

The DSoD example above, did not allow a user to activate two confliction roles, initiator and authorizer, cross multiple user sessions. For this example the task of initiate and authorize payments are separated in two different roles. However, in cases where it is desired to accumulate these tasks in one role and still consider the operations conflicting, one need to enforce ObjDSoD. This facilitates that any user assigned to a particular role may carry out initiate and authorize payment, however, not both on the same payment.

*Example of a ObjDSoD constraint.*

- In English: A user is not allowed to initiate and authorize the same payment.
- In LTL: ∀u∈ U, ∀r∈ allowed_roles(u),□((initiate ∈ allowed_ops (r, payment)) → !◊(authorize ∈ allowed_ops(r, payment)))

## 6 Case Study

In this section we demonstrate our idea through a case study. We have applied our approach a scenario described in [12, 13] that deals with the task of processing invoices subsequent to purchasing items.

When purchasing items there are three types of sub-tasks involved in processing an invoice: recording the arrival of an invoice and additional information; verifying that the received items is in correspondence to the invoice received; and authorizing the payment of purchased goods. In order to minimize the risk of fraud, we need to separate these sub-tasks in order to prevent that possible payments made out to fictitious companies without actual delivery of the "purchased" items [12]. There are different methods on how to separate these sub-tasks in RBAC (see Sect. 3). We show how separation of sub-tasks can be performed by use of SSoD and ObjDSoD. To be able to validate the conformance between RBAC implementation and an RBAC policy in SPIN, the RBAC implementation is modeled in PROMELA and RBAC policy constraints are expressed by means of LTL expressions[1].

### 6.1 Static SoD

SSoD is the simplest method to separate the sub-tasks by assigning one sub-task to each role and a user should never be assigned to more than one of these roles. Thus, we can define three different role types that cannot have common users: clerks that `record` the arrival of an invoice by entering it in the system; purchasing officers that `verify` the invoice and check that the items have been received; and supervisors who `authorize` payments. Data structure for RBAC-based implementation for processing invoices are given in Fig. 3. It contains information about users, roles and permissions. There exist three possible operations: record invoices, verify invoices and authorize payments, that can be executed in the task of processing invoices. Figure 3(a) and 3(b) shows that SSoD is implemented by assigning only one possible permission (operation on the object) to each role, and by assigning distinct users to distinct roles. This should ensures that no user is capable of performing more than one operation on an invoice. The corresponding presentation in PROMELA is shown below.

```
active proctype currentRBACImpl (){
  usr_role_assmnt(Alice,Supervisor); usr_role_assmnt(Bob,Officer);
  usr_role_assmnt(Claire,Clerk); perm_role_assmnt(Supervisor,authinv);
  perm_role_assmnt(Officer,verinv); perm_role_assmnt(Clerk,recordinv);
}
```

[1] We also show how SSoD constraints can be translated into PROMELA by use of assertions.

| Users | Roles |
|---|---|
| Alice | supervisor |
| Bob | officer |
| Claire | clerk |

(a) User-role assignments.

| Roles | Permissions |
|---|---|
| clerk | p1 |
| officer | p2 |
| supervisor | p3 |

(b) Permission-role assignments.

| Permissions | Object, Operation |
|---|---|
| p1 | invoice, record |
| p2 | invoice, verify |
| p3 | invoice, authorize |

(c) Permission mapping.

**Fig. 3.** Data structure from RBAC-implementation $I_{P_1}$

The user-role assignment function (`user_role_assmnt(user, role)`) is an *inline* function in PROMELA that defines a 2-dimensional array (`users[user].roleID[role]`) containing a Boolean value, equal to TRUE if the role is assigned to the user and FALSE otherwise. As a part of security policy `P`, we can demand that a user should never be assigned to more than one of the roles Supervisor, Officer or Clerk. In the notation from Sect. 5 it means that `SSoD={Supervisor,Officer,Clerk}`. To check if the current RBAC implementation $I_{P_1}$ conforms with the security policy `P`, we need also to translate policy `P` into a language that the verification process understand.

**Assertions Representing Security Policy.** An *Assertion* is one of several PROMELA language constructions that can be used to check if an RBAC implementation conforms with a security policy. The assertion statement evaluates the expression it holds and failure is detected by the verification process when the expression is evaluated to the boolean value FALSE.

```
assert((us&&uo) != 1); assert((us&&uc) != 1); assert((uo&&uc) != 1);
```

The code example above shows how assertion can be used to express SSoD constraints in PROMELA to evaluate the conformance between RBAC implementation and security policy related to user-role assignments. To be able to check if any conflict arise, we need to check the existing user-role assignments in the implementation. So if, Claire is temporary assigned to the officer role (`uo == TRUE`) in addition to the originally assigned clerk role (`uc == TRUE`), the assertion statements will report failure from the verification run `((uo&&uc) == TRUE)`.

**LTL Expression Representing Security Policy.** Another approach to express the SSoD constraint is to use LTL as follows.

```
□(!((supervisor ∈ assigned_roles(u) && clerk ∈ assigned_roles(u)) ||
(supervisor ∈ assigned_roles(u) && officer ∈ assigned_roles(u)) ||
(clerk ∈ assigned_roles(u) && officer ∈ assigned_roles(u))))
```

In SPIN the last expression can be presented as: `[](!((us && uo) || (us && uc) || (uo && uc)))`. That is, it is always (denoted `[]`) the case that `((us && uo)`

|| (us && uc) || (uo && uc)) should remain FALSE throughout the verification run. In other words, if any user u, for some reason, is assigned to more than one of the conflicting roles, the result of the verification of the system model would report an error corresponding to the rule of SSoD.

### 6.2 Object-Based DSoD (ObjDSoD)

For some organizations, enforcing SSoD constraints on the time when users are authorized for a role might be too restrictive. Normally, purchasing officers should be able to record the arrival of an invoice if there exists a backlog of unprocessed invoices. Similarly, supervisors should be able to perform verification of invoice details [12, 13] when necessary. Data structures from RBAC implementation, $\mathtt{I}_{P_2}$, is shown in Fig. 4. The object-operation mappings remain unchanged from Fig. 3(c). Assigning additional privileges to roles `officer` and `supervisor` result in that we need at least two users assigned to each of the roles to avoid any unverified or unauthorized invoices. The roles `officer` and `supervisor` have additional privileges compared to Fig. 3(b). To avoid any fraudulent behavior, an officer entering an invoice should not be able to verify this invoice, and a supervisor that verifies an invoice should not be able to authorize a payment for purchased goods related to the same invoice. This constraint corresponds to ObjDSoD. Therefore, if a user, acting in one of the roles `officer` or `supervisor`, performs an operation on an invoice, will it exclude the user from performing a different operation on the same invoice. Based on this, we can formulate two constraints as follows:

**Constraint 1.** *A user assigned to the role of an* `officer` *should never be able to perform both operations,* `record` *and* `verify`*, on the same invoice. That is,* $\mathtt{ObjDSoD}_1$ `= {{record, verify}, invoice, 2}`.

**Constraint 2.** *A user assigned to the role of an* `supervisor` *should never be able to perform both operations,* `verify` *and* `authorize`*, on the same invoice. That is,* $\mathtt{ObjDSoD}_2$ `= {{verify, authorize}, invoice, 2}`.

| Users | Roles |
|---|---|
| Alice | supervisor |
| Bob | officer |
| Claire | clerk |
| Arthur | supervisor |
| Elise | officer |

(a) User-role assignments.

| Roles | Permissions |
|---|---|
| clerk | p1 |
| officer | p1, p2 |
| supervisor | p2, p3 |

(b) Permission-role assignments.

**Fig. 4.** Data structures from RBAC-implementation $I_{P_2}$

**PROMELA Representation of the RBAC Implementation.** Dynamic policy restrictions such as ObjDSoD must be validated during run-time when the users are performing operations on an invoice, ruled by the RBAC implementation. Thus we have developed an execution environment modeling the RBAC implementation where users perform tasks governed by their access privileges.

The behavior of the model consists of four different processes, where first, a random user is selected to perform some task. Then, according to the user's role-assignments, a role is activated. A random permission is selected among the permissions assigned to the activated role (`recordinv`, `verinv` or `authinv`). The system then enters into a critical phase, where the actual operation is executed on the invoice. In this critical phase we implement Constraints 1 and 2, specified in the security policy.

```
p_verify:
do
::((invoice[i].pID[verinv]==noUser)&&(invoice[i].pID[recordinv]!=noUser)
  &&(invoice[i].pID[recordinv]!=vu))->invoice[i].pID[verinv]=vu; break;
:: else -> i++; goto p_verify;
od
```

If Constraint 1 is implemented correctly, its corresponding PROMELA representation would be as shown in the code example above. The 2-dimensional array (`invoice[i].pID[permissionID]`) keeps track of the users who have performed operations on the invoice `i`. Three conditions must be satisfied; first, no user has verified the invoice, second, the invoice must be registered (recorded) in the system, and third, the user who recorded the invoice must not be the same as the one that is going to verify it. If all conditions are satisfied, the user's userID (`vu`) is registered as the user who verified the invoice.

```
p_authorize:
do
::((invoice[i].pID[authinv]==noUser)&&(invoice[i].pID[recordinv]!=noUser)
   &&(invoice[i].pID[verinv]!=(noUser)&&(invoice[i].pID[verinv]!=au))->
   invoice[i].pID[authinv] = au; break;
::else -> i ++; goto p_authorize;
od
```

Similarly, the code example above, shows the implementation of Constraint 2 in PROMELA. The conditions that need to be satisfied is similar to that above; no user has authorized the invoice, the invoice must be recorded and verified, and the user who verified it is not the user who tries to authorize (`au`) it.

**LTL Expressions Representing Security Policy.** To be able to validate the conformance between security policy (in LTL) and its RBAC implementation (in PROMELA), we need to add some variables to the model for verification purposes. In this case, on a successful verification or authorization of an invoice, the userID of the users who recorded, verified and authorized the invoice will be stored in variables `urecorded, uverified` and `uauthorized` correspondingly.

Let us rephrase Constraint 1. The user who recorded the invoice should never be the one that verified the invoice. This result in the following LTL ex-

pression: □(!((record ∈ allowed_ops(officer, invoice) && verify ∈ allowed_ops(officer, invoice)))). In SPIN this can be expressed as [](!(p)), where the property p is (urecorded == uverified). Similar, Constraint 2 leads to the following LTL expression: □(!((verify ∈ allowed_ops(supervisor,invoice) && authorize ∈ allowed_ops(supervisor,invoice)))). In SPIN this is can be expressed as [](!(q)), where the property q is (uverified == uauthorized). If, during run-time, it is possible for a user to record and verify (i.e., property p == TRUE) (or verify and authorize, i.e. property q == TRUE) the same invoice, running the verification process would detect the error and the RBAC implementation needs to be altered to meet the requirements defined in the security policy.

## 7 Conclusion

The purpose of this paper is to demonstrate how to use finite model checking to conformance testing between security policy expressed in form of LTL claims and its implementation in RBAC framework. The purpose is to develop a practical tool that will help security administrators to maintain their RBAC based access control system in conformance with security policy claims.

## References

1. Ferraiolo, D.F., Kuhn, D.R., Chandramouli, R.: Role-Based Access Control. Artech House (2003)
2. Holzmann, G.: The Spin Model Checker. Addison-Wesley, Massachusetts (2004)
3. Giuri, L., Iglio, P.: A formal model for role-based access control with constraints. In: 9th IEEE Computer Security Foundations Workshop. (1996) 136–145
4. Simon, R., Zurko, M.E.: Separation of duty in role-based environments. In: 10th IEEE Computer Security Foundations Workshop. (1997) 183–194
5. Kuhn, D.R.: Mutual exclusion of roles as a means of implementing separation of duty in role-based access control systems. In: 2nd ACM workshop on Role-based access control. (1997) 23–30
6. Gligor, V.D., I.Gavrila, S., Ferraiolo, D.: On the formal definition of separation-of-duty policies and their composition. In: IEEE Symp. Sec. Priv. (1998) 172–183
7. Nyanchama, M., Osborn, S.: The role graph model and conflict of interest. ACM Trans. Inf. Syst. Sec. **2** (1999) 3–33
8. Hansen, F., Oleshchuk, V.: Spatial role-based access control model for wireless networks. In: IEEE Vehicular Technology Conf. Volume 3. (2003) 2093 – 2097
9. Hansen, F., Oleshchuk, V.: SRBAC: A spatial role-based access control model for mobile systems. In: 7th Nordic Workshop on Secure IT Systems. (2003) 129–141
10. Bertino, E., Bonatti, P.A., Ferrari, E.: TRBAC: A Temporal Role-based Access Control Model. ACM Trans. Inf. Syst. Sec. **4** (2001) 191–223
11. Joshi, J.B.D., Bertino, E., Latif, U., Ghafoor, A.: Generalized Temporal Role Based Access Control Model (GTRBAC). Technical Report, CERIAS TR 2001-47, Purdue University, USA (2001)
12. Clark, D.R., Wilson, D.R.: A comparison of commercial and military computer security policies. In: IEEE Symp. Sec. Priv. (1987) 184–194
13. Nash, M.J., Poland, K.R.: Some conundrums concerning separation of duty. In: IEEE Symp. Sec. Priv. (1990) 201–209

# A Practical Aspect Framework for Enforcing Fine-Grained Access Control in Web Applications

Kung Chen and Chih-Mao Huang

Department of Computer Science, National Chengchi University,
Wenshan, Taipei 106, Taiwan
{chenk, g9224}@cs.nccu.edu.tw

**Abstract.** Access control is a system-wide concern that has both a generic nature and an application dependent characteristic. It is generic as many functions must be protected with restricted access, yet the rule to grant a request is highly dependent on the application state. Hence it is common to see the code for implementing access control scattered over the system and tangled with the functional code, making the system difficult to maintain. This paper addresses this issue for Web applications by presenting a practical access control framework based on aspect-oriented programming (AOP). Our approach accommodates a wide range of access control requirements of different granularity. AOP supports the modular implementation of access control while still enables the code to get a hold of the application state. Moreover, framework technology offers a balanced view between reuse and customization. As a result, our framework is able to enforce fine-grained access control for Web applications in a highly adaptable manner.

## 1 Introduction

The principle difficulty in designing security concern such as access control into an application system is that it is a system-wide concern that permeates through all the different modules of an application. Although there is a generic need to enforce access control for protected resources, yet the specific constraint for granting access to each individual resource may not be the same. Hence in current practices it is very often to see the code for implementing access control scattered over the whole system and tangled with the functional code. This is not only error-prone but also makes it difficult to verify its correctness and perform the needed maintenance; Web applications are no exceptions. Indeed, "broken access control" is listed as the second critical Web application security vulnerability on the OWASP top ten list [15].

A better way to address this problem is to treat security as a separate concern and devise a framework where the generic requirements of access control are captured and the resource-specific constraints can be encapsulated and separated from the functional part of an application [3]. This will not only improve the application's modularity but also make the task of enforcing comprehensive access control more tractable. The Java Authentication and Authorization Services (JAAS) of J2EE [17] is a well-known attempt toward such a solution. Furthermore, it takes one step forward to pro-

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 156 – 167, 2005.

vide declarative security where access control constraints can be specified declaratively in a configuration file without actual coding. This makes access control highly adaptable.

While sufficient for meeting common access control requirements encountered in Web application development, there are many fine-grained requirements that cannot be satisfied in a modular manner using middleware services such as JAAS. Most notably are the cases when the access control constraints must be defined at the data level. For example, in a B2B E-Commerce site, users from any registered organizations have the privilege to execute the "viewOrder" function, but they are allowed to view only orders of their own organization. This is called instance-level access control constraints [11]. Furthermore, within a data record, certain specific fields, such as credit card number, may have to be excluded from screen view to protect the user's privacy. We refer to this as field-level access control constraints. The declarative security mechanism of JAAS does not cover such fine-grained constraints; to handle them well, we still have to write scattered and tangled code.

It is thus highly desirable to devise a systematic approach and proper mechanisms to realize such fine-grained access control requirements for Web applications in a modular and adaptable manner. We worked toward this goal from two opposite ends and managed to meet in the middle. At one end, the objective is to accommodate requirements. We use a flexible modeling scheme based on user-function-data relationship that can satisfy a wide range of access control requirements of various granularity levels, including both instance and field levels. At the other end, since access control is a system-wide crosscutting concern, we must impose considerable architectural disciplines on Web applications to layout a good foundation for enforcing the required access control modularly. In particular, we follow the well-accepted Model-View-Controller (MVC) [7] architectural pattern and adopt the popular Apache Struts framework [1] to structure our Web applications.

Next, to bridge these two ends, we devise a flexible implementation scheme that does not only support those access control requirements but also integrates seamlessly into the underlying application architecture. Specifically, the emerging techniques of aspect-oriented programming (AOP) [12] are employed to enforce access control constraints in Struts-based Web applications. We take full advantage of AspectJ [13] to design and implement an aspect framework that can meet the diverse information needs of those requirements while allowing reuse and application-specific customization. Furthermore, the codes that implement the required access control are encapsulated and linked to functional modules in a very low-coupling manner, rendering the resulting systems highly modular and adaptable.

The rest of this section gives a brief introduction to AspectJ and the Struts framework. Section 2 describes our approach to access control modeling. Section 3 presents our aspect framework in detail. Section 4 compares related work. Section 5 concludes and sketches future work.

### 1.1 AOP and AspectJ

AOP is a new programming paradigm to support separation of concerns in software development. It addresses the program modularity issues of a crosscutting concern through a new kind of modules, called *aspect,* and new ways of module composition.

In AOP, a program consists of many functional modules, e.g. classes in OOP, and some aspects that captures concerns that cross-cuts the functional modules, e.g. security. The complete program is derived by some novel ways of composing functional modules and aspects. This is called *weaving* in AOP. Weaving results in a program where the functional modules impacted by the concern represented by the aspect are modified accordingly. In languages such as AspectJ, the weaver tool is tightly integrated into the compiler and performs the weaving during compilation.

To facilitate the weaving process, a set of program *join points* are introduced to specify where an aspect may cross-cut the other functional modules in an application. Typical join points in AspectJ are method execution and field access. A set of join points related by a specific concern are collected into a *pointcut*. Code units called *advice* in an aspect are tagged with a pointcut and determine how the application should behave in those crosscutting points. There are three kinds of advice in AspectJ: *before*, *after*, and *around*. The before advice and the after advice are executed before and after the intercepted method, respectively. The case for the around advice is more subtle. Inside the around advice, we can choose to resume the intercepted method by calling the special built-in method *proceed()*, or simply bypass its execution.

Furthermore, AspectJ also allows aspect inheritance, abstract aspect, and abstract pointcut. We can write an aspect without any reference to a join point by declaring the pointcut abstract. A sub-aspect then extends the abstract aspect and defines the concrete pointcut. As we shall demonstrate later, these abstract aspects open the way to build generic aspects that are essential to an aspect framework.

### 1.2 Web Application Architecture and the Struts Framework

We follow the mainstream approach to Web application architecture that structures an application using the Model-View-Controller (MVC) architectural pattern [7]. The model components encapsulate the application components in the business logic and data tiers. The view components are those pieces of an application that display the information provided by model components and accept input. The controller is a special program unit that coordinates all the activities of the model and view components. It acts as a central point of control within an application.

Currently, most Web application frameworks are MVC-based. In particular, we choose the popular open source Struts framework [1]. It is compatible with Sun's J2EE platform and primarily based on Servlet [19] and JSP [18] technology. Figure 1 illustrates the structure of a Struts-based Web application. The controller is implemented using a dedicated *ActionServlet* with the assistance of an XML configuration file, *struts-config.xml*. Every user request is dispatched to an *Action* class by the *ActionServlet* according to the *action mapping* defined in the configuration file. These actions are responsible for serving user requests or passing them to the correct business model components, and for returning the correct view element that the *ActionServlet* should forward to afterwards. This view forwarding is also based on the mapping information specified in the configuration file. The view element is often a JSP page, extended with custom tag libraries provided by Struts.

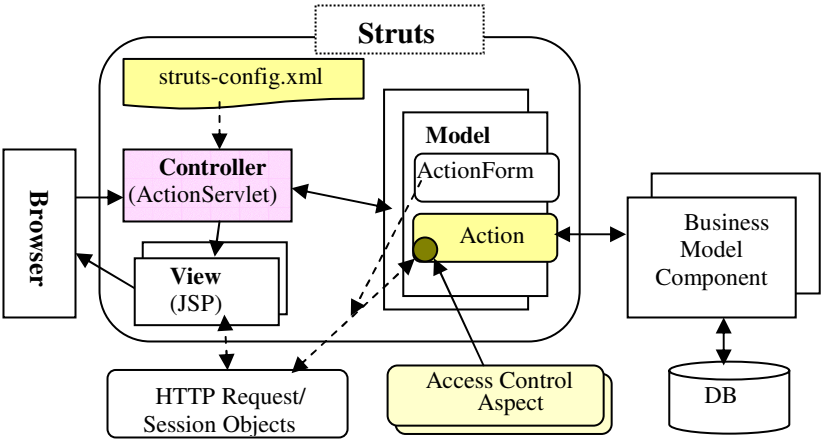



**Fig. 1.** Struts-Based Web application architecture

Moreover, the HTTP request object and the session object provided by the Servlet framework serve as the shared data repository between action classes and JSP views. User input captured via an *ActionForm* is stored in the two objects; action classes also put their execution results in these objects. Hence JSP views can fetch both user input and execution results from them for display. For example, user authentication results and user profiles are kept in the session object for authorization and other purposes.

## 2 Access Control Modeling

Access control, also known as *authorization*, is a process by which users, after being identified, are granted certain privileges to information, system functions, or resources. The step to identify a user is usually called *authentication*. Username and Password check is the most basic form of authentication, while digital certificates and biometrics are more advanced ways of authentication.

Since RBAC [16], there have been many approaches proposed to model access control requirements for applications purposes. Here we take a simple yet generic approach that can support a wide range of access control requirements. We model the interaction between a user and a Web application as a sequence of access tuples of three elements: <*user, function, data*>, indicating a user's request to execute the function on a specific type of data object(s). The access control rules of an application determine which access tuples are allowed and which must be denied. They are derived from the application's access control requirements.

In designing the form of our access control rules, we focus on the functionalities of an application and specify the access control rules in a function-oriented manner. Furthermore, as authentication is required prior to authorization, we also make "*AuthenticationType*" part of the rule; the type can be id/password (PWD), digital certificate (DC), or any other supported methods of user identification. Specifically, the access control rules take the following form:

*Rule: <funName, authType, dataClassName[-fieldNames], constraint>*

Here *funName* is the name of a function whose access needs to be restricted, *authType* is the required user authentication type, and *dataClassName* refers to the type of the data objects being operated by *funName*. Note that *[-fieldNames]* following *dataClassName* is optional; when present, it specifies the list of fields whose contents must be masked before presenting the data object(s) in context to the user. Last, the *constraint* is a Boolean expression which must be evaluated to true to grant the attempted access. We assume a few basic relational operators, such as *equal*, *less* and s*et membership*, available for specifying the constraint expression. If the constraint does not refer to any information on the data object, the *dataClassName* element can also be left unspecified as "__".

Clearly, the more related entities we can refer to in the constraint expression the wider the scope of access control requirements we can support. For generic purposes, we take an object-based approach to specify the constraints and supply three generic objects: *User*, *Fun*, *Data*, with various attributes that the constraint expression can refer to. Typical attributes for the *User* object include user's name and roles in an organization. The attributes of the *Fun* object include the function's full name and the arguments passed to it, while the fields of the *dataClassName* are the default attributes of the *Data* object to support fine-grained control. The specific set of attributes depends on individual application's needs.

Furthermore, to accommodate more access control requirements, we provide another two objects for specifying the constraints. First, the context object (*Cxt*) provides methods to retrieve the time and location of the attempted access. This is the most often used contextual information for access control. Second, the application object (*App*) is global to an application and stores various specific parameters related to access control. For example, certain functions are accessible only during working days and from specific machines. These application-wide parameter definitions can be provided easily through a standard Java property file.

**Example:** The following is a set of access control requirements and corresponding rules for a typical online shopping and order management system. Note that "&&" stands for the *and* operator, and "||" the *or* operator.

C1: Only registered (authenticated) users can search the detailed product catalog.
R1: <*searchDetailedCatalog, PWD, __, true*>

C2: All registered users (a.k.a. customer) can create order, but only VIP customers can create orders whose total amount exceed $100,000.
R2: <*createOrder, PWD, __, ( lessEq(Fun.getArgument("total"), 100000)*
*|| equals(User.getAttr("VIP"), true) )* >

C3: Only sales managers authenticated through digital certificates can delete order objects.
R3: <*deleteOrder, DC, Order, contains(User.getAttr("Roles"), "Sales")*
*&& contains(User.getAttr("Roles"), "Manager")*>

C4: Customers can list (view) their own orders, but the credit card number should be excluded from display.

R4: <*listOrders, PWD, Order[-creditCardNumber],*
*equals(User.getAttr("Name"), Data.getAttr("Owner"))*>

C5: Unclassified orders can be printed in batch mode by sales from dedicated machines during working days.
R5: <*batchPrintOrder, PWD, Order,*
*contains(User.getAttr("Roles"), "Sales")*
*&& contains(App.getAttr("WorkingDays"), Cxt.getDay())*
*&& contains(App.getDedicatedMachines(), User.getAttr("clientIP"))*
*&& equals(Data.getAttr("SecurityLevel"), "Unclassified")*>

This form of access control rules is very flexible and can model a multitude of requirements, from simple RBAC to sophisticated instance and field level constraints.

## 3 Building the Aspect Framework

This section describes our design of the aspect framework and presents it in detail.

### 3.1 Choosing the Pointcuts

As stated earlier, access control is a cross-cutting concern that AOP aims to modularize well. Yet, it is not something like function tracing that is completely orthogonal to the functions and all one has to do in AOP is to define the functions to be traced as the pointcuts without any involvement form the underlying functions. In contrast, access control decisions, especially fine-grained ones, need substantial information from the application about the context in which the decisions are required. Hence access control aspects depend on the cooperation from the application to a significant degree. On the other hand, as we are not inventing a totally new style of application architecture, we need to carefully look for the proper pointcuts to weave in aspects so that a suitable balance between information need and non-invasiveness is achieved.

Given the requirements above, central to our design of the aspect framework is to determine which type of program units in a Struts-based application to weave in the aspect code. This in turn is greatly influenced by the following design considerations. First of all, all the information required for enforcing access control rules and filtering out unauthorized contents must be available to the aspect code, since fine-grained access control cannot be realized without detailed application state. Second, in case that the attempted access must be denied, exception propagation and handling must not incur significant impacts on the program structure. Third, the correspondence between an access control rule and an enforcing aspect should be direct and clear for management and maintenance purposes.

Based on these considerations, we conducted some experiments and investigation. Finally, we choose the *user action classes* in Struts as the targets for access control aspect weaving. Action classes play the role of gateway between the presentation tier

and the business and the data tiers. All action classes must inherit from the class *Action* and implement an `execute` method[1] with the following signature:

```
public ActionForward execute
            (ActionMapping mapping, ActionForm form,
             HttpServletRequest req,
             HttpServletResponse resp)
```

It turns out that the `execute` method is indeed the proper join point to weave in our advice code for its arguments expose the right information we need. Firstly, user input and any intermediate results, including user authentication records and user profiles, are all stored in the request or session objects, which are available through the `HttpServletRequest` argument, `req`. Therefore, via the `args()` pointcut of AspectJ, our aspects will have all the information needed for enforcing the access control and filtering out unauthorized contents.

Secondly, the control transfer between actions and views is specified in the *struts-config.xml* file, and is realized through the action mapping argument passed to the method and the `ActionForward` object returned by an action class. Hence we can define proper exception pages in the configuration file and forward to them when an access request must be denied. Apparently, this exception handling scheme fits very well into Struts. Lastly, as will be shown later, an access control rule corresponds to its enforcement aspect quite directly since a user function will usually be supported by an action class.

### 3.2 Constructing the Aspects

Having chosen the user actions as our weaving targets, the next step is to investigate the structure of access control aspects for the framework's purpose. Following the general spirit of framework construction [4][6], we shall divide the aspect code into two parts: generic part realized by abstract aspects and rule specific part realized by concrete aspects. This division is derived from a detailed analysis of the nature of information needed for enforcing the various kinds of access control rules presented in Section 2. The rule specific part of an aspect is easier to grasp. It is obvious that the authentication type, pointcut definitions, the constraint to check, and the removal of unauthorized contents are specific to each individual rule. The interesting case is the generic part of the access control aspects. For this, we must look into the inner structure of the access control rules and constraints.

#### 3.2.1 Authentication Aspects

Firstly, the simplest type of access control is indeed user authentication. Here the main concern is how to accommodate different schemes of user authentication. Besides, since all access controls must be preceded by proper user authentication, we shall make the various authentication codes available to all aspects in the framework. This is achieved by defining a root aspect that also acts as the factory of authentication objects. We call this root aspect AAAspect[2] and here is its code:

---

[1] Since Version 1.1, Struts also supports method-based dispatch units through a new class called *DispatchAction*. This can also be covered by our framework with little adaptation.

[2] For spaces' sake, we omit the four arguments captured at the pointcuts (the *execute()* method).

```
public abstract aspect AAAspect {
  abstract pointcut pc(..);// arguments omitted
  abstract protected String getAuthType();
  private static Hashtable authTable = new Hashtable();
  static {
     authTable.put("PWDAuth",PWDAuth.getInstance());
     authTable.put("DCAuth", DCAuth.getInstance());
     … // other authentication schemes }
  protected Authentication getAuthObj(){ //factory method
     return (Authentication)authTable.get(getAuthType());
  }
}
```

The `getAuthObj()` method is the factory method that will return the right authentication object according to the result of executing the `getAuthType()` method, which is rule specific and defined in concrete aspects. For each authentication type, there will be a singleton authentication object containing the code to verify that a user has passed the identification check and the login page name for re-direction purpose in case the user has not. These objects are instantiated from subclasses of the `Authentication` class, which prescribes the interface for verifying authentication checks (i.e., `isAuthenticated()`) and for obtaining the re-direction page (i.e., `forwardTo()`). In the code listing above, we have included two such objects instantiated from class `PWDAuth` and class `DCAuth`.

The user authentication task is a coordinated work between authentication aspects and authentication objects. Authentication aspects are responsible for invoking the methods of right authentication objects to perform the check at right places according to the access control rules. We factor out the common code of authentication aspects into the `Authentication` aspect and leave to its sub-aspects, e.g., `PWDAuthAspect`, the specifications regarding where to do the checking (pointcut) and which type of authentication to use. Here is the code for the `Authentication` aspect:

```
public abstract aspect AuthenticationAspect extends AAAspect{
   ActionForward around(..) : pc(..) { // arguments omitted
     Authentication auth = getAuthObj(); // factory call
     if (auth.isAuthenticated(request))
         return proceed(mapping, form, request, response);
     else // login re-direction
         return mapping.findForward(auth.forwardTo());
   }
}
```

### 3.2.2 Access Control Aspects

The structure of our access control aspects is determined by the constraints in access control rules. Specifically, the availability of the information referenced in the constraints, such as user roles, function arguments and data contents, will shape the structure of our aspect codes. We now explain how this relates to our design as follows.

As described earlier, in Struts-based Web applications, user profile information, user input, and any execution results are all kept in the session object or the request object, hence accessible to the aspect code[3]. The real issue here is *when* the required

[3] The `Cxt` and `App` objects are of global type, so they are always available.

information is available. Previous work [4][5] tacitly assumed that the information is available to the aspect code for making the access control decision before executing the designated function. However, this is not the case for content-based fine-grained access control requirements. For instances, both rule R4 and rule R5 given in Section 2 need to examine the data contents to filter out unauthorized data. Such constraints cannot be enforced without first invoking the function requested.

A close look at the constraint expressions of R4 and R5 reveals that it is the reference to the attributes of the *Data* object that calls for additional data filtering. Indeed, unlike the attributes of the *User* and *Fun* objects, the attributes of the *Data* object are not available before executing the designated function. Therefore, we propose two different kinds of access control aspects for realizing these requirements, namely *pre-checking* and *post-filtering*. The pre-checking aspects will handle the common cases where the constraint involves only the User and *Fun* objects while the post-filtering aspects are needed if the constraint also refers to the attributes of the *Data* object.

**Listing 1.** The PostfilteringCollection aspect

```
public abstract aspect PostfilterCollection extends AAAspect {
  abstract protected Boolean
          constraint(HttpServletRequest request);
  private ActionForward forwardToErrorPage(..){ … };// omitted
  abstract protected boolean filter
             (HttpServletRequest request, Object data);
  abstract protected Collection getRS(HttpRequest request);
  protected abstract void mask(Object data);
  protected void remove(Iterator i) { i.remove();} // a record
  ActionForward around(..) : pc(..) { // arguments omitted
    Authentication auth = getAuthObj();
    if (auth.isAuthenticated(request)) { //authentication check
      if (constraint(request)) { //privilege check
         ActionForward forward = proceed(..); //resume
         Collection col = getRS(request);
         Iterator i = col.iterator();
         while(i.hasNext()){
           Object data = i.next();
           if (!filter(request, data))
              remove(i);    // unauthorized record
           else mask(data); // fields masking
         }
         return forward; // completed
      } else // access denied
         return forwardToErrorPage(request, mapping);
    } else  // login re-direction
         return mapping.findForward(auth.forwardTo());
   }
}
```

Both pre-checking and post-filtering aspects must first get an authentication object and make a call to its `isAuthenticated()` method. This will ensure that access control enforcement is preceded by user authentication. Moreover, the advice in the post-filtering aspects must also conduct an access constraint check before proceeding

to execute the designated function and filter out unauthorized contents[4]. Since pre-checking aspects is the base for post-filtering aspects, we shall only present post-filtering aspects due to the space limitation.

As the code structure for handling a single record retrieved by key-based queries is different from that of handling a collection of data obtained by ad-hoc queries, we further divide the post-filtering aspect into two styles: *single* and *collection*. Listing 1 shows the code of the `PostfilterCollection` aspect. Inside the advice, after performing the authentication and access checks, it resumes executing the intercepted function, retrieves the query results from the request object and iteratively applies the filter and the field mask to each individual data record. The specific definitions of the constraint expression, filter condition and fields mask are left to concrete aspects that inherit it. The code for `PostfilterSingle` is very similar yet simpler, since only one data record is being examined.

Figure 2 shows the whole aspect framework and depicts how it works. In summary, to realize a particular access control rule, one has to define a concrete aspect that inherits from a proper abstract aspect, and provide the definitions of pointcut, constraint method, filter condition, and field masking method, if needed.

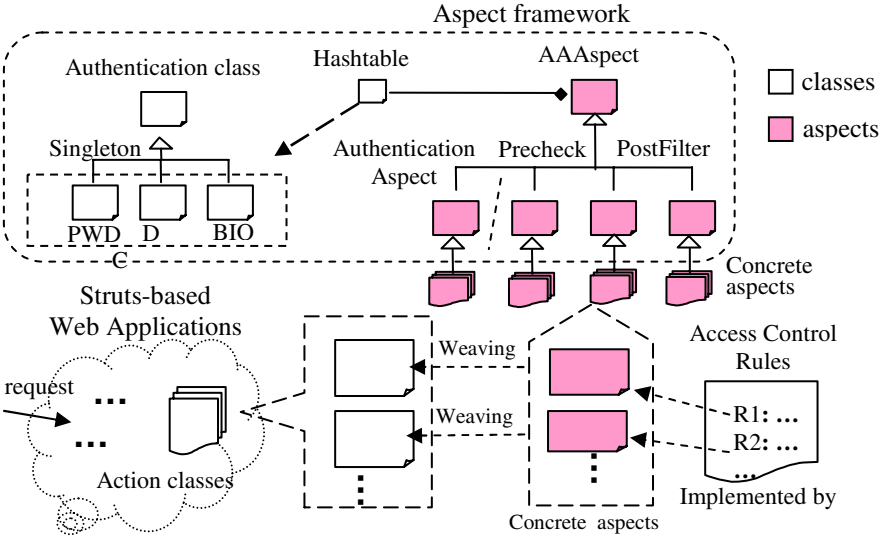



**Fig. 2.** The structure and operation of the aspect framework

## 4 Related Work

Role-based access control (RBAC) [16] is the most often cited guiding principle for application-level security. Since then there are many extended proposals for modeling

[4] This is indeed undesirable code duplication caused by the lack of proper advice reuse mechanisms is AspectJ. But this issue is beyond the scope of this paper.

fine-grained access control. We have role-templates [10], domain-type enforcement [2], content-based [20], team-based [9], and instance-level [11], just to name a few. Basically, they all attempt to base access control constraints on some more detailed relationships among the user, the function requested, and the data to be accessed, as we did here. Context information for access control is investigated in [9] [14] [20].

Our work bears a closer relationship with that of Goodwin et al [11]. First, they used the four-tuple access control rules: [user group, action, resource, relationship], where a set of predefined relationship, as opposed to our general constraint, is defined for each resource type. The major concern is instance-level constraints, no field-level constraint covered, though. Second, they also adopt an MVC-like architecture. The controller intercepts all user commands and queries a centralized manager to make the authorization decision. But they did not consider different authentication types and neither did they use aspect-oriented programming to build their framework.

Applying AOP to security concerns is pioneered by [4] [5]. They also sketched how to build frameworks in AspectJ for handling access control. However, they did not focus on Web applications, and neither did they look into access control modeling in detail as we did. The proposed aspects check the constraint before the attempted access and use per-object stateful aspects. In contrast, we have both pre-checking and post-filtering aspects and use stateless singleton aspects. Hanenberg et al [6] describes some idioms for building framework in AspectJ. Georg et al. [8] studies the use of aspects for modeling security concerns from system analysis perspective.

## 5 Conclusion and Future Work

Security is attracting more and more concerns in the development of Web applications. However, current practices are not capable of supporting a modular implementation of fine-grained access control for Web applications. In this paper, we have presented a systematic approach to model fine-grained access control requirements and an aspect framework for enforcing such requirements for Struts-based Web applications. Our modeling scheme can satisfy a wide range of requirements with different granularity. By employing AOP and framework technology, we have obtained a highly modular implementation of fine-grained access control and achieved a good balance between reuse and customization. Furthermore, the correspondence between access control rules and associated aspects is direct and hence easy to adapt.

A direction we plan to explore is to prepare the access control rules in a configuration file and develop a declarative implementation scheme. Currently, the access control aspects are manually derived from the rules, which require certain programmatic efforts. We shall investigate a translation scheme to generate the concrete aspects automatically from access control rules. This will lead us toward a more declarative implementation and greatly improve the manageability and maintainability of access control requirements.

**Acknowledgements.** This work was supported in part by the National Science Council, Taiwan, R.O.C. under grant number NSC-93-2213-E-004-009.

## References


[1] The Apache Struts Web Application Framework: http://struts.apache.org/

[2] Chandramouli, R.: A Framework for Multiple Authorization Types in a Healthcare Application System, *17th Annual Computer Security Applications Conference*, Dec. 2001.

[3] De Win, B., Piessens, F., Joosen, W. and Verhanneman, T., On the importance of the separation-of-concerns principle in secure software engineering, *Workshop on the Application of Engineering Principles to System Security Design*, 2002.

[4] De Win, B., Vanhaute, B., and De Decker, B., Building Frameworks in AspectJ, ECOOP 2001, *Workshop on Advanced Separation of Concerns*, pp.1-6.

[5] De Win, B., Vanhaute, B. and De Decker, B., Security Through Aspect-Oriented Programming, *Advances in Network and Distributed Systems Security*, Kluwer Academic, pp. 125-138, 2001.

[6] Hanenberg, S. and Schmidmeier, A., Idioms for Building Software Frameworks in AspectJ, 2nd AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS), Boston, MA, March 17, 2003.

[7] Gamma, Helm, Johnson and Vlissides, *Design Patterns.* Addison-Wesley, 1995.

[8] Georg, G., Ray, I., and France, R., Using Aspects to Design a Secure System, *Proc. of the 8th IEEE Int'l Conf. on Engineering of Complex Computer Systems*. December 2002.

[9] Georgiadis, C.K., Mavridis, I., Pangalos, G., and Thomas, R.K.: Flexible Team-based Access Control Using Contexts, *Sixth ACM Symposium on Access Control Models and Technologies (SACMAT 2001)*, Chantilly, VA, USA, May 2001.

[10] Giuri, L., and Iglio, P., Role Templates for Content-Based Access Control, Proceedings, *2nd ACM Workshop on Role-Based Access Control*, Fairfax, VA (October 28–29, 1997), pp. 153-59.

[11] Goodwin, R., Goh, S.F., and Wu, F.Y., Instance-level access control for business-to-business electronic commerce, *IBM System Journal*, vol. 41, no. 2, 2002.

[12] Kiczales, G., Lamping, J., Menhdhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M., and Irwin, J., Aspect-Oriented Programming, in ECOOP '97, LNCS 1241, pp. 220-242.

[13] Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., and Griswold, W.G., Getting Started with AspectJ, *Communications of ACM,* vol. 44, no. 10, pp 59-65, Oct. 2001.

[14] Kouadri Mostefaoui, G., and Brezillon, P., A generic framework for context-based distributed authorizations, In: *Modeling and Using Context (CONTEXT-03), LNAI 2680*, Springer Verlag, pp. 204-217.

[15] Open Web Application Security Project: *The Top Ten Most Critical Web Application Security Vulnerabilities.* http://www.owasp.org/documentation/topten

[16] Sandhu, R., Coyne, E., Feinstein, H., and Youman, C., Role-Based Access Control Models, *IEEE Computer*, 29(2):38–47, 1996.

[17] Sun Microsystems, Java Authentication and Authorization Service (JAAS), http://java.sun.com/products/jaas/index.jsp

[18] Sun Microsystems, JavaServer Pages Technology (JSP): http://java.sun.com/products/jsp/

[19] Sun Microsystems, Java Servlet Technology : http://java.sun.com/products/servlet/

[20] Tzelepi1, S.K., Koukopoulos, D.K., and Pangalos, G.: A flexible Content and Context-based Access Control Model for Multimedia Medical Image Database Systems. *ACM SIGMM Electronic Proceedings*, 2001.

# A Task-Oriented Access Control Model for WfMS

Xu Liao[1], Li Zhang[2], and Stephen C.F. Chan[3]

[1] School of Software, Tsinghua University, Beijing, China
Liaox02@mails.tsinghua.edu.cn
[2] School of Software, Tsinghua University, Beijing , China
lizhang@tsinghua.edu.cn
[3] Department of Computing, The Hong Kong Polytechnic University, Hong Kong
csschan@comp.polyu.edu.hk

**Abstract.** One of the shortcomings of the Role-Based Access Control model (RBAC), used in Workflow Management Systems (WfMS), is that it cannot grant permissions to users dynamically while business processes are being executed., We propose a Take-Oriented Access Control (TOAC) model based on RBAC to remedy this problem. In TOAC, permissions are associated with tasks as well as roles. Users can get permissions through tasks that they carry out in certain processes. And when they are out of processes, permissions can be granted by the roles that they are associated with. Moreover, to facilitate delegation in WfMS, we present a task delegation model which is aim at TOAC.

## 1 Introduction

Workflow management systems are widely used in all kinds of application domains, such as purchase order processes, information integration systems, and e-government systems. Protecting application data in workflow system through apt access control policies has recently been widely discussed.

In 1996, R.Sandhu proposed a series of access control models[1]: $RBAC_0$, $RBAC_1$, $RBAC_2$, $RBAC_3$, and discussed a variety of constraints and policies including role hierarchy and separation of duties (SoD). These models are called the RBAC96 models. The central idea of this model is that access rights are associated with roles, to which users are assigned in order to get appropriate authorizations. It also involves the role hierarchy that enables the permission heritage. Since the roles in enterprises are relatively stable and the number of roles is much smaller then that of users, the work of administrators can be greatly relieved by applying the concept of roles. Thus it is more adaptable to dynamic environments to a certain extent. However, there is no concept of tasks in RBAC, which makes it difficult to satisfy completely the access control requirements in a rapidly-changing dynamic environment.

Other access control models armed at the WfMS environment have also been proposed. For example, ShengLi Wu et al. [9] proposed an authorization and access control model concentrated on the application data in WfMS. The authors emphasized that some types of data were inaccessible or only partially accessible to users. These

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 168 – 177, 2005.

include control data – data maintained by the workflow enactment service to identify the state of individual processes or activity instances or other status information, and workflow relevant data – data used by the WfMS to determine the state transitions of a workflow process instance. Such data usually do not require special access control. They suggested that while workflow application data was application specific and may be used by large numbers of users, their access control is exactly what we need be concerned about.

In this paper, we propose a task-oriented access control model (TOAC), aimed at the requirements in WfMS. The characteristics of this model are:

1. Permissions are authorized both to roles and to tasks
2. Users can get permissions through tasks when they execute a process
3. Users can get permissions through roles when they are out of the processes
4. Delegation between users can be implemented directly through tasks

The rest of this article is organized as follows: Section 2 discusses the access control requirements of a workflow management system in an enterprise environment, focusing on two aspects - dynamic authorization and task delegation. Section 3 illustrates the TOAC model and our proposed solution for the requirements discussed in section 2. Section 4 presents a task delegation model which is applied to our model. Section 5 compares the RBAC and TOAC. Section 6 presents a conclusion.

## 2 Access Control Requirements in WfMS

Processes are initiated and run in enterprises to implement all kinds of tasks. A process contains a set of one or more tasks and activities linked together to realize a business objective or policy goal. The data involved in a process will flow to different users or departments along with the execution of the process. On the one hand, it facilitates information sharing in an enterprise and improves the efficient execution of tasks. On the other hand, it creates new requirements for a more delicate security management model. In this paper, we focus on discussing two aspects of access control requirement in large enterprises.

### 2.1 Dynamic Authorization:

The permissions of participants in a process may change according to the task node that is being executed. Take a typical process as an example, as illustrated in fig 1:

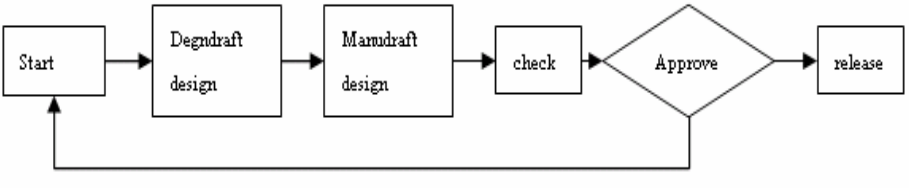



**Fig. 1.** A simple approval process.

It is a simple process widely used in manufacturing enterprises dealing with draft approval, which can be modeled in a workflow management system as a template consisting of five task nodes and one route node.

$T_1$: "start". It is an automatic task node, in which a user can initiate a process.

$T_2$: "Degndraft design". A designer can design and submit a draft for design.

$T_3$: "Manudraft design". A designer can design and submit the draft for manufacture in this task, and he may refer to the draft submitted in $T_2$ but cannot edit or delete it.

$T_4$: "check". An auditor or designer manager can check the submitted draft in this task.

$T_5$: "release". It is the end node of this process and the draft is released to other departments of the enterprise.

$R_1$: Decision is made in this route node. If the draft is approved, it will be released, otherwise it will be rollbacked to start node.

Suppose that a user is assigned the role of "designer", his permission is fixed in different processes or tasks according to the RBAC model. However, in a real situation, a user may need to be assigned different permissions between $T_2$ and $T_3$. Or, a user normally unqualified to access the design draft may be permitted to access the design draft when assigned to $T_4$. Moreover, the permission to access the design draft should be granted just at the beginning of his execution of $T_4$ and should be revoked as soon as he finishes this task. Thus, authorizations should be dynamic and users should be given different privileges while performing different tasks.

### 2.2 Task Delegation

In general, delegation occurs as one active entity in a system delegates its authority to another entity to carry out some function on behalf of the former.

In current workflow management systems, the RBAC model is widely adopted, where system administrators assign roles to users. Though it is more convenient for administers to manage roles than to manage users directly, such complete dependence on administrators will inevitably increase their workload in a large distributed environment, if they are involved in every role assignment. Therefore, delegation is needed to relieve the workload of security administers. For example, if a user cannot complete his accepted tasks because of illness or resignation, his tasks should be delegated to other users to avoid delaying the whole process. Two solutions are commonly adopted: one is managing delegation through an administrative infrastructure outside the direct control of users; the other is that users mediate the delegation to other users directly. In any case, an integrated delegation policy is required to ensure proper operation and to prevent abuses [11].

The rest of this article will propose a task-oriented access control model, and illustrate how this model can meet the requirements discussed above.

## 3 Task-Oriented Access Control Model

### 3.1 Process-Instance Based Datagroup

Various objects are used in processes, including documents, tables and drafts. Since objects may be added into a process for reference and new objects can be created by

users executing the tasks, one can not precisely know beforehand which objects will be used in a process. Moreover, the objects added or created in one task node may be edited or referred by the executors of other subsequent task nodes. Take the process illustrated in fig 1 as an example. Users can create a copy of a design draft and add a relevant document for reference while executing $T_2$. The designer who executes $T_3$ probably needs to read the draft created in $T_2$, but generally he cannot edit or delete the draft created in $T_2$. Thus, it is impossible for an administrator to precisely define the permissions of executors of every task when he customizes the process templates.

This paper proposes the concept of a process instance-based datagroup (or simply called datagroup) in order to manage the data involved in every process instance. A datagroup is a temporary container used to hold the objects and data in a process instance. However, it should be emphasized that an object added in a datagroup is only a reference to the real object in the enterprise. Similarly, the deletion of an object from a datagroup only excludes the reference from this datagroup. There is a many-to-one relationship between datagroups and processes. A datagroup can only belong to one process, while in a process we can define several datagroups. However, it is in the task nodes of this process that each datagroup is operated by users. The relationship between datagroups and task nodes is many to many. A task node can be associated with several datagroups that belong to the process and a datagroup can be used in several task nodes. To make an analogy in the context of programming languages, a datagroup is like a local variable in a process instance, because each datagroup only belongs to one process instance and is unavailable beyond that process instance. However, inside a process instance, a datagroup is like a global variable in the sense that a datagroup can be used in one or more task nodes in the process according to the process template, and the addition or deletion of objects in a datagroup will affect the next usage of the datagroup.

### 3.2 Task-Oriented Access Control Model

In a workflow management system, a process is composed of many tasks which are indivisible atomic tasks or divisible composite tasks. This paper focuses on the latter task type for the reason that most current commercial workflow management systems support composite tasks and process nesting, that is, one task node may embed another process or a set of tasks. So we can abstract a hierarchical relationship between tasks.

The Task-Oriented Access Control model based on RBAC model, datagroup and task is introduced in this section, as illustrated in fig 2. Its entities and relationships can be defined as follows:

U: the set of all users, u denotes a user in the enterprise.

R: the set of all roles, r denotes a role in the role set. There is a hierarchical relationship between roles. For example, a general manager can supervise several department managers, that is, general manager is superior to department managers.

OBJ: the set of all data and objects in system, obj denotes a particular object. The data and objects in an enterprise include files such as all kinds of documents, drafts and tables, as well as hardware and infrastructure.

P (permission): the set of all permissions that can be assigned to users. p denotes an item of permission.

T: the set of all tasks in process, t denotes a task executed by users.

DG (datagroup): the set of all datagoups, dg denotes a datagroup.

RH (Role Hierarchy): RH $\subseteq$ R×R is a partial order on R. ($r_i$, $r_j$) $\in$ RH denotes that $r_i$ is a role superior to $r_j$, as a result, $r_i$ automatically inherits the permissions of $r_j$.

TH (Task Hierarchy): TH $\subseteq$ T×T is a partial order on T. ($t_i$, $t_j$) $\in$TH denotes that $t_i$ is a higher task than $t_j$, that is, $t_i$ includes $t_j$ or equals to it.

URA (User Role Assignment): URA $\subseteq$ U×R is a many to many assignment relation between users and roles. ($u_i$, $r_j$) $\in$ URA denotes that $u_i$ is assigned to role $r_j$, and he will get all the privileges associated with $r_j$.

RPA (Role Permission Assignment): RPA $\subseteq$ R×P is a many to many permission to role assignment relation. (ri, pj) $\in$ PA denotes assignment permission $p_j$ to role $r_i$.

RTA (Role Task Assignment): RTA$\subseteq$R×T is a many to many assignment relation between roles and tasks. ($r_i$, $t_j$) $\in$RTA denotes that the members of role $r_i$ can execute task $t_j$.

DgTA (Datagroup Task Assignment): DgTA $\subseteq$ DG×T is a many to many relationship between datagroups and tasks. ($dg_i$, $t_j$) $\in$ DgTA denotes that datagroup $dg_i$ is associated with task $t_j$.

Function Users: R$\rightarrow 2^U$ is a function mapping each user to role, Users(r)$=$\{u | (u,r) $\in$ URA\}.

Function Participants: T$\rightarrow 2^R$ is a function mapping each task to role, Participants(t)$=$\{r | (r, t) $\in$RTA \}.

In our access control model, authorization can be presented as a 5-tuple: grant = (t, roleset(t), o, p, scope). t $\in$T denotes a task. roleset(t)$=$\{r | (task,r) $\in$RTA\} $\subseteq$Role denotes the minimal role set allowed to execute task t. p $\in$Privilege denotes a kind of permission. Scope denotes the available domain of this item of authorization. The value of scope can be: process, task or datagroup It denotes that the domain of the authorization is the whole process, one of the task nodes and inside of a datagroup respectively. The 5-tuple means that the executors of task t own the privilege to object o while executing the task and the available domain of the authorization is within the defined scope.

In this model, authorization relationships can be presented by the partial order: $g_p > g_t > g_d$, where $g_p$, $g_t$ and $g_d$ respectively denote the authorizations whose scopes are process, task and datagroup. If $g_p$ is available in the whole process, the tasks or datagroups that belong to this process will inherit $g_p$ automatically. It is the same case with $g_t$ whose available domain is one of the tasks in a process. If one of datagroups is not explicitly granted an authorization, it will still inherit the authorizations from the task it is associated with.

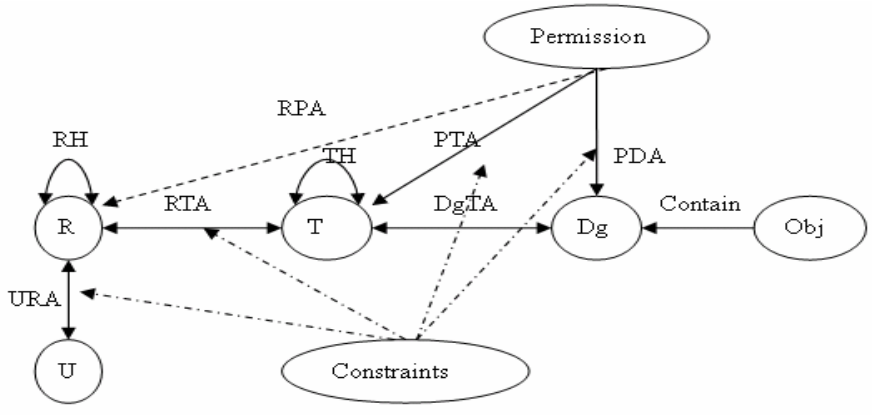



**Fig. 2.** The task-oriented access control model in WfMS

As is shown in fig 2, users can get permissions from their roles as well another channel - tasks. But these two types of permissions are different. The permission gained from a role is static and can be modified only by administrators or security officials. It will not be altered along with the execution of a process. However, the permission gained from a task is dynamic and temporary and may change as the process is executed. The permission will be revoked totally when the process is finished or the task is submitted.

### 3.3 An Example Case

Taking the process described by fig 1 as an example, one can define two datagroups – "design draft datagroup" ($dg_1$) and "manufacture draft datagroup" ($dg_2$), to contain the objects belonging to "design draft" and "manufacture draft" respectively. It is specified that $dg_1$ is associated with task $t_2$ and $t_3$, and $dg_2$ is associated with task $t_2$. In task $t_2$, as the main purpose of this task is to design the "design draft" which is put into $dg_1$, one can specify that the executors of $t_2$ can create, browse, edit and delete the object in $dg_1$. In task $t_3$, users may refer to the objects in $dg_1$ when they design "manufacture draft", but they can not modify those drafts. So the permission of $dg_1$ in task $t_3$ is specified to be "read" or "browse". While, the executors of $t_3$ can create, browse, modify and delete the objects in dg2.

Suppose that a user $U_1$ is assigned the role "designer", he can get all the permissions associated with a designer as derived from the role permission assignment. When he begins to implement task $t_2$, He will own the permissions associated with $t_2$ as well. As datagroup $dg_1$ can be accessed in $t_2$, he can access the objects in dg1, even though they are not created or owned by him. He can create, browse or modify objects in $dg_1$ according to the authorization defined in template. If it is permitted, he can also add the references of some existing documents to $dg_1$ so that the executors of latter tasks can

access the newly added objects if $dg_1$ is associated with that task. However, all his permissions gotten from tasks will be revoked after he submits task $t_2$. When the process is completed, all the permissions granted to this process will be unavailable, and the relevant data will be stored into the database as historical data to create log files.

## 4 Task Delegation

Different types of delegation models, including user to machine, user to user, and machine to machine delegation, have been proposed [10~12]. The model that we are proposing focuses on user to user delegation.

R. Sandhu proposed a role based delegation model (RBDM0) [12], after the RBAC96 model. In RBDM0, role is introduced at the first time and some problems such as hierarchical roles, partial delegation and delegation step are also discussed. Based on RBMD0, L. Zhang and G. Ahn proposed the RDM2000 model [10,11], which focused on the relationship between components of a delegation. Furthermore, they also proposed a rule-based framework for role-based delegation, and a rule-based specification language to describe their model.

The delegation models listed above are based on roles and not easily adaptable to workflow environments. In WfMS, users usually require the capability to delegate the tasks to other users directly. As permissions can be associated with the tasks in the task-oriented access control model, it is very easy for a user to delegate his tasks to another user, and the permission will be transferred along with the tasks delegated.

### 4.1 Basic Elements in Task Delegation Model

As the introduction of delegation mechanism, User Task Assignment (UTA) can be extended into two versions. One is the original UTA, the tasks originally assigned by WfMS. The other is delegated UTA, the tasks delegated by other users.

UTA (User Task Assignment): $UTA \subseteq U \times T$ is a many to many assignment relation from user to task. $(u_i, t_j) \in UTA$ denotes that user $u_i$ can execute task $t_j$.

Function Tasks: $U \rightarrow 2^T$ is a function that map each user to tasks he executes. $Tasks(u) = \{ t \mid (u, t) \in UTA \}$.

$UTAO \subseteq U \times T$ is a many to many original user to task assignment relation

$UTAD \subseteq U \times T$ is a many to many delegated user to task assignment relation

$UTA = UTAO \cup UTAD$.

Task_o(u) denotes the original task set assigned to user u. $Tasks\_o(u) = \{t \mid (u,t) \in UTAO \}$

Task_d(u) denotes the delegated task set assigned to user u. $Tasks\_d(u) = \{t \mid (u,t) \in UTAD\}$

A new relationship – Delegation Relation (DLR), is defined, in order to denote the relationship among components in a delegation. There are four components in a user to user delegation, which include delegating user, delegating task, delegated user, and

delegated task. For example, $(u_1,t_1,u_2,t_2)$ means user $u_1$ - the original assigned user of task $t_1$, delegates task $t_2$ to user $u_2$, where $t_1 \rightrightarrows t_2$ or $(t_1,t_2) \in TH$, that is, $t_1$ equals or contains $t_2$.

DLR can be further differentiated into two subtypes: Original Delegation Relation (ODLR) and Delegated Delegation Relation (DDLR). For example, user u1 is assigned task t1 by WfMS, and then $u_1$ delegates it to another user $u_2$ - the original delegation. If $u_2$ then delegates $t_1$ to another user $u_3$, then this is delegated delegation.

Based on the conception above, we can define the entities:

DLR $\subseteq$ UTA×UTA=U×T×U×T denotes a many-to-one task delegation relationship.
ODLR $\subseteq$ UTAO×UTAD is an original task delegation relationship.
DDLR $\subseteq$ UTAD×UTAD is an delegated task delegation relationship.
DLR=ODLR $\cup$ DDLR.

### 4.2 Task Delegation and Revocation Rule

When a task is delegated to another user, constraints are imposed to prevent the delegation from violating the security policies. So, we define the Constraints (C):

Constraint C is a Boolean expression composed of operator '&' (and), 'I' (or). It can constrain the tasks or users to be delegated. For example, $c = (-u_1) \mid (u_2)$ means a constraint that indicate the delegated user should not be $u_1$ or should be $u_2$.

The delegation rule can be defined:

can_delegate $\subseteq$ T×C×N,

where T denotes task set, C denotes constraint set, N denotes delegation step. (t, c, n) $\in$ can_delegate means the assigned user of task t can delegate t to another user, satisfying constraint c and without exceeding the max delegation step n.

The revocation rule is an important part of the delegation mechanism. For example, if task $t_1$ was delegated to $u_2$ by original assigned user $u_1$, when $u_2$ abuses the permission delegated to him or executes $t_2$ by error, both $u_1$ and the administrator should be able to revoke the delegated task.

Task revocation rule can be defined as:
can_revoke $\subseteq$ T

where T denotes the task set that can be revoked. For example, $t_1 \square$ can_revoke means task $t_1$ can be revoked.

## 5 Comparison Between RBAC96 and TOAC

The task-oriented access control model was compared with the RBAC96 model proposed by Sandhu, widely accepted as the main RBAC model. Results are summarized below:

**Table 1.** Comparison between RBAC96 and TOAC

| Criteria: | RBAC96 | TOAC |
|---|---|---|
| 1 supporting roles | Y | Y |
| 2 supporting tasks | N | Y |
| 3 permitting access control by users | Y | Y |
| 4 permitting access control by tasks | N | Y |
| 5 supporting role hierarchy | Y | Y |
| 6 supporting authority inherence | Y | Y |
| 7 supporting SoD constraint | Y | N/A |
| 8 supporting delegation | N/A | Y |
| 9 supporting passive access control | Y | Y |
| 10 supporting active access control | N | Y |

Compared with the RBAC model in which users get permissions through roles, the TOAC model also permits users to get permissions from tasks. Because the permission can vary with the tasks and processes, the TOAC model can support active access control successfully. Moreover, users are granted privileges just at the start of tasks, which are revoked as soon as the tasks are finished. So privilege leakage caused by granting permission too early or revoking permission too late can be avoided. And the process deadlock caused by the lack of sufficient permissions can also be minimized. Furthermore, users can be granted different permissions in different task nodes, as permissions are associated with tasks.

## 6 Conclusion

Enormous amounts of information in enterprises flow along processes and are shared by many different users. There security must be assured. In this paper, we firstly analyze the relevant requirements in workflow management systems. Then, based on the RBAC model, we propose the task-oriented access control model. This model can grant authorizations to tasks directly and that process-instance based datagroup is introduced to facilitate the authorization. The partial order and permission inheritance between the three-layer authorization – process, activity and datagroup is used to enhance authorization. The model also implements task delegation.

## References

1. R. Sandhu, E. Coyne, H. Feinstein, C. Youman.. Role-Based Access Control Models. IEEE Computer, Feb 1996, Vol.29, No.2; 38~47.
2. R. Sandhu, P. Samarati, Access Control: Principles and Practice .(http://www.list.gmu.edu / journal_papers.htm). Sept 1994.

3. G. H. Ahn, R. Sandhu, Role-Based Authorization Constraints Specification. ACM Transactions on Information and System Security, Vol. 3, No. 4, November 2000; 207~226.
4. D. F. Ferraiolo, R. Sandhu, S. Gavrila. Proposed NIST Standard for Role-Based Access Control, ACM Transactions on Information and System Security, August 2001, Vol.4, No.3; 224~274.
5. V.Atluri, W. K. Huang. An Authorization Model for Workflow, Proceedings of the Fourth European Symposium on Research in Computer Security, Sep 1996; 44~64.
6. V.A. Atluri, A. Gal. An Authorization Model for Temporal and Derived Data: Securing Information Portals. ACM Transactions on Information and System Security, Vol. 5, No. 1, February 2002; 62~94.
7. S. Castano, F. Casati, M. Fugini. Managing Workflow Authorization Constraints through Active Database Technology. Information Systems Frontiers, Special Issue on Workflow Automation And Business Process Integration, 2001; 319~338.
8. E. Bertino, E. Ferrari, V. Atluri. The Specification and Enforcement of Authorization Constraints in Workflow Management Systems, ACM Transactions on Information and system Security, Feb 19999, Vol.2, No.1; 65~104.
9. S. WU, A. Sheth, J. Miller. Authorization and Access Control of Application Data in Workflow Systems. Journal of Intelligent Information Systems, 18, 2002; 71-94.
10. L. H. Zhang, G. J. Ahn, B. T. Chu. A Rule-Based Framework for Role-Based Delegation and Revocation. ACM Transaction on Information and System Security, Vol.6,No.3, Aug 2000; 404~441.
11. L. H. Zhang, G. J. Ahn, B. T. Chu. A Rule-Based Framework for Role-Based Delegation. Proceedings of the sixth ACM symposium on Access control models and technologies, May 2001; 153 -~162.
12. E. Barka, R. Sandhu. Framework for Role-based Delegation Model. Proceeding of 23th National Information Systems Security Conference, Oct 2000.

# A Brief Observation-Centric Analysis on Anomaly-Based Intrusion Detection

Zonghua Zhang and Hong Shen

Graduate School of Information Science,
Japan Advanced Institute of Science and Technology,
1-1 Tatsunokuchi, Ishikwa, 923-1292, Japan
Tel: 81-761-51-1209, Fax: 81-761-51-1149,

**Abstract.** This paper is focused on the analysis of the anomaly-based intrusion detectors' operational capabilities and drawbacks, from the perspective of their operating environments, instead of the schemes per se. Based on the similarity with the induction problem, anomaly detection is cast in a statistical framework for describing their general anticipated behaviors. Several key problems and corresponding potential solutions about the normality characterization for the observable subjects from hosts and networks are addressed respectively, together with the case studies of several representative detection models. Anomaly detectors' evaluation are also discussed briefly based on some existing achievements. Careful analysis shows that the fundamental understanding of the operating environments is the essential stage in the process of establishing an effective anomaly detection model, which therefore worth insightful exploration, especially when we face the dilemma between the detection performance and the computational cost.[1]

## 1 Introduction

Anomaly-based intrusion detection is to discern malicious behavior patterns from the regular ones in the variables that characterize computer systems. The basic assumption for anomaly detection is that the intrinsic characteristic or regularity of the observable subjects deviate significantly from that of intrusive anomalies, therefore, the preprocess and analysis of the operating environment, which is composed of specific observations, is an initial but important stage for the modeling of anomaly detectors (ADs). However, due to the increasing complexity of modern computer systems and the diverse nature of the network, it is generally agreed that there is no such thing as a typical and perfect "system normality description". A possible way, which is also the trend of current anomaly detection research, is to develop methods for characterizing a given operating environment sufficiently well so that optimal ADs for that environment can be designed. The effort that must be paid of such work is to allow the limits of ADs, in terms of

[1] Short version.

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 178–191, 2005.

expected false alarm rate, to be predicted. Although many attacks can be identified by ADs, unperfect description of the normality and the novel legitimate activities make them suffer from an uncontrollable false alarm rate. Furthermore, most of existing ADs pay more attention to the techniques themselves, rather than the fundamental understanding of the working field, which restricts them to a broader application. In addition, the evaluation of ADs is deficient and inconvincible due to the limits of the so-called benchmark dataset, especially for the researches that have been focusing on a specific method for a particular operating environment, which built based solely on "expert" knowledge. Our work aims to explore the fundamental attributes of some observable subjects, and analyze several typical ADs' operating environments, in general, including: Cast the anomaly detection in a statistical framework, and characterize ADs' behaviors in a general way; Give a critical analysis on some ADs' operating environments, as well as their operational limits; And then conclude the current evaluation schemes, and propose our own idea for better measurement metric.

The rest of paper is organized as follows. Section 2 establishes a statistical framework to describe the ADs' general behavior. Section 3 gives a general characterization of the selected observation normality. Section 4 has some case studies. In section 5, we propose our scheme for better evaluation metrics based on some former contributions. Section 6 gives a general discussion.

## 2 A General Statistical Description

From the functional perspective, an AD can be roughly regarded as a simple kind of inductive inference system. In this system, an incoming observation $O_i$ is regarded as a "question", while the normal behavior model $M$ that stored in the memory is regarded as "answers". Given a new $O_i$, the system tries to find an appropriate answer $M_i$ so that $\mathrm{AD}(O_i) \Rightarrow M_i$. The aim of an AD design is to look for effective "answers" that have the highest *a priori* - that has "accurate descriptions". In generating such answers, some primitive normal behaviors have to be previously defined. From probabilistic prediction we can gradually to deterministic prediction, that is, whether the current "question" is an anomalous behavior. Due to the fact that the sample size of "$M$" is limited but the number of questions "$Q$" are infinite and long-standing, the ken and adaptability of AD is a key to answer diverse questions successfully. A general statistical framework can be utilized to describe the AD's behavior:

***Notations:***
$H(t)$: a hidden stochastic process which maps the activities of legitimate users and attackers to a finite space $S$ in terms of discrete time step "$t$"; at time step $t$, if $H(t) = 0$, means legitimate user traces is generated, if $H(t) = 1$, means attacker traces is generated.
$h(x)$: a hidden stochastic process for generating event $x$.
$O_t$: Observation that captured at time step $t$, its meaning varies on the specific detection schemes, and its generation is governed by the hidden process $H$;
$Set(O_t, w)$: a set of observations $O_i$ within window $w$ at time step $t$.

$N(t)$: a legitimate stochastic process that is generated at time unit $t$, $H(t) = 0$;
$M(t)$: a malicious stochastic process that is generated at time unit $t$, $H(t) = 1$;

What an AD cares is the current observation $O_t$, a pair of probability distribution therefore can be considered as follows:

$Pr\{O_t|H(t) = 1, O_{t-1}O_{t-2}...O_1, t\}$, and $Pr\{O_t|H(t) = 0, O_{t-1}O_{t-2}...O_1, t\}$.

if the property of the observation sequence $\{O_{t-1}O_{t-2}...O_1\}$, are not taken into account, the above two probability distribution can be generalized as follows:

$Pr\{O_t|H(t) = 1, Set(O_t, w), t\}$, and $Pr\{O_t|H(t) = 0, Set(O_t, w), t\}$.

as the description in [5], a posterior probability of anomaly intrusion detection can be given as:

$$Pr\{H(t) = 1|O_t, Set(O_t, w), t\} = \frac{Pr\{O_t|H(t) = 1, Set(O_t, w), t\} \cdot (1 - \lambda)}{Pr\{O_t, Set(O_t, w), t\}} \quad (1)$$

where $\lambda = Pr\{H(t) = 0, Set(O_t, w), t\}$. As $Pr\{O_t, Set(O_t, w), t\} = Pr\{O_t|H(t) = 0, Set(O_t, w), t\} \cdot \lambda + Pr\{O_t|H(t) = 1, Set(O_t, w), t\} \cdot (1 - \lambda)$, equation (1) can be simplified as:

$$Pr\{H(t) = 1|O_t, Set(O_t, w), t\} = \frac{c \cdot (1 - \lambda)}{c \cdot (1 - \lambda) + \lambda} \quad (2)$$

which represents *a priori* probability of the legitimate pattern which contains $w$ consecutive observations that have been generated by $h(x)$, and an unknown constant $c = Pr\{O_t|H(t) = 1, Set(O_t, w), t\}/Pr\{O_t|H(t) = 0, Set(O_t, w), t\}$, for equation (2), $Pr\{H(t) = 1|O_t, Set(O_t, w), t\} > \alpha$ iff $c > \alpha\lambda/(1-\alpha)(1-\lambda)$. Thus it is easy to find that the performance of ADs are related directly with the value of $Pr\{H(t) = 1|O_t, Set(O_t, w), t\}$, and it increases with the value of $c$. Based on the equation, a simple intrusion detection model can be defined as:

$$\tilde{ID}(O_t) = c, \quad or \quad ID(O_t) = \begin{cases} 0 \text{ if } \tilde{ID}(O_t) < \alpha \\ 1 \; otherwise \end{cases} \quad (3)$$

Obviously, due to the lack of prior knowledge about $\lambda$, or $c$, it is almost impossible to carry the detection model in (3) into practice directly. Moreover, a good estimate of $\lambda$ and a fundamental understanding of distributions of the processes $N(t)$ and $M(t)$, which we call system normality, are not readily available, which make the detection task deem to be $NP$-hard. The ultimate goal of the anomaly detection model, intuitively, is to characterize the observation normality perfectly well.

### 2.1 Frequency-Based Analysis

Assume that at time instant $t$, an AD observes a set of events $e_1, e_2, ...e_n$, which is generated by H(t). The frequency of those events (strings of symbols) $F(e_1, e_2, ...e_n)$ can be taken as a measurement to characterize the system normality, i.e., $O_t = F(e_1, e_2, ...e_n)$. Given a new observation at time $t$, $O_t$, what

is the probability that it belongs to the set $[O_i]$? A well fitting AD with good description for the known set of events is expected. The universal distribution [13] gives a criterion for goodness of fit of such description. According to the equation (3), the universal distribution $D_{\tilde{AD}}$ for AD, $\tilde{AD}$ can be regarded as a weighted sum of all finitely describable probability measures on finite events: $D_{\tilde{AD}}([O_i]) = \sum_j \beta_j \prod_{i=1}^{t} p_j(O_i)$, where $t$ is the time step representing the number of available observation set $[O_i]$, $\beta_j$ can be taken as the weight of the $j^{th}$ probability distribution on finite observations, and its definition based on the particular detection model, for example, for an AD using string match method, $\beta_j = 1$, if ongoing events match the exact pattern $\phi$ that stored in normal pattern set $\Phi$. Suppose that $[O_i], i = 1, 2, ...t$ is a set of $t$ observations generated by stochastic process $h(x)$, the probability that $D_{\tilde{AD}}([O_i])$ assigns to a new observation $O_{t+1}$ is $Pr(O_{t+1}) = D_{\tilde{AD}}([O_i] \bigcup O_{t+1}) / D_{\tilde{AD}}([O_i])$. The probability assigned to $[O_i]$ by stochastic generator $h(x)$ is $h([O_i]) = \prod_{i=1}^{t} h(O_i)$.

For an ideal AD, given a suitable set of observations $[O_i]$ that used for characterizing system normality, the probability assigned by $D_{\tilde{AD}}$ by $Pr(\cdot)$ should be very close to those generated by hidden stochastic process $h(x)$, that is, a maximal prior information an AD can posses is the exact knowledge of $\lambda$, but in many cases the true generating process $h(\cdot)$ is not known, what we expect is that an AD based on $D(\cdot)$ performs well with small expected errors between $D(\cdot)$ and $\lambda$. For such two probability distributions on finite number of observations, a corollary derived from Hutter [8] can be given as:

**Corollary 1.** *The expected value of the sum of the squares of the differences in probabilities assigned by the stochastic generator $h(\cdot)$, and anomaly detector $D(\cdot)$ to the elements of the observation are less than a certain value, and the expected error in probability estimate might decreases rapidly with growing size of the normal data set.*

The corollary guarantees theoretically that predictions based on $D(\cdot)$ are asymptotically as good as predictions based on $\lambda$ with rapid convergence. Any *a priori* information that can be insert into $D(\cdot)$ to obtain less errors, and we believe that if all of the needed *a priori* information is put into $D(\cdot)$, then $Pr(\cdot)$ is likely to be the best probability estimate possible to $h(\cdot)$, and thus anomaly detector could achieve one hundred percent accuracy.

### 2.2 Sequence-Based Analysis

In many cases, the ordering property rather than the frequency property dominates the characteristic of observable subjects, the pattern of $Set(O_t, w)$ rather than the individual event $O_t$ is thus of potential interest. Similarly, the estimation of $Pr\{O_t | H(t) = 1, Set(O_t, w), t\}$ and $Pr\{O_t | H(t) = 0, Set(O_t, w), t\}$ can also be roughly considered as a simple inductive inference problem: *Given a string $O_{<t}$ (denote $O_1, O_2, ...O_{t-1}$), take a guess at its continuation $O_t$.* Specially, the generation of the event sequence $O_1, O_2, ...O_{t-1}$ is governed by a hidden stochastic process $h(\cdot)$, and $\mu$ is unknown probability distribution for taking $O_t$ at particular time instant $t$ based on the available event $O_1, O_2, ...O_{t-1}$, i.e.

$\mu(O_t|O_{<t})$, while $\rho$ is a guess probability distribution close to $\mu$ or converges, in a sense, to $\mu$, and we expect that an AD based on $\rho$ performs well. Assume $P := \{p_1, p_2, ...p_n\}$ is a countable set of candidate probability distributions on event sequences, a universal probability distribution $\pi$ hence can be defined as: $\pi(O_{1:t}) := \sum_{p\in P} w_p p(O_{1:t}), \sum_{p\in P} w_p = 1, w_p > 0$. $P$ is known and might contain the true distribution $\mu = p_i$ if $P$ is sufficiently large or with well characterization. Based on those assumptions, two corollaries therefore can be deduced from theorems of [8] as follows for modeling anomaly detection models:

**Corollary 2.** *Convergence: Assume anomaly detector observe a sequence $O_1O_2...$ over a finite space $S$ drawn with probability $\mu(O_{1:n})$ for the first $n$ events. The universal conditional probability $\pi(O_t|O_{<t})$ of the next symbols $O_t$ given $O_{<t}$ is related to the true conditional probability $\mu(O_t|O_{<t})$ in the following way:*

$$\sum_{t=1}^{n} E_{<t} \sum_{O_t} (\mu(O_t|O_{<t}) - \pi(\mu(O_t|O_{<t}))^2 \leq \ln w_\mu^{-1}$$

*where $E_{<t}[..] := \sum_{x<t\in P^{t-1}} \mu(x_{<t})[..]$ is the expectation and $w_\mu$ is the weight (4) of $\mu$ in $\pi$.*

which shows that the predication accuracy of anticipated ADs are asymptotically as good as predications based on the stochastic generator $h(\cdot)$ with rapid convergence. However, in practice, ongoing observation might not have exact matching pattern in $P$, i.e., $\mu \notin P$, in such case, a "nearby" distribution $\hat{\mu}$ with weight $w(\hat{\mu})$ is expected, and the distance between $\hat{\mu}$ and $\mu$ is bounded by a constant. The convergence of ADs determines the amount of training time or data required to have a stable model, and the AD converges well when most of the "anticipated" patterns appear repeatedly and are extracted well.

## 3 Normality Characterization of Observable Subjects

This section is focused on the system normality characterization, which is based on a natural taxonomy in accordance with the computer networks components, namely, hosts in the networks and the communication links among the hosts. From a high level view, several criteria to the selection of observable subjects need consideration, including *availability, tangibility, operability*, and *sensitivity.*

### 3.1 Observational Normality of Hosts

A great number of variables could be employed to characterize the state of a host, such as command line strings, system call traces [3], audit events[16], call stacks, resource consumption patterns, etc, and all of them could be encompassed into the framework we established in the last section. However, in fact, the normal behavior of many variables has no obvious pattern, which would be taken as "noise" of "normality". Burgess et al. [1] ever gave a careful analysis of the computer system normality, according to their definition, the system can be distinguished as three scales:

- *Microscopic,* details exact mechanisms at the level of atomic operations, such as the individual system calls and other atomic transactions in operating systems (in terms of *milliseconds*).
- *Mesoscopic,* looks at small conglomerations of microscopic processes and examines them in isolation, such as the individual process or session, or a group of processes executed by one program (in terms of *seconds*).
- *Macroscopic,* concerns the long-term average behavior of the whole system, such as the periodical activities of the users and their corresponding resources consuming patterns.

All the host subjects fall into these three categories, and can be taken as the objects for ADs, whether it aims to look for suspicious patterns or attempts to identify the values that deviate from the acceptable distribution of values. But actually, most of the available host-based ADs take subjects at *mesoscopic* level due to their better controllable properties to characterize system normality. For instance, Forrest et al. [3, 6] ever proposed an immunological detection model by analyzing system calls sequences, which focus on the *mesoscopic* level of UNIX operating system, and some subsequent independent works also take system calls sequences as observable subjects. Therefore, the motivation to analyze the normality of the mesoscopic scale is obvious, that is, why system calls sequences can be selected as observation? What properties these sequences have? Whether the regularity of such computing environment benefits the anomaly detection? Actually, Forrest et al. [3] has given an satisfied answer for the first question, but for the last two questions, there has not satisfied answers so far.

Most the work took the name of the system calls as the observable (other parameters passed to the system calls are ignored), after sequence is established, namely, $(s_1, s_2, ...s_l)$, detection methods such as Enumerating Sequences, Frequency-based methods, Data mining techniques, HMM, or some text categorization methods were applied to identify anomalies. Lee et al. [9] showed that additional information to the sequence elements would improve detection performance without considering the trade-off between detection accuracy and computational cost. They gave an analysis on the regularity of these objects using information-theoretic measures, such as entropy, conditional entropy, information gain and information cost, which gives us a good clue for the characterization of the system normality, and provides us some fundamental understanding about the regularity of computing environment that the ADs work.

To measure the computer system normality from a macroscopic level, Burgess et al. [1] applied a scaling transformation to the measured data, and the distribution of fluctuations about the mean was approximated by a steady-state, maximum-entropy distribution with modulation by a periodic variation. However, due to its approximate nature, any attacks with normal pattern appearance are difficult to be identified based on such model, moreover, what information are required or effective for detecting anomalies need further exploration, and it heavily depends on what will we do once anomalies have been discovered. Intuitively, those observable subjects' normality at mesoscopic and macroscopic scales can be combined to achieve a better performance, macroscopic

normality is used to monitoring the variant of system coarsely, while mesoscopic give doubtful activities further analysis and fine-grain characterization.

### 3.2 Observational Normality of Networks

Due to the diverse nature of the computer networks, it is almost impossible to establish an ideal mathematical model with perfect characterization of the normality of observable subjects, i.e. network packets, nor it is easy to design efficient intrusion detection techniques for networking. However, this does not hold only for intrusion detection, but more or less for other fields, such as traffic modelling and analysis, the fundamental understanding of basic protocol behavior is a possible way to go. In addition, due to the inherent limitations of the available IDSs and the increasing application of encryption in communication, such as IPSec, SSL, intrusion detection and prevention have once again moved back to the host systems.

So far, *tcpdump data* has been widely applied to detect attacks from the protocol scale (connection behavior). Generally, each record describes a connection using several features: timestamp, duration, source port, source host, source bytes (outbound bytes from source to destination), destination port, protocol type(TCP, UDP, ICMP or others), destination host, destination bytes (inbound bytes from source to destination), and flag. Due to the huge data amount generation everyday and the transient nature, it is really difficult to describe the system normality in details, and therefore simplification and preprocess is needed. For instance, some techniques for online analysis of continuous stream could help us to capture the transient nature of network subjects [4, 2], while some network traffic modeling methods [10] facilitate us to monitor and obtain the necessary information for measuring network normality at a macroscopic level. A brief framework for measuring network normality can be simply constructed through a top-down procedure shown as figure 1.

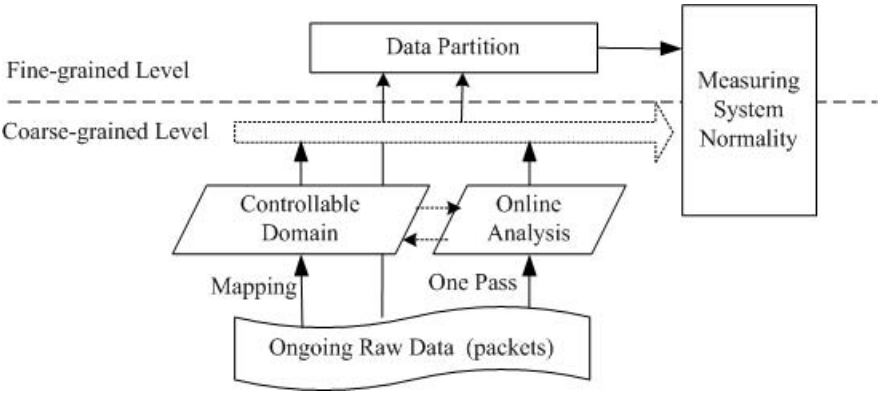



**Fig. 1.** A simple framework for measuring network normality

1. *Coarse-grained Level:*
   - Mapping network traffic into wavelet domain to discover the periodicity of the specific network activities, which can disclose the sudden system collapse and unrhythmic activities;
   - Sketch-based techniques and clustering methods are applied to a certain doubtful time-scale (or a periodicity) to have further insightful investigation.
2. *Fine-grained Level:*
   - Information-theoretic measures are used to divide the processed network data from coarse-grained level into more "pure" data sets with higher regularity;
   - Building anomaly detection models based on the characterization of system normality.

Actually, collection and monitoring of network observable subjects in a discrete way rather than a continuous way may not deteriorate the performance [1]. From the point of view of the observable subjects, we envision a framework in which several levels of data analysis are used as the basis to be combined to yield a single but effective system normality characterization. We envision further an approach in which anomaly detection models are built on the fundamental understanding of their operating environments, and have the adaptability in response to changing situation. The hope is that a collection of simple, elaborate surrogates based on specific observable subjects can evolve into generic models without performance deterioration.

## 4 Case Studies

This section takes several representative ADs (frequency-based and sequence-based) as instances to inspect their detection coverage from the perspective of their respective operating environments.

### 4.1 STIDE Detector

The detailed detection scheme can be found in [3]. The main idea is: STID acquires normal system call sequences by sliding a window of size '$w$' over the training data, and stores those sequences in a normal profile. The detector scans the testing sequences using the $w-$sized window, and notes the number of mismatches between sequences from the normal profile and the test data. The number of mismatches occurring within a temporally local region (Locality Frame Count) is used to characterize the anomaly signal and determine the extent to which the testing sequences are anomalous. The algorithm is easy and effective, some more sophisticated models do not have significant performance improvement over the original model [15].

In the original work, the sliding window of the stide detector always selects 6, Lee et al. [9] ever gave an analysis using conditional entropy to explain the selection of the magic number, but Tan et al. [14] undermined their method using

a random data set, furthermore, they gave a thorough analysis on the selection of detector window using a synthetic data set. Actually, this phenomena depends heavily on its operating environment, and the detector essentially works in an exhaustive way, its performance thus is effected by the normal data set, any foreign elements or sequences that unincorporated in the normal data set would be detected easily.

### 4.2 Cross Entropy-Based Anomaly Detector

Based on the assumption that the occurrence frequencies of different observable subjects can be measured during a certain time scale, a probability distribution can be used to represent the occurrence pattern during this period. Obviously, observable subjects are considered without sequential nature, it thus belongs to a static method [17]. We do not intend to undermine the contribution of their work, and we only want to point out that a careful analysis on the operating environments could also get the same conclusion as that from expensive trial-and-error experiments.

The detector operated with two kinds of observable subjects in the original work, one is program profiles based on Unix system calls, another is user profiles based on Unix shell commands. As we know, system calls executed by the same process have certain temporal pattern, namely, system calls from a specific process have the sequential correlation, at least the order between several system calls always keep unchanging. While for the shell command data, although individual user has a particular pattern during his/her login session, that is, the token was recorded almost always keep the same entropy, the frequency of tokens rather than the sequential relations have more contribution to the characterization of user behavior. Under such cases, the ADs which can capture temporal characteristics, such as HMM-based AD, obviously have better performance in the system calls data set than that of in the shell command data set. On the contrary, frequency distributions-based AD have the inverse performance due to the properties of operating environment.

### 4.3 Probabilistic Anomaly Detectors

Ye et al. [16] have given a nearly thorough analysis on the probabilistic techniques-based ADs with computer audit data, including decision tree, Hotelling's $T^2$ test, chi-square multivariate test and Markov chain. What we concern is the basic data model that all those probabilistic ADs applied. In the model, audit events are represented as frequency distribution $(X_1, X_2, X_3, ...X_N)$, the value of $X_i$ was computed by a exponentially weighted moving average method (EWMA).

From the standpoint of the observable subjects, two aspects of the data modelling worth insightful consideration, i.e., the parameters selection, and the correlation among data points. In the original model, after a certain period, the weights drops close to zero, but the speed is different due to the various value of a parameter $c$. Although we do not expect that some unknown $c$ could improve the modelling performance dramatically, a comparative study worth doing to insight

the impacts of different values, and thus select one for better modelling. Furthermore, $c$ might vary in different situation, due to the drifting of system normality, a constant value thus can hardly characterize all the normal activities well. In addition, in the original data model, only the audit event type was considered, while other attributes, such as user ID, process ID, session ID, the system object accessed, were omitted. To incorporate those necessary additional information, a multivariate EWMA [7] can be used as $X_i(t) = C * O_i(t) + (1 - C) * X_i(t - 1)$, where $X_i(t)$ is the $i$th EWMA vector, $O_i(t)$ is the $i$th observation vector at time $t$, $i = 1, 2, 3...n$, $C$ is the diag $(c_1, c_2, ...c_p)$ which is a diagonal matrix with $c_1, c_2, ...c_p$ on the main diagonal, and $p$ is the number of variables, i.e., the number of attributes that we are considering. The MEWMA model takes into account all the necessary variables of audit events, and thus can be used to capture the process shift in multi-scales. Although it is much more complex than the univariate EWMA, a better performance is expected to be achieved if some scalability problems are solved well.

As we have analyzed above, all the ADs have their own detection coverage and blind spots, and their detection capability vary on different working environments. In essence, the statistical modelling in section 2 facilitates the comparison between those ADs, in terms of their general expected behaviors. Several additional aspects are compared in table 1, where $N$ (the same mathematical form does not mean the same meaning) is the size of normal data set that has been constructed in a particular form, while $L$ is the size of ongoing trace being detected. It is worth noting that the detection cost of STID can be reduced to $L * logN$, if normal data are stored in an effect form, i.e., forest of trees. The detection cost of probabilistic detectors are different on specific techniques, for instance, $Hotelling's\ T^2$ requires a large memory to store the variance-covariance matrix and much time to compute the matrix multiplication and inverse, its time complexity for detection nearly $O(N^2)$ $(L << N)$, while *Markov chains* or *chi-squre* multivariate test need less computational overhead, i.e., $O(N)$ or so.

Although the original ADs have their own operating environments. Careful analysis facilitate them to be extended to a broader application field. For

**Table 1.** A Comparison between Three Representative Anomaly Detectors

| Anomaly Detector | | Observation | Main Property | | Detection Cost |
|---|---|---|---|---|---|
| | | | Frequency | Ordering | |
| STID | | System calls | | √ | $N * (L - w + 1)$ |
| MCE | | System calls/ User commands | √ | | $N * L$ |
| Probabilistic Detectors | Markov Chain | Audit Events | √ | √ | $N * L$ |
| | Hotelling's $T^2$ | | | | $N^2 * L$ |
| | Chi-square | | | | $N * L$ |

example, STIDE was originally developed with system calls of privileged programs, but it can also be applied to audit events provided the scope of activities is not so wide, based on the similar properties of those two observations. Similarly, the probabilistic anomaly detectors that were originally operated with audit events and shell command lines can also be extended to system calls, if enough ordering property are included during the data modelling. Among those detectors, STID has the highest detection capability in general case, because it stores all the unique system calls sequences in the normal profile. Any ongoing traces with system call sequences that never appeared in normal profile will be detected as anomalies (determined by LFC). According to Corollary 2, STID has a good convergence due to the high average value of $w_\mu$. Generally, two elements contribute to the higher detection capability of STID:

- Observation, i.e., system calls. As we know, system calls of privilege processes is a good level to reflect the user behaviors due to its limited range of actions, sensitivity to changes, and stability over time. While shell command lines and audit events have less characteristics compared with system calls.
- Almost exhaustive match mode. All the available unique system calls sequences are used to characterize system normality, which constructs a broad boundary to encompass normal behavior.

For frequency-based ADs, less characteristics of their operating environments, e.g., unpredictable range of activities, instabilities over time, make them suffer from low detection capability. According to the corollary 1, only the huge size of normal data set provides them an opportunity to decrease expected error between probability estimation and stochastic generator to a low level.

## 5 Evaluation of the Anomaly Detectors

Another challenging problem left in this research community is the ADs' evaluation. Most existing IDSs take 1998 and 1999 DARPA Intrusion Detection System Evaluations Data Set as benchmark for evaluating their performance, and most of them focus on tallying with detection accuracy and false positive rate of detection methods, rather than the fundamental understanding of evaluation environment. Therefore, the specific design of ADs based on particular situation and strong assumption limit their application to some extent.

Mchugh [12] gave a thorough analysis of the so-called benchmark data set, and proposed the essential conditions that ideal measurements should have. The first is the primary method, i.e ROC (Receiver Operating Curve), to present the results of the evaluation provides no insights into the root-causes for IDS performance, therefore more helpful metrics should be developed; Second, the curse of the false alarms generation has not been explained clearly, henceforth, the useful description of the difference between activities that are identified correctly as an attack and those that provoke a false alarm needs more insightful investigation. Third, to make sure that the false alarm rate for synthetic data has an obvi-

ous relationship to that of real data, background traffic data characterization is needed for calibrated artificial test data sets.

Up to now, we have not found such work that meet the strict requirements completely. With the problem that whether the environment regularity has effect on the probabilistic algorithms-based ADs, Maxion and Tan [11] provided an idea for successful synthesis, and the results verified their hypothesis. But their model is too simple to interpret more complex models, and some additional observational work from real data is still needed. In addition, only juxtapositional anomalies was considered in that model, while temporal anomaly detection was left.

Inspired by those former works, we have a primary idea to generate synthetic data for the general evaluation of ADs. Although it is still during the process of verification, we believe that it will contribute to the development of anomaly detection evaluation to some extent: First, collect pure real normal data source from a real environment, and mapping those collected data into controllable domain (for example, mapping network packets into wavelet domain and approximate host audit data as the Planck distribution respectively); Second, apply some candidate anomaly detectors to the controllable data set, and analyze the data that ever provoked false alarms. This step should be done recursively to prune the data as pure normal data without confused false alarms; Third, in order to ensure the regularity of processed data, information-theoretic measures could be used to divide the data as smaller but purer ones; Finally, artificial anomalies (such as foreign symbols or sequences, and rare sequences) are incorporated into the data. One way to make it more effective is to add predefined anomalies one by one, until to a determined amount.

## 6 Concluding Remarks

In general, following questions motivate us to do this work:

- The operational limits of ADs are from themselves or from the operating environments they run.
- Whether a better system normality characterization can improve the performance of ADs, if it does, how?
- How to select proper ADs for a particular situation when we take into consideration the trade-off between performance and cost.
- Although there are many available ADs, it is hard to find a way to evaluate their performance in terms of admitted criteria.

Above questions have been analyzed and discussed in a general way based on some available achievements, although there are still some problems pose open, and some proposed ideas remain verification and implementation, we believe that future work along this line could contribute additional insights into the research and application of anomaly detection. We believe that it is important to develop a framework for the anomaly detection field, including characterization, identification and evaluation of their operating environment in order to guarantee

their formal and rapid development, and it seems more important than just pruning detector itself regardless of its insightful understanding and broader application.

## Acknowledgement

This research is conducted as a program for the "Fostering Talent in Emergent Research Fields" in Special Coordination Funds for Promoting Science and Technology by Ministry of Education, Culture, Sports, Science and Technology.

## References

1. Mark Burgess, Harek Haugerud, and Sigmund Straumsnes, "Measuring System Normality," *ACM Transactions on Computer Systems,* Vol.20, No.2, May 2002, Pages 125-160.
2. Graham Cormode, Mayur Datar, Piotr Lndyk, and S.Muthukrishnan, "Comparing Data Streams Using Hamming Norms(How to Zero In)" *IEEE Transaction on Knowledge and Data Engineering,* Vol.15, No.3, May/June 2003, P529-540.
3. Forrest,S.,Hofmeyr,S.A.,&Longstaff,T.A, "A sense of self for UNIX processes," *In proceedings of 1996 IEEE Symposium on Security and Privacy,* Los Alamitos, CA: IEEE Computer Society Press.
4. Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, Liadan O'Callaghan, "Clustering Data Streams: Theory and Practice," *IEEE Transaction on Knowledge and Data Engineering,* Vol.15, No.3, May/June 2003, P515-528.
5. Paul Helman and Gunar Liepins, "Statistical Foundataions of Audit Trail Analysis for the Detection of Computer Misuse," *IEEE Transaction on Software Engineering,*Vol.19, No.9, September 1993.
6. Steven A. Hofmeyr, Stephanie Forrest, Anil Somayaji, "Intrusion Detection using Sequences of System Calls," *Journal of Computer Security,* Page:151–180, 1998.
7. Stefan H.Steiner, "Grouped Data Exponentially Weighted Moving Average Control Charts, " *Technical Report,* Universtiy of Waterloo, 1997.
8. M.Hutter, "Optimality of universal Bayesian sequence prediction for general loss and alphabet," *Journal of Machine Learning Research* , 4 (2003): 971-1000.
9. W.Lee and D.Xiang, "Information-theoretic meaasures for anomaly detection," In *IEEE Symposium on Security and Privacy,* 14-16 May 2001, Oakland, California, pages 130-143, IEEE Computer Society Press, Los Alamitos, California, 2001.
10. Sheng Ma, and Chuanyi Ji, "Modeling Heterogeneous Network Traffic in Wavelet Domain," *IEEE/ACM Transactions On Networking,* Vol.9, No.5, October 2001, P634-649.
11. Roy A. Maxion, and Kymie M.C. Tan, "Anomaly Detection in Embedded Systems," *IEEE Transaction on Computers,* Vol.51, No.2, Feb,2002.
12. John Mchugh, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory," *ACM Transactions on Information and System Security,* Vol.3, No.4, November 2000, Pages 262-294.
13. Ray J.Solomonoff, "Three Kinds of Probabilistic Induction: Universal Distributions and Convergence Theorems," *Machine Learning,*

14. Kymie M.C. Tan and Roy A.Maxion, ""Why 6" Defining the Operational Limites of stide, an Anomaly-Based Intrusion Detector," *Proceedings of the 2002 IEEE Symposium on Security and Privacy (S&P'02)*, 2002.
15. Christina Warrender, Stephanie Forrest, Barak Pearlumtter, "Detecting Intrusions Using System Calls: Alternative Data Models," *1999 IEEE Symposium on Security and Privacy,* May,1999.
16. Nong Ye, Xiangyang Li, Qiang Chen, Syed Masum Emran, and Mingming Xu, "Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data," *IEEE Transaction on Systems, Man, and Cybernetics-Part A:Systems and Humans,*Vol.31, No.4, July 2001.
17. Dit-Yan Yeung, Yuxin Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern Recognition 36* (2003) 229-243.

# Detection of Distributed Denial of Service Attacks Using Statistical Pre-processor and Unsupervised Neural Networks

Rasool Jalili, Fatemeh Imani-Mehr,
Morteza Amini, and Hamid Reza Shahriari

Department of Computer Engineering,
Sharif University of Technology, Tehran, Iran
jalili@sharif.edu, {imani, shahriari}@mehr.sharif.edu
amini@ce.sharif.edu

**Abstract.** Although the prevention of Distributed Denial of Service (DDoS) attacks is not possible, detection of such attacks plays main role in preventing their progress. In the flooding attacks, especially new sophisticated DDoS, the attacker floods the network traffic toward the target computer by sending pseudo-normal packets. Therefore, multi-purpose IDSs do not offer a good performance (and accuracy) in detecting such kinds of attacks.

In this paper, a novel method for detection of DDoS attacks has been introduced based on a statistical pre-processor and an unsupervised artificial neural net. In addition, SPUNNID system has been designed based on the proposed method. The statistical pre-processing has been used to extract some statistical features of the traffic, showing the behavior of DDoS attacks. The unsupervised neural net is used to analyze and classify them as either a DDoS attack or normal. Moreover, the method has been more investigated using attacked network traffic, which has been provided from a real environment. The experimental results show that SPUNNID detects DDoS attacks accurately and efficiently.

**Keywords:** DDoS Attacks, Intrusion Detection System, Unsupervised Neural Nets, Statistical Pre-Processor.

## 1 Introduction

Generating successful flooding DoS[1] attack on today's powerful computers is not possible using single ordinary computers. One of the solutions to have a succeed attack is to distribute the attack among a group of computers around the network. Moreover, tracing an attack originated from multiple sources is much harder than from single source attacks. Consequently, attackers can generate distributed DoS attacks and sustain any network bandwidth using thousands of computers.

A DDoS attack consists of three main components: master or main attacker, slave computers, and the victim computer. The main attacker initiates the attack from the master computer and tries to find some slave computers to be involved in the attack.

[1] Denial of Service.

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 192 – 203, 2005.

A small piece of software is installed on the slave computers to run the attacker commands. The attack scenario continued through a command issued from the attacker resides on the master computer toward the slave computers to run their pieces of software. The mission of the piece of software is to send dummy traffic destinated toward the victim. Therefore, the victim will not be able to do anything to prevent this attack. To reduce the network traffic, the victim should detect the attack just in time to be able to block some IP addresses. Although on time detection of an attack has an important role in preventing the progress of the attack, detection of DDoS[2] attacks is not so easy. As DDoS attack generation tools and methods try to increase traffic toward the victim computer using generating normal packets, signature based intrusion detection systems are unable to detect such attacks.

In this paper a new approach to detect distributed DoS attacks using statistical perprocessor and unsupervised neural nets is presented. In this method an unsupervised artificial neural net has been used to analyze and classify extracted statistical features.

The rest of the paper organized as follows: section 2 represents related works. The overall approach is described in section 3. Section 4 describes the base SPUNNID system [1], as the base architecture. Section 5 presents evaluation results of our method. Section 6 draws some conclusions and future works.

The preparation of manuscripts which are to be reproduced by photo-offset requires special care. Papers submitted in a technically unsuitable form will be returned for retyping, or canceled if the volume cannot otherwise be finished on time.

## 2 Related Work

Flooding attacks create enormous amount of normal packets and create anomalies in the target network traffic. Therefore, most approaches to detect such kinds of attacks, tries to detect these anomalies using semi statistical methods. In this paper, we use statistical methods to construct a Statistical Pre-Processor to extract some features, which demonstrates the behavior of DDoS attacks. Then an unsupervised neural net is used to analyze and classify these features. Accordingly, in this section we overview some research on Detection of DDoS attacks and IDSs based on unsupervised neural nets.

Authors in [2] introduced MULTOPS data structure to detect DDoS attacks. Using the data structure, they detected DDoS attack by searching for significant asymmetries between packets to and from different subnet. AGURI [3] as a monitoring tool uses the traffic pattern aggregation method, to monitor the traffic in a long term and detect DDoS attacks. In [4] a scheme to detect DDoS attacks by monitoring the increase of new IP addresses was proposed. In addition, it was presented that a sequential change point detection algorithm can identify when an attack has occurred. In [5], efficient adaptive sequential and batch sequential methods for an early detection of DDoS attacks have been developed.

In [6], the network traffic, which is expressed in terms of *Tcp flag rates* and *protocol rates,* has been analyzed and it has been shown that when the flooding attacks are in effect, intuitively the two rates can be distinctive and predictive, due to the explo-

[2] Distributed Denial of Service.

sion of TCP flags or specific protocol packets. In [7] an approach to reliably identifying signs of DDOS flood attacks based on LRD (long-range dependence) traffic pattern recognition has been discussed. In [8] the authors introduced an automated methodology for analyzing DoS attacks that is based on ramp-up and spectral analysis to build upon existing approaches of header analysis. This identification framework can be used as part of an automated DDoS detection and response System.

In [9], Feinstein and Schnackenberg, present statistical methods to identify DDoS attacks by computing entropy and frequency-sorted distributions of selected packet attributes. In [10] the effects of multivariate correlation analysis on the DDoS detection were discussed and a covariance analysis model for detecting SYN flooding attacks was proposed. In [11] a combined data mining approach for modeling the traffic pattern of normal and diverse attacks was proposed. This approach used the automatic feature selection mechanism for selecting the important attributes.

Some early research on IDSs attempted to use neural nets for intrusion detection. Such systems must be used in intrusion detection after initial training on normal or attack behaviors (or hybrid of theses behaviors). Both supervised and unsupervised neural nets have been used in IDSs till now such as MLFF, Recurrent, Adaptive, SOM and ART nets.

In [12] the authors presented a robust neural network detector for Distributed Denial-of-Service (DDoS) attacks in computers providing Internet services. A genetic algorithm was used to select a small number of efficient features from an extended set of 44 statistical features, which are estimated only from the packet headers.

Most supervised neural net architectures require retraining in order to improve analysis capability due to changes in the input data, but unsupervised net offers increased level of adaptability to neural nets and are able to dynamically improve their analysis capability [13].

Most of the network-based systems in unsupervised based IDSs used self-organizing maps (SOMs) neural nets and only a few systems used other types of unsupervised neural nets. In [14], multiple SOMs are used for intrusion detection, where a collection of more specialized maps is used to process network traffic for each protocol separately. Each neural net was trained to recognize the normal activity of a single protocol.

Most statistical methods reviewed in this section use some thresholds to detect DDoS attacks. In many cases, such thresholds cannot distinguish the normal behavior from the attack behavior precisely. In this paper, some statistical features showing the behavior of DDoS attacks have been extracted, and using unsupervised neural nets, they have been classified into normal and attack. In application phase of the neural net, this classification is used to detect DDoS attacks.

## 3 IDS for DDoS Attacks

General network based intrusion detection systems are either packet based or connection based. These systems attempt to detect attacks through analyzing packet or connection templates correspondingly. In DDoS attacks, the attacker floods the network traffic toward the target computer by sending semi normal packets. Therefore, multi purpose IDSs do not have a good performance (and accuracy) in detecting such kinds

of attacks. The DDoS victim, which is encountered with a vast number of normal packets, is overloaded, cannot provide services to its legitimate network users and denies them. The attacker tries to flood target network traffic with semi normal packets, but some statistical features on target network traffic will change under DDoS attack such that it becomes different from (and more complicated that) normal ones. Extraction of these features and analyze them is helpful to detect DDoS attacks. The normal network traffic of each computer depends on its user services. In addition, relatives to user services of the computer, it may be different during the day and night. Therefore, the value of some statistical features on DDoS attack generated network traffic, usually is different from values of the features on normal network traffic of the target computer.

In this paper, statistical features in a minor time interval have been extracted from network traffic, and it is shown that these features in attack network traffic differ from those in normal network traffic. Using such exploited result, it is possible to detect DDoS attacks more precisely. Due to the extend of network traffic in the target computer, regardless of the type of traffic in different times, the area of our extracted statistical features is extensive. Because of high clustering power, it seems that unsupervised artificial neural net is a more suitable tool for this purpose.

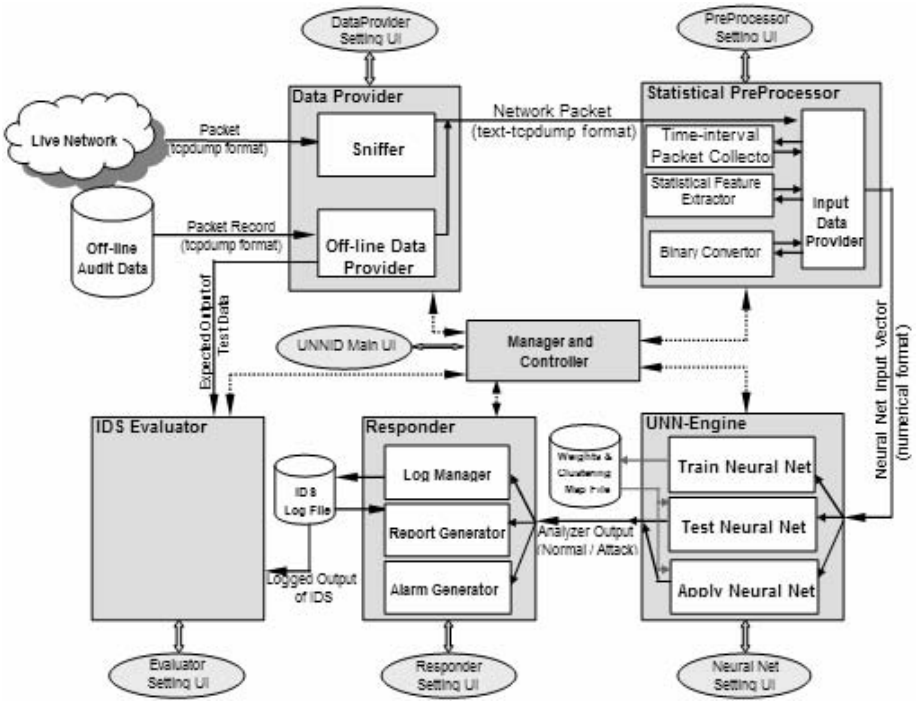



**Fig. 1.** Statistical Pre-Processor and Unsupervised Neural Net based Intrusion Detector (SPUNNID) System Architecture

In this paper, provided normal and attack network traffic is divided into minor time intervals, denoted as *T*. The time interval *T* includes all packets that their timestamps agree with that interval. Then we extract our statistical features from these time intervals. These features form the training vector of neural net. A typical time interval T can be labeled as "Normal" or "DDoS Attack" relative to its attack packets rate. Neural net processes these vectors and automatically clusters them as "Normal" or "DDoS Attack" as well as belonging of them to special belong to their variety.

In the application phase of neural net, packets belonging to the time interval *T* have been captured and statistical features have been extracted. The unsupervised neural net uses these features as input vector, clusters them, and detects their type as "normal" or "DDoS Attack".

In our previous research on IDSs, we introduced an Unsupervised Neural Net based Intrusion Detector (UNNID) system [1], which was network based. That system could detect attack through analyzing packets or connections signature. We promote UNNID system by adding the statistical pre-processor to detect DDoS attacks; yield in a new system called Statistical Pre-Processor & Unsupervised Neural Net based Intrusion Detector (SPUNNID). The architecture of SPUNNID system is presented in the next session.

## 4 System Architecture

The architecture and main components of our SPUNNID system is shown in figure 1. The system is designed firstly to facilitate training, testing, tuning and evaluating different types of unsupervised neural nets for intrusion detection, and secondly to apply them for analyzing network traffic in on-line and off-line mode in order to classify classifying network traffic into normal and attack.

In SPUNNID, *Data Provider* collects packets from network audited data file (off-line mode) or live network (on-line mode) and send data in the text form to the *Statistical Pre-Processor* component. *Statistical Pre-Processor* extracts some features from packets of the specified time interval using statistical techniques. *The component* converts extracted feature vector into the numerical form and if needed converts numerical data into binary or normalized form, and send them to Unsupervised *Neural Net based Engine*. The *UNN-Engine* uses data either for training and testing its neural net or for analyzing and detecting denial of service attacks. The analyzer output (normal or DoS attack type) is given to *Responder* for recording in the system log file and generating alarm in case an attack is detected. The *IDS Evaluator* component provides a facility for reporting true detection rate, false positive detection rate, false negative detection rate, and other criteria to evaluate our system in detecting denial of service attacks. This component calculates these criteria by comparing the output of IDS and expected output of the system, which is determined by labels on records of test data. The criteria are:

- Exact True Type Detection Rate (detecting normal traffic from attack and recognizing the known attack type);
- True Detection Rate (only separating normal traffic from attack);
- False Positive Detection Rate (miss-detecting attack);
- False Negative Detection Rate (failing to detect attack when it is occurs);

Finally the *Manager & Controller* component in this system manages and directs other components to work in one of the possible modes (e.g. training, testing, and detecting) based on the command and parameters delivered from the operator.

### 4.1 UNN-Engine

One of the main components of UNNID is UNN-Engine, which has the role of analyzing the network traffic and detecting denial of service attacks using one of the convenient unsupervised neural nets named Adaptive Resonance Theory (ART) nets. In unsupervised ART nets, input patterns may be presented several times and in any order. Each time a pattern is presented, an appropriate cluster unit is chosen and related cluster weights are adjusted to let the cluster unit learn the pattern. In these nets, choosing a cluster is based on the relative similarity of an input pattern to the weight vector for a cluster unit, rather than the absolute difference between the vectors (that is used in SOM nets). As in the most cases of clustering nets, the weights on a cluster unit may be considered an exemplar (or code vector) for the patterns placed on that cluster. ART nets are designed to allow the user to control the degree of similarity of patterns placed on the same cluster that can be done by tuning the *vigilance parameter* in these nets. In ART nets, the number of clusters is not required to be determined in advance, so the vigilance parameter can be used to determine the proper number of clusters to decrement probability of merging different types of clusters to the same cluster. Moreover, ART nets have two other main characteristics. First is *stability* which means a pattern not oscillating among different cluster units at different stages of training, and second is *plasticity* which means the ability of net to learn a new pattern equally well at any stages of learning.

Stability and plasticity of ART nets and the capability of clustering input patterns based on the user controlled similarity between them, made these nets more appropriate for using in IDSs, rather than most of other types of unsupervised nets (such as SOM) for classifying network traffic into normal and intrusive/ attack. For this purpose, we used a type of unsupervised ART nets, named ART-1. ART-1 is the first type of ART nets, which designed for clustering binary inputs.

In *SPUNNID*, *Statistical Pre-Processor* feeds binary input vector to ART-1 net. In the training phase, the input vectors are clustered through ART-1 net regardless of their nature (normal or intrusive). Following the training phase, system must determine the neurons of each type of cluster and assign name to each cluster using the label of each time interval records (in train data). Each cluster has the same name as its units. Each unit is named based on the type of the majority of input data that the unit represent the winning or best matching for. This reduces to constructing a *Clustering Map.* In the map, units are clustered together to indicate either the normal traffic, known trained attacks, or possibly a new DoS attack. New attacks may appear in abnormal traffic, which is neither a normal traffic nor a known DoS attack.

### 4.2 Statistical Feature Selection

The attacker uses flooding attacks such as UDP flood, Syn flood, ICMP flood, and ICMP Smurf to generate DDoS attacks and finally floods the target network. When the DDoS attack is being initiated, the number of packets in the network increases

instantly. The number of packets may also be increased in normal network traffic. Increasing the number of packets in normal heavy network traffic and attacked network traffic cannot be distinguished easily. Analyzing the DDoS attack and the normal network traffic shows that packet rates are a better criterion for deciding about happening of the attack. Tcp flags rates and protocol rates in normal network traffic change slowly. Flooding packets, which are used by the attacker, specify the behavior of the DDoS attack and the effect of this attack in changing the statistical features. In attacked network traffic, depends on its behavior, these rates change and become different from the normal traffic. Therefore, we extracted the features that shown these rates in a typical time interval *T*.

These features are:

- $N_{ICMP}$: the percent of *ICMP* packets.
- $N_{UDP}$: the percent of *UDP* packets.
- $N_{TCP}$: the percent of *TCP* packets.
- $N_{TCPSYN}$: the percent of *SYN* packets in *TCP* packets.
- $N_{TCPSYNACK}$: the percent of *SYN+ACK* packets in *TCP* Packets.
- $N_{TCPACK}$: the percent of *ACK* packets in *TCP* packets.
- $A_{Packet\ Header\ Sizes}$: The packet header sizes average.
- $A_{Packet\ Data\ Sizes}$: The packet data sizes average.

After extracting these features, the neural net input vector is constructed.

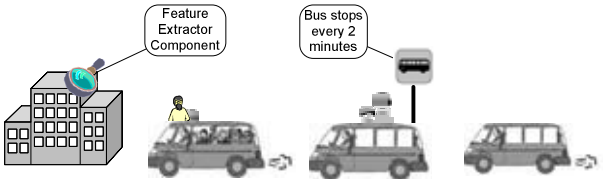



**Fig. 2.** Statistical pre-processor is similar to bus station system

The operation of *Statistical Pre-Processor* is similar to an operation of the Bus Station system. The packets, which have been come from Data Provider, wait for coming of a bus. The bus comes to the station every T seconds (the length of time interval), and takes captured packets to the Statistical Features Extractor component. This component processes the coming packets, extracts selected features and converts these features to binary or normalized form in order to feed neural net sensors in UNN-Engine component.

## 5 Evaluation

Providing proper data (including attack and normal network traffic) play main role to get a high performance in detection of DDoS attacks. If the data covers normal traffic (during day and night) and all types of DDoS attacks, the training phase of the neural net will be done better and thus having a better performance for the net. In the follow-

ing subsection, a method of gathering suitable data, for training and testing phase of the neural net, will be explained and then evaluation results will be shown.

### 5.1 Data Gathering

Generating DDoS attacks on a real environment and gathering normal and attack data are so costly. Meanwhile it has been tried to maintain the gathered data comparable with the real one.

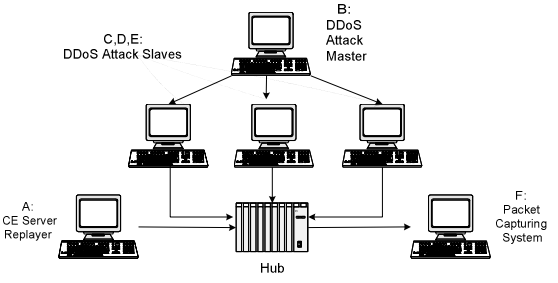



**Fig. 3.** Topology of Data Gathering Network For Evaluation

The network topology, shown in figure 3, is used to generate data in our evaluation. In this topology, the captured network traffic on the real environment has been replayed using the computer A. This computer plays the role of the real Computer CE[3]. The computer B is considered as the DDoS attack master. To generate an attack, B orders the DDoS attack slave computers including C, D and E to initialize the attack. These computers initialize the attack through sending flooding packets into the network. These packets are compounded with the CE computer network traffic and form the traffic under DDoS attack, captured by the computer F. If DDoS attack does not flow in the network, F captures the CE normal network traffic. Finally the captured network traffic by F includes both normal and attack network traffic.

The traffic of CE was captured as normal network traffic for the training and testing phase. Using the network topology, the traffic was replayed, and DDoS attack traffic was generated at once. Regarding this scenario, flooding attacks for the training data include UDP Flood, SYN Flood, ICMP Flood, ICMP SMURF, and mixture of these flooding attacks, each flowing for a period of 15 minutes. So the generated training traffic includes normal and attack network traffic. The traffic for the testing phase was generated using this method. This traffic contains the network traffic under the DDoS attack, trough 15 minutes.

### 5.2 Evaluation Results

Evaluation is achieved using the network topology described above. Parameters considered in the evaluation phase are:

[3] The server in Department of Computer Engineering, Sharif University of technology.

- The number of clusters in ART1 neural net.
- The number of epochs in the training phase of ART1 neural net.
- The vigilance parameter of ART1 neural net.
- The length of the time intervals.
- The vigilance parameters value in the application phase of SPUNNID.

The effect of these parameters has been evaluated based on the Exact True Type detection Rate (ETTR), True Detection Rate (TR), False Positive detection Rate (FPR) and False Negative detection Rate (FNR).

**The Number of Clusters in ART1 Neural Net**

Specifying the number of clusters depends on the variation of training data. Furthermore, the volume of training data and its variations depends on the length of time intervals in the introduced method. So changing the length of time intervals, results in changing the number of clustered neural net clusters. Generally, the number of clusters must cover the selected data of the training phase.

**The Number of Epochs in the Training Phase of ART1 Neural Net**

The next parameter, which can affect on our experimental results, is the number of epochs or iterations in neural net training phase. Our experiments show that, 100 is a good option in this regard.

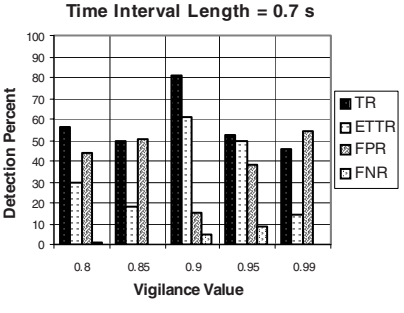



**Fig. 4.** The effect of changing the Vigilance value on *SPUNNID* detection performance

**The Vigilance Parameter of ART1 Neural Net**

The vigilance parameter is the most important parameter that can affect on ART1 clustering and classification quality. This parameter specifies the degree of similarity of patterns placed on the same cluster. Neither high value nor low value for this parameter is suitable for our system. To specify the appropriate value of vigilance parameter and its effect on the system performance, the system was trained with different vigilance values and ETTR, TR, FPR, and FNR criteria were evaluated. Results of the experiments are presented in figure 4. The results show that ART1 with vigilance value of 0.9 offers the best level of detection performance.

**The Length of Time Intervals**

Defining the best value for the length of time interval is a tradeoff between the load of captured network traffic for training phase of ART1 neural net, the existence of enough processing power, and the expected efficiency. Neither high value nor low value for this time interval length is suitable for our system. Increasing or decreasing the length of time intervals resulted in increment or decrement of the number of neural net input data. Also in a short time interval, the number of data variation is less than the longer time interval. Thereby, the selection of longer time intervals is more efficient, if the number of training data of neural net is big sufficiently, and there is enough processing power for training of neural net. In this case, the variation and the number of clustering data become greater, thus the training power of neural net will increase.

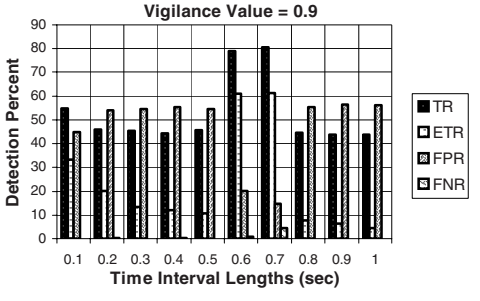



**Fig. 5.** The effect of changing the length of Time Interval on SPUNNID detection performance

Generally, the response time of the introduced method corresponds to the selected time interval length. If the length of time intervals is selected very long, the response time will increase. However, the increasing of response time is not so impressive for computer network administrators. In addition, the length of time intervals should not exceed from the length of happened DDoS attacks time. It seems that the length of time intervals cannot exceed from 2 minutes.

If the training data is not very high, selection of shorter time interval can produce suitable results. The minimum length of time intervals is a value in which the effect of DDoS attacks can be represented using the selected features. For example, assume a time interval, which contains only two packets. The type of these packets cannot effect on our features so as they can use to suitable DDoS attack detection. Furthermore, these two packets cannot create a sufficient variation of selected features. Therefore, the good selection of time interval lengths is tradeoff between the load of captured network traffic for training phase, the existence of enough processing power, and the expected efficiency.

Figure 5 shows the best time interval length, which determines the appropriate value for time interval length in our gathered data.

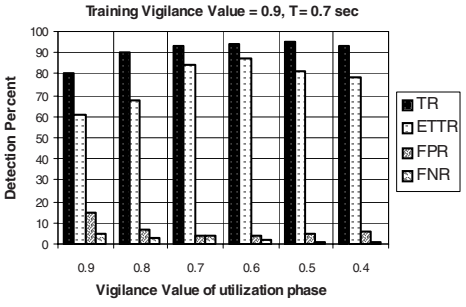



**Fig. 6.** The effect of changing the vigilance value in testing phase on SPUNNID detection performance

**The Vigilance Value in Application Phase of ART1 Neural Net**

One of the main disadvantages of many commercial and expert system based IDSs are their weakness in detecting changed known attacks and also new attacks. So for enhancing our system flexibility and generality we decreased system sensitivity in clustering input patterns by changing in vigilance parameter in application phase of SPUNNID system. It has been shown that the selection of the smaller value for vigilance parameter in application phase can decrease the sensitiveness of clustering in unsupervised neural net based engine. Therefore in many cases, if the value of vigilance parameter decreases, the number of "Can Not Cluster" messages decreases too. Figure 6 shows this obtained result for this purpose.

## 6 Conclusion and Feature Works

In this paper, we introduced a new approach for detection of DDoS attacks, using unsupervised neural nets and statistical methods. In this approach, the statistical features showing the behavior of these attacks along a minor time interval, have been extracted. The ART1 net has been used to analyze and classify these features. Evaluation results show that the approach in 94.9 percent of times is able to recognize the attacked traffic from the normal one. In addition, the introduced approach detects DDoS attack in less than a second (0.7 second in best case).

One of the neural net advantages, specially unsupervised neural nets, is that it can classify the input data automatically without human intervention. Therefore, extraction of proper features to train neural nets (statistical or non-statistical) which can show the effect of a typical attack can be useful to promote detection performance.

In our future investigation, we intend to use a FIFO queue of extracted statistical feature vector as the neural net input vector. When DDoS attack happens, the correlation between our statistical parameters in some neighbor time intervals decreases. Using the queue of statistical features of the neighbor time intervals as a neural net input vector shows the change of this correlation better than considering single time interval.

## References


1. M. Amini, and R. Jalili, *"Network-Based Intrusion Detection Using Unsupervised Adaptive Resonance Theory (ART)"*, Proceedings of the 4th Conference on Engineering of Intelligent Systems (EIS 2004), Madeira, Portugal, 2004.
2. T. M. Gil and M. Poletter. *"Multops: a data-structure for bandwidth attack detection"*, In Proceedings of USENIX Security Symposium'2001, 2001.
3. R. Kaizaki, K. Cho, O. Nakamura, *"Detection Denial of Service Attacks Using AGURI"*, International Conference Telecommunications, Beijing China, June 2002.
4. T. Peng, C. Leckie and R. Kotagiri. *"Proactively Detecting Distributed Denial of Service Attacks Using Source IP Address Monitoring",* In Proceedings of the Third International IFIP-TC6 Networking Conference (Networking 2004), Athens, Greece, 2004.
5. R. Bazek, H. Kim, B. Rozovskii, and A. Tartakovsky, *"A novel approach to detection of enial-of-service attacks via adaptive sequential and batch-sequential change-point methods*", IEEE Systems, Man and Cybernetics Information Assurance Workshop, June 2001.
6. Sanguk Noh , Cheolho Lee ,Gihyun Jung, Kyunghee Choi, "Using Inductive Learning for the Detection of Distributed Denial of Service Attacks", International Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine and Mobile Technologies on the Internet, 2003.
7. L. Ming Li, *"An approach to reliably identifying signs of DDOS flood attacks based on LRD traffic pattern recognition"*, Computers & Security, Vol. 23, Issue 7, Elsevier, ISSN 0167-4048, April 2004.
8. A. Hussain, J. Heidemann, and C. Papadopoulos. *"A Framework for Classifying Denial of Service Attacks".* In Proceedings of the ACM SIGCOMM Conference, pp. 99-110, Karlsruhe, Germany, August 2003.
9. L. Feinstein, D. Schnackenberg, R. Balupari, D. Kindred, *"Statistical Approaches to DDoS Attack Detection and Response*", DARPA Information Survivability Conference and Exposition, 2003.
10. Shuyuan Jin, Daniel S. Yeung, "*A Covariance Analysis Model for DDoS Attack Detection*", IEEE Communications Society, 2004.
11. K. Mihui, N. Hyunjung, C. Kijoon, B. Hyochan, and N. Jungchan, "A Combined Data Mining Approach for DDoS Attack Detection", Information Networking: Networking Technologies for Broadband and Mobile Networks International Conference (ICOIN 2004), Busan, Korea, February 2004.
12. D. Gavrilis, I. Tsoulos, E. Dermatas, *"Feature selection for robust detection of distributed Denial-of-Service attacks using genetic algorithm"*, Methods and Applications of Artificial Intelligence: Third Hellenic Conference on AI (SETN 2004), Samos, Greece, May 2004.
13. J. Cannady, *"Artificial Neural Networks for Misuse Detection"*, In Proceedings of National Information Systems Security Conference, 1998.
14. B.C. Rhodes, J.A. Mahaffey, and J. D. Cannady, *"Multiple Self-Organizing Maps for Intrusion Detection"*, In Proceedings of 23rd National Information Systems Security Conference, 2000.

# Visual Spoofing of SSL Protected Web Sites and Effective Countermeasures

Andre Adelsbach, Sebastian Gajek, and Jörg Schwenk

Horst Görtz Institute for IT Security, Ruhr Universität Bochum, Germany
{andre.adelsbach, sebastian.gajek, joerg.schwenk}@nds.rub.de

**Abstract.** Today the standard means for secure transactions in the World Wide Web (WWW) are the SSL/TLS protocols, which provide secure (i.e., private and authentic) channels between browsers and servers. As protocols SSL/TLS are considered secure. However, SSL/TLS's protection ends at the "transport/session layer" and it is up to the application (here web browsers) to preserve the security offered by SSL/TLS.

In this paper we provide evidence that most web browsers have severe weaknesses in the browser-to-user communication (graphical user interface), which attackers can exploit to fool users about the presence of a secure SSL/TLS connection and make them disclose secrets to attackers. These attacks, known as "Visual Spoofing", imitate certain parts of the browser's user interface, pretending that users communicate securely with the desired service, while actually communicating with the attacker. Therefore, most SSL/TLS protected web applications can not be considered secure, due to deficiencies in browser's user interfaces.

Furthermore, we characterise Visual Spoofing attacks and discuss why they still affect today's WWW browsers. Finally, we introduce practical remedies, which effectively prevent these attacks and which can easily be included in current browsers or (personal) firewalls to preserve SSL/TLS's security in web applications.

## 1 Introduction

The recent growth of the World Wide Web (WWW) and its broad acceptance even for security critical applications, such as banking and auctions, requires that web browsers provide means to establish secure, i.e., authentic and private, communication channels to servers. Today the Secure Socket Layer (SSL) [1] protocol and its successor, the Transport Layer Security (TLS) protocol [2], are the de-facto standard for secure communication on the Internet. SSL/TLS are publicly specified and have undergone a wide peer-review, which is the reason for them, as protocols, being believed to be secure [3]. Besides a secure connection, there must be a trustworthy and reliable way to inform users about the security properties of a connection such that users are able to distinguish a secure connection from an insecure one. In general, a connection's properties are indicated by several features of a browser's user interface, e.g., the padlock icon in the status bar or the certificate dialog. In the following we will refer to the specific

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 204–216, 2005.

features of a web browser's user interface, which visualise information about the security status of a connection, as the browser's secure connection indicators (*BSCIs*).

Attacks, which tamper with these indicators to fool the user about the connection's real status, are called *Visual Spoofing* (*VS*) attacks. These attacks exploit the flexibility of browser's user interfaces to replace real BSCIs with fake ones[1] and imitate the look and feel of a trusted web site. By simulating a secure connection to a trusted service, the victim can be tricked to disclose any secret information[2], destined for the trusted web site, to the attacker. Consequently, VS attacks are a great threat for any secure web service, such as single-sign-on services, online banking or online shops.

VS attacks on their own, hosted on the attacker's web server, are harmless in general, because a victim would hardly direct his browser to that VS page by chance. Therefore, an attacker has to direct the unwitting user (or his browser) to the site hosting his VS attack. We refer to this auxiliary step as the *mounting attack*. Mounting attacks can be categorised into those operating on the network layer and those operating on the application layer. Prominent examples of the first category are ARP, IP or DNS spoofing attacks [4], whereas e-mail spoofing and URL spoofing are examples of the latter category. Due to the large number and diversity of mounting attacks, an attacker has many effective options for initiating VS attacks.

VS attacks have been studied for several years [5, 6, 7, 8] in the research community, but with only little impact on today's browsers. Recent studies [9, 10] show a strongly increasing number of VS attacks in the Internet, because, in contrast to buffer-overflow attacks, VS attacks do not require sophisticated expert knowledge of operating systems and low-level programming; VS is applicable by moderately experienced attackers. Therefore, there is still the necessity of discussing these attacks and, even more important, to develop practical, i.e., effective and easy to use, countermeasures.

Our paper is structured as follows: In Section 2 we briefly review the basic idea of VS attacks, discuss existing BSCIs and point out vulnerabilities of deployed web browsers that allow an attacker to trick even experienced, security aware users about the status of a SSL/TLS connection. In Section 3 we review several ways how to exploit these vulnerabilities such that all BSCIs indicating a trustworthy secure connection can be perfectly faked in design as well as in functionality. Then, in Section 4 we discuss mounting attacks that can be used to route users to sites hosting the actual VS attack. We review related work, especially proposed countermeasures, in Section 5 and assess their suitability

[1] More concretely, browser's BSCIs are deactivated and fake BSCIs are rendered in the display area of the browser, such that an user can hardly (if at all) distinguish fake from real BSCIs.

[2] Examples are payment information, such as credit card numbers and authorisation information (login, PIN and TAN) of online banking systems, or complete web identities administered by single-sign-on services (e.g., Microsoft's Passport) or attribute wallets.

and effectiveness to counter VS attacks. In Section 6 we propose effective countermeasures against VS attacks, which can be easily integrated into common deployed browsers and which allow even average, naive users to detect VS attacks without restricting user's convenience. In Section 7 we discuss two possible ways of implementing these countermeasures. A demonstrator is available online [11]. Finally, in Section 8 we summarise our results and conclude.

## 2 Visual Spoofing (VS)

The SSL/TLS protocol has undergone intense peer review without finding severe vulnerabilities in the latest versions. However, to achieve overall security in real world applications, it is important to carefully integrate SSL/TLS, which, among other things, comprises the way security relevant information is displayed to the user.

VS attacks exploit vulnerabilities in the presentation of security relevant information. It is important to note that VS attacks are not limited to web browsers, but can, in principle, be applied to almost any application, which offers remote access to the user interface. However, as web browsers are widely deployed, used for various types of security critical applications and specifically designed to offer remote web designers extensive control over the rendering process (unfortunately, including the browser's user interface), they are perfect targets for VS attacks.

VS attacks on web browsers exploit the rich features offered to web designers to fake those parts of a browser's user interface, which display information about the connection's security status. As a result, users believe to communicate over a secure channel with the desired web server (as indicated by the browser's user interface), while actually communicating with a rogue service or over an insecure channel.

### 2.1 Browser Secure Connection Indicator (BSCI)

All current web browsers provide means to inform users about the status of a SSL connection, which we will refer to as *Browser's Secure Connection Indicators* (BSCIs). BSCIs allow users to distinguish a secure connection from an insecure one by displaying information, such as the server's certified identity and the cryptographic property of the connection. As BSCIs are the browser's only means for users to get information about the security status of a connection or retrieved document, their authenticity is crucial to the overall security. Common BSCIs (here exemplarily described for Internet Explorer) are:

- The most eye-catching indicator for a SSL-protected connection is a **padlock icon**, which is displayed in the status bar if the current web page has been retrieved over an SSL connection. A double click on this icon opens the so called certificate dialog (see below).
- The **certificate dialog** displays detailed information about SSL's current status, such as the server authentication information (including server name

and certification authority) and the concrete cryptographic algorithms and key-lengths being used to protect the transmission of the rendered web page. For the user it is the prime means to evaluate the web site's authenticity.
- The **location bar** can be used to manually direct the browser to a certain web page, specified by a so called Uniform Resource Locator (URL). A URLs consists of the protocol's name followed by the address of the service. The prefix "https" in an URL indicates the use of SSL and is a further hint for a secure connection. After a web page has been retrieved and rendered the location bar displays the corresponding URL. Choosing an accurate address (mainly the domain name) for a web site can strengthen the trust in a document's source.
- The **menu bar** is an "indirect" indicator, as it contains no immediately visible information about the connection's status; it rather provides features, which may disclose a VS attack, and provides access to further BSCIs: Firstly, the menu bar ("View" menu) indicates the visibility of the browser's status and address bar. It is the only indicator for the real presence (and authenticity) of these BSCIs. Secondly, it provides access to the "document source" dialog, from which an experienced user can detect a VS attack, as well as access to the "document property" dialog, which contains details about cryptographic algorithms used to protect the retrieved document (e.g., cipher suite, key length).

### 2.2 Technical Preconditions for VS Attacks

For being susceptible to VS attacks browsers need to fulfil following preconditions: First, the browser's user interface has to be controllable by active web languages, e.g., Visual Basic Script or JavaScript, such that a retrieved web page can deactivate any BSCI when being rendered without the browser asking the user for permission or warning him. Second, the browser must have a standardised user interface. This allows an attacker to fake the SSL indicators without special knowledge about the user's user interface, as he can easily guess the browser's look and feel as expected by the user and fake it accordingly.[3]

Most currently available browsers, such as Microsoft's Internet Explorer 6, Netscape Navigator, and Firefox, fulfil these preconditions and are, as a matter of fact, susceptible to VS attacks. To our knowledge the Opera web browser does *not* fulfil these preconditions, as it does not allow an attacker to control the user interface by means of active web languages.

## 3 Proof of Concept Implementations of VS Attacks

The common principle of existing VS attacks is to deactivate the browser's BSCIs and display fake ones in the browser's rendering area by using design features of standard web languages.

[3] The latter condition holds for almost any standard application. Thus, it also holds for web browsers immediately.

An early proposed way to fake all BSCIs is to include images of BSCIs in a web page (see Felten et al. [5]). However, pure image-based VS attacks lack dynamic behaviour of the faked parts (e.g., certificate dialog or menu bar). Such static implementations are easily detectable by users and are only a minor threat in practice.

In [7] Li and Yongdong describe a VS attack containing a faked certificate dialog. A click-event on the padlock icon opens a fake certificate dialog, which displays wrong authentication information, while preserving the usual behaviour (response to mouse events). However, as this attack is based on Java Applets this leads to noticeable delays in rendering the fake certificate dialog.

*Our Proof of Concept Implementation.* Our proof of concept VS attack uses DHTML, i.e., it renders static BSCIs components with standard HTML and implements dynamic behaviour with JavaScript and Cascading Style Sheets (CSS). All these techniques are standard means in web development, available in almost any browser. We mainly focused on Microsoft's Internet Explorer 6, as it is the most widely used browser today. The VS attack opens a new browser window with deactivated BSCIs (menu, button and status bar). The fake status bar (with lock icon) is included as an image. A double click event on the lock icon of the fake status bar opens a fake certificate dialog (browser window), which contains faked certificate information. The location bar is faked by using a HTML form, because this allows us to intercept user inputs and react accordingly to simulate the standard user interface behaviour, as expected by the victim; it can be used to analyse users' surf behaviour and to redirect the victim to further spoofed web sites. The buttons in the button bar change (onFocus-event) when the user moves the mouse pointer over a button as in the standard IE user interface. Furthermore, the button's functionality (e.g., back, refresh, stop) can be simulated by binding suitable JavaScript functions to the onClick-event of the corresponding button image.

A new finding of our proof of concept VS implementation is that even the whole menu structure of the menu bar can be spoofed by means of the layer feature of dynamic CSS. This comprises the "View" menu, which allows us to trick users about the actually activated bars. It is even possible to fake other BSCIs (see Section 2.1) like the source code or cryptographic properties of the rendered page by applying the same techniques. As a figure would hardly show any differences between the faked user interface and the original user interface, we provide a demonstrator of the VS attack for Internet Explorer 6 online [12]. With this demonstrator, we prove that IE's user interface including all BSCIs, can be nearly perfectly faked. There remain two "imperfections" in our proof of concept VS attack:

- The title bar of the fake certificate dialog contains "Microsoft Internet Explorer", because it is rendered in an Internet Explorer window instead of a local operating system (Windows) dialog.
- The certificate dialog does not open if a pop-up blocker is active.

To remove these remaining "problems" an attacker may use an alternative implementation of the fake certificate dialog based Macromedia's Flash. A demonstrator is also online [12]. Furthermore, we want to stress that it is also possible to implement the whole VS attack in Flash. However, tests in our department showed that nobody, although having a strong background in IT-Security, was able to distinguish the original browser's user interface from the one of our VS demonstrator based on these imperfections. Therefore, we believe, that most users will not be able to detect such an advanced VS attack as well. In the following section we will address complementary mounting attacks.

## 4 Mounting Attacks

To mount a VS attack in practice, the attacker has to direct his victim to a web server hosting the actual VS attack. In this section, we review three well-known preparative *mounting attacks* and discuss their efficiency to illustrate the ease of mounting visual spoofing in practice.

*E-Mail Spoofing.* The attacker sends an e-mail to the user, which seems to origin from a trusted company that commonly contacts their clients with standardised mails, e.g., PayPal or eBay. The mail urges the user to follow a hyperlink referring to a malicious server that hosts the actual attack. Examples can be found in [9]. This mounting attack combined with a VS attack is known as *Phishing*. E-Mail spoofing is the most popular way of mounting VS, because it does not require sophisticated technical knowledge. Spoofed e-mails are commonly sent randomly to a large number of recipients in the hope, that at least some of the recipients are customers of the specific company forged by the attacker.

*URL Attacks.* Here, the adversary hosts the implementation of the VS attack in a web domain, which has a name similar to (and easy to confuse with) the domain name of the spoofed web site. Now the attacker regularly publishes the fake domain in search engines, or includes links to this domain on other web sites (e.g. advertisements). Examples for URL attacks are:

1. "`http://www.signin.ebai.com`", which is can be easily confused with the real URL "`http://www.signin.ebay.com`".
2. URLs such as "`http://www.paypal.com@the.attacker.com`" exploit a rarely used feature, which allows to include a login name in an URL by prepending a string "login@" to the address part of an URL. Therefore, this URL refers to domain "`the.attacker.com`" instead of `www.paypal.com`, which is interpreted as a login name instead of an address.

*Include or Refer to VS Attacks in Third Party Sites.* Another way to mount VS attacks is to include VS attacks in third party sites, which are used and trusted by many users (cross-site-scripting). A recent example includes VS code in an Ebay online auction to open a fake login page, which may send login and password to an attacker [13]. This attack may even be combined with elements to fake BSCIs. An attacker may also advertise wrong URLs in the name of some

trusted company (e.g., a bank) which actually refers to a VS attack instead of the company's real web site.

*Network Based Attacks and Man-In-The-Middle Attacks.* In this type of mounting attack the attacker intercepts the communication between client and server. To this end the attacker may apply techniques such as ARP or DNS spoofing to push himself between the communication of client and server. This mounting attack is very powerful, because is allows the attacker to target selected users and it does not depend on users' interaction (e.g., by following some advertised link). On the other hand, such network based mounting attacks require more technical skills than sending fake phishing emails.

We want to stress that VS attacks enable successful man-in-the-middle attacks against SSL, because a man-in-the-middle attacker may visually spoof the authentication information (including the certificate dialog) of the server, such that the user does not notice the man-in-the-middle. Furthermore, this is the reason, why server-based countermeasures against VS attacks cannot be effective.

## 5 Existing Countermeasures

First of all, we want to note that visual spoofing can be countered indirectly, by countering mounting attacks. However, in this paper our focus is on direct countermeasures against VS, such that we do not go into the details of preventing the mounting attacks outlined above.

Felten et al. [5] proposed to deactivate all active web languages which facilitate spoofing (e.g. JavaScript, ActiveX or Java). From today's point of view this proposal seems to be impractical, because active web languages strongly improve the service offered by web sites – in fact, most popular web sites would not work anymore if a user would disable active web languages in his browser. As the WWW gained popularity through these languages, their restriction would severely decrease the web's usability, comfort and acceptance.

In [14] the authors introduced an idea for a new web browser as a consequence of lacking long-term solutions. The authors proposed to include unspoofable features in web browsers that reveal the presence of VS attacks. More concretely, they proposed to apply synchronised random dynamic boundaries (SRDs). The idea of SRD is to distinguish authentic parts of the browser GUI from rendered content received from a server by changing the boundary colours of the real GUI pseudo-randomly and unpredictable for remote attacker. Users have to compare the changing colours of browser windows with those of a reference window. Boundaries of an original browser window will be synchronised with the reference window, while spoofed browser windows won't be correctly synchronised. This proposal has practical drawbacks: Firstly, blinking features may disturb users. Secondly, the proposed implementation seems to be weak, because it is based on the XML User Interface Language (XUL). XUL allows also remote users to change the look and feel of a browser by applying means of CSS and JavaScript, which may allow an attacker to spoof this feature as well. In [15] the author

describes how this vulnerability can be exploited. Therefore, we conclude that this proposal is not suitable to protect users against VS.

Li and Wu [7] proposed to prevent that the status bar is being deactivated by active web languages.[4] We consider this to be a good first step, but it is still not sufficient to completely counter VS attacks: an attacker can simply apply for a SSL certificate and host the spoofing attack on a SSL-enabled web server. A naive user will recognise the padlock icon, stemming from the attacker's certificate, and the spoofed URL displayed in the faked address bar. Therefore, he will believe to communicate securely to the requested service. Another proposal of Li and Wu is to improve the SRD concept by defining the reference point within the window (e.g. menu bar). This is also not an effective countermeasure, as such references can be spoofed even more easily.

In [8] the authors introduce the concept of *trusted credential area* (*TCA*). TCAs are solid, unspoofable areas in browsers' user interface, which visualise the authenticity of a web site by means of extended graphical credentials (e.g. brand logos, icons, seals). Thereby, certification occurs either by a trusted third party called *Logo Certification Authority* (*LCA*) or by self-certification. To reduce the involved overhead, the authors propose an extension of the TLS protocol. Our conclusion is that the idea of visualising trusted credentials is very good, especially for naive users. A brand logo is easier to understand than a list of cryptographic parameters. However, we see disadvantages in the LCA-proposal: Firstly, it either involves significant overhead or a new variant of the TLS protocol. Secondly, it is costly for a certification authority to verify that certified logos are sufficiently distinct, such that they cannot be confused by users. This problem already exists today in the context of brand imitations. This is why we highly appreciate the idea of self-certified logos. In independent work, we developed a countermeasure similar to the approach of Herzberg and Gbara: Whereas Herzberg and Gbara propose to use self-certified logos to authenticate web sites, we propose to authenticate the browser's user interface, which significantly reduces the "certification"-load put on users. In contrast to the approach of Herzberg and Gbara, the user needs only one personal logo to authenticate the user interface of his browser (see Section 6.1). In the following section we will discuss possible effective countermeasures in more detail.

## 6 Effective Countermeasures

The key-observation is that VS attacks aim at the browser's user interface and its perception by (naive) users. To counter VS effectively, two complementary types of measures are required:

1. **Improve User's Security Awareness:** The first and probably most important measure is to train users and improve their security awareness. Adequate security awareness of users is the ultimate prerequisite against any

[4] This has recently been implemented in Windows XP's Service Pack 2.

kind of VS attack. As long as a user does not care about security any technical countermeasure will be ineffective.

2. **Supporting Technical Measures:** As mentioned before, advanced VS attacks against standard browsers are hard to detect even for security specialists. Therefore, it is necessary to offer users reliable technical means to detect VS. Obviously, server-based countermeasures cannot completely prevent VS attacks, because an attacker can still circumvent such countermeasures by faking the user interface accordingly. On the one hand, personalisation of web sites (e.g., with logos selected by users) may improve the complexity of VS attacks, because each VS attack has to be adapted to the respective user and may only be possible for man-in-the-middle attackers. On the other hand, however, VS attacks by man-in-the-middle attackers cannot be completely prevented by server-based remedies. Instead, effective countermeasures have to eliminate weaknesses in the user interface. The fundamental weakness enabling VS attacks is the lack of BSCI authentication.

Due to the technical focus of this paper, we concentrate on the technical means to counter VS attacks. Our emphasis is on countermeasures that do not restrict users (e.g., by deactivating widely used features such as JavaScript), because this would strongly limit their acceptance by users. However, we want to stress that the effectiveness of these measures ultimately relies on the security awareness of users. In the following Sections we introduce several complementary concepts to counter VS attacks.

### 6.1 Personalisation

As discussed above, VS exploits vulnerabilities in browsers' SSL integration (displaying SSL meta-information) to simulate a secure connection to a trusted server by displaying fake meta-information. A straightforward method to prevent VS attacks is to implement BSCIs as tamper-resistant components of the user interface, which cannot be deactivated or changed by (active) web languages (at least for SSL connections).

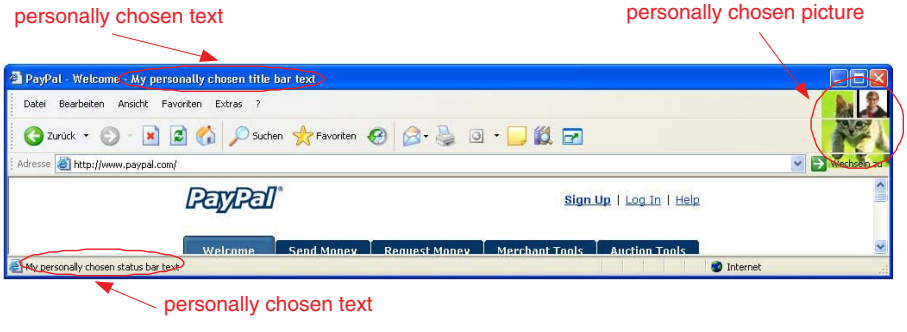



**Fig. 1.** Personalised User Interface. Marked regions are personalised BSCIs

We propose to authenticate BSCIs by applying the concept of personalisation with individually chosen background bitmaps (see Figure 1), as introduced by [16] in the context of trojan horses. We believe that the use of personalised bitmaps, e.g. the picture of the user's pet or friend, is more eye-catching than the proposal of blinking boundaries, while at the same time being less annoying. Compared to the proposal of self-certified logos (see Section 5), the configuration overhead for the user is significantly reduced, because the user only has to select one background image instead of self-certifying a logo for every SSL-enabled web site for which a user wants to counter VS attacks.

This has two advantages: first, the user has a stronger relation to his browser what makes changes more eye-catching. Furthermore, it is improbable that a *remote* attacker is able to determine (and fake) this individual modification. Due to security policies of web languages, the remote attacker is unable to neither figure out the position, nor the concrete background image chosen by the user. Therefore, an attacker can only act on the assumption that his victims use the standard user interface and a personalised user interface would immediately expose this VS attack. Figure 1 shows a screenshot of our demonstrator for Internet Explorer 6.

### 6.2 Independent Authenticated BSCIs

The idea of this concept is to introduce an additional helper application, which collects various relevant information from reliable sources and displays them in a trustworthy and authenticated way (see personalisation concept above). Our prototype implementation collects connection information (e.g., IP address of the server) from the operating system's network stack[5], and status information from the browser. The latter includes meta-information such as the status of activated bars (status bar, menu bar, etc.), URL and a brief summary of the SSL-related information. Based on this summary of security relevant information a user can verify whether the used web page is trustworthy or not. In case of a VS attack, the information displayed by the browser and the information displayed in the independent authenticated BSCI will differ and the VS attack will be noticed by the user.

### 6.3 Semantic and Syntactic Analyser

The main idea of this concept is to analyse the source code of a web page that is to be rendered in order to decide whether it contains VS code or not. The outcome of this analysis can be either used to warn the user, or to block suspicious content completely. However, we want to stress that the quality of syntactic and semantic analysis depends on the acquired knowledge base and that it might be hard to find suitable filtering rules in practice.

[5] Note, that in case of network based mounting attacks, the information in the network stack will contain wrong information. However, this will be noticed, as this information will be different from the certificate information.

## 7 Implementation of Countermeasures

We propose two possible approaches of implementing our concepts. The first implementation combines all concepts in an adaptive web browser toolbar, which summarises all relevant information and allows the user to get this crucial information at a glance. As this toolbar is a local component of the user's system, a remote attacker cannot access it by means of active web languages.[6] The advantage of this implementation is that a user has a permanent and reliable overview about the status of his web connection. Once a user has personalised the browser's GUI (e.g. during installation), users achieve sufficient security against VS attacks. Users only have to verify the web browser's personalisation and the certificate information, which is always displayed. This is why this approach is suitable to protect even naive users. We have implemented a prototype of this toolbar as a Browser Helper Object (BHO)[7] for Microsoft's Internet Explorer Release 6. It can be downloaded at [11].[8]

A disadvantage of the toolbar described above is that its implementation depends on the underlying browser, as it must be integrated in the web browser's user interface. To overcome this drawback, we propose a proxy-based approach: a web proxy runs between the web browser and web site. Before the proxy forwards a requested web page to the browser, it embeds meta-information into the retrieved HTML-document. The web proxy may either operate on corporate gateways, e.g., as part of a firewall, or it may operate as a local software on each client, e.g., as part of a personal firewall. Its task is to summarise information about the connection's status, which is normally displayed by the BSCIs. This information may be displayed in a fixed, unspoofable HTML frame similar to an additional "status bar".

To make this idea even more practicable and secure, we propose to visualise the information as intuitive icons, e.g. a big lock icon, indicating a SSL connection. As the embedded information is encoded as HTML and rendered by the browser, care must be taken that the additional frame is resistant against manipulation by means of active web languages. This can be achieved by filtering selected instructions before forwarding the augmented web page to the browser and personalising the frame to authenticate it. The main advantage is that the HTML-encoded information can be displayed by any HTML web

[6] At least not with reasonable security settings and without asking the user for permission. However, in these exceptional cases, no security is achievable anyway, as an attacker may completely corrupt the user's computing base.

[7] A BHO is a COM-component, which is automatically mounted during the start of the Internet Explorer application.

[8] Alternatively, one could use *Microsoft's Group Policy Editor* (*GPE*), which also allows to customise Internet Explorer's toolbars except the status bar. Thus, users are also able to personalise the browser's GUI. However, as the GPE does not provide means of personalising (authenticating) the status bar, we recommend the installation of Internet Explorer's Service Pack 2, which denies the status bar's deactivation by remote users.

browser, whereas the main drawback of this proposal is that it breaks the end-to-end security of SSL/TLS-connections between the client and the server. However, as the proxy is operated by the user himself or by the company, we do not consider this to be a problem in practice. Our future work aims at analysing this approach in more detail and implementing this web proxy.

## 8 Conclusion

In this paper, we analysed the character of Visual Spoofing (VS) attacks. Visual Spoofing exploits vulnerabilities in the web browser's integration of SSL/TLS, specifically the mechanism for indicating security relevant information, such as the use of SSL and the certified identity of the server, to users. The main reason why VS attacks succeed in today's web browsers is their ability to deactivate those parts of browser's user interface that visualise crucial meta-information and render faked ones instead.

We propose countermeasures, which authenticate those parts of the user interface that display security critical information and include an unspoofable summary of this information. Our countermeasure can be integrated directly in a web browser by means of a Browser Helper Object or by means of a proxy, which enriches retrieved HTML documents with security relevant information. These countermeasures can easily be included in existing web browsers, corporate firewalls or personal security suites.

## References

1. Freier, A.O., Kariton, P., Kocher, P.C.: The SSL Protocol: Version 3.0. Internet draft, Netscape Communications (1996)
2. Dierks, T., Allen, C.: The TLS protocol version 1.0. Internet Request for Comment RFC 2246, Internet Engineering Task Force (1999) Proposed Standard.
3. Schneier, B., Wagner, D.: Analysis of the SSL 3.0 protocol. In: Proceedings of the 2nd USENIX Workshop on Electronic Commerce, Oakland, USA, USENIX Press (1996)
4. Ornaghi, A., Valleri, M.: Man in the middle attacks Demos. In: BlackHat Conference, USA (2003)
5. Felten, E.W., Balfanz, D., Dean, D., Wallach, D.S.: Web Spoofing: An Internet Con Game. In: Proceedings of the 20th National Information Systems Security Conference, Baltimore, USA (1997)
6. Zishuang Eileen Ye, Y.Y., Smith, S.: Web Spoofing Revisited: SSL and Beyond. Technical report tr2002-417, Dartmouth PKI Lab (2002)
7. Li, T.Y., Yongdong, W.: Trust on Web Browser: Attack vs. Defense. In: Proceedings of the International Conference on Applied Cryptography and Network Security, Kunming, China (2003)
8. Herzberg, A., Gbara, A.: Protecting (even) NaiveWeb Users, or: Preventing Spoofing and Establishing Credentials of Web Sites. Internet draft, Bar Ilan University, Computer Science Department (2004)
9. Anti Phishing Working Group: Phishing Attack Trend Report July (2004) http://www.antiphishing.org.

10. Litan, A.: Phishing Victims Likely Will Suffer Identity Theft Fraud. Gartner Research Note (May 14, 2004)
11. Adelsbach, A., Gajek, S., Schwenk, J.: Visual spoofing toolbar (2004) http://www.nds.rub.de/forschung/gebiete/UI/VS/download/visual spoofing toolbar. exe.
12. Adelsbach, A., Gajek, S., Schwenk, J.: Visual Spoofing Demonstrator based on DHTML (2004) http://134.147.40.90, Username:visual, Password:spoofing.
13. Heise News Ticker: eBay konnte Passwortklau nicht verhindern (December 23, 2004) http://www.heise.de/security/news/meldung/print/54605.
14. Ye, Z.E., Smith, S.: Trusted Paths for Browsers. In: Proceedings of the 11th USENIX Security Symposium, San Francisco, USA (2002)
15. Mozilla.org: weak XUL security allows chrome UI spoofing (phishing attack) (2004) https://bugzilla.mozilla.org/show bug.cgi?id=252198.
16. Tygar, J.D., Whitten, A.: WWW Electronic Commerce and Java Trojan Horses. In: Proceedings of the 2nd USENIX Workshop on Electronic Commerce, Oakland, USA, USENIX Press (1996)

# Model Redundancy vs. Intrusion Detection

Zhuowei Li, Amitabha Das, and Sabu Emmanuel

School of Computer Engineering, Nanyang Technological University,
50, Nanyang Avenue, Singapore 639798
zhwei.li@pmail.ntu.edu.sg
{asadas, asemmanuel}@ntu.edu.sg

**Abstract.** A major problem faced by intrusion detection is the intensive computation in the detection phase, and a possible solution is to reduce model redundancy, and thus economize the detection computation. However, the existing literature lacks any formal evaluation of the significance of model redundancy for intrusion detection. In this paper, we try to do such an evaluation. First, in a general intrusion detection methodology, the model redundancy in the behavior model can be reduced using feature ranking and the proposed concept of '*variable-length signature*'. Then, the detection performance of the behavior model before and after model redundancy is compared. The preliminary experimental results show that the model redundancy in the behavior model is useful to detect novel intrusions, but the model redundancy due to the overlapping distinguishability among features is insignificant for intrusion detection.

## 1 Introduction

Intrusion detection becomes more and more indispensable among the security infrastructures to protect our computing resources. Basically, the task of intrusion detection is to detect intrusions from the audit trails left by computing resources. However, with the increasing bandwidth of the Internet and great volume of electronic transactions, the audit trails left for intrusion detection are also increasing. To make matters worse, most intrusion detection techniques are computation intensive, especially for anomaly-based intrusion detection (Debar et al. [1]). Thus, it is a critical issue to decrease the computation in the detection phase for intrusion detection.

In this paper, we will address *the issue of the computational cost of the detection phase* and explore ways to reduce it. In general, the intensive computation can be mitigated if the redundant information in the behavior model is reduced. For example, if we can determine an intrusion (e.g., *portsweep*) with only two features, additional features for intrusion detection other than the two features are redundant, and they will cause additional computation in the detection phase. Thus, it is feasible to lessen the computational intensity by reducing the model redundancy. However, in the past literature, there is no formal evaluation on whether reduced model redundancy will affect the efficiency of the

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 217–229, 2005.

behavior model for intrusion detection. Based on a formal methodology, we do such evaluation in this paper.

The main contributions of this paper are as follows. First, the features are ranked for intrusion detection using a formal methodology, and some features are discarded if they do not contribute to the distinguishability of the behavior model. Then, the concept of variable-length signature is introduced to further reduce model redundancy. Lastly, based on the ranked features and variable-length signatures, a framework is designed to evaluate the model redundancy. The threat to variable-length signatures from mimicry attacks is also analyzed.

The remaining parts are organized as follows. In Section 2, a methodology for intrusion detection is described briefly, and the variable-length signature is introduced. The model redundancy in the behavior model is discussed in Section 3, and a mechanism is designed to reduce the model redundancy. In Section 4, we describe the experiments performed to evaluate model redundancy. The related works are discussed in Section 5, and we conclude the paper in the last section.

## 2 A Formal Intrusion Detection Methodology

In general, the behavior models of the resources for intrusion detection are built using a set of features, or a *feature vector* $FV = \{F_1, F_2, \ldots, F_n\}$, where $F_i$ is a nominal, discrete or continuous feature. Based on whether it is found in normal and/or intrusive audit trails, a feature value of any feature in the feature vector is labeled as *normal*, *suspicious* or *anomalous*. More specifically, if a feature value occurs only in the normal audit trails, it is a *normal* feature value. If it occurs only in the intrusive audit trails, for example, in the signatures of SID, it is labeled as *anomalous*. Otherwise, i.e., if it occurs in the normal and intrusive audit trails, it is labeled as *suspicious*. For brevity, we refer to the normal, suspicious, or anomalous label as the **NSA label** of a feature value.

### 2.1 Feature Ranges

In the feature space of discrete/continuous features, the range of values between any two feature values will be called *a feature range*. The concept of NSA labels (i.e., normal, suspicious and anomalous) can be extended to feature ranges as follows. For continuous and discrete features, if two neighboring (c.f. Section 3) feature values have identical NSA label, then the intervening range will also have the same NSA label as the two end points. This label will be valid as long as no known feature value emerges with a different NSA label within that range. On the other hand, if two neighboring feature values have different NSA labels, then the intervening range will be divided into two subranges with respect to a user-define strategy, and each subrange will be assigned the NSA label of the original feature value associated with that subrange. For nominal features, the concept of feature range is not applicable, but, for the sake of uniformity of description, each nominal feature value is also referred to as a feature range with its corresponding NSA label.

Based on the NSA labels of feature ranges, the feature space $Def(F)$ can be split into feature subranges $\{R_F^1, R_F^2, \ldots, R_F^m\}$, such that the neighboring feature ranges (such as $R_F^j$ and $R_F^{j+1}$) have different NSA labels. Finally, by grouping the feature ranges with identical NSA labels, we can partition the feature space into three feature subspaces: normal, suspicious and anomalous. We will denote the normal feature subspace of a feature $F$ as $N(F)$, in which all its feature ranges come from the normal audit trails. Its suspicious and anomalous feature subspaces are denoted as $S(F)$ and $A(F)$ respectively. Thus,

$$N(F) = \{R_F^j |\ 1 \le j \le m,\ label(R_F^j) = \text{`}normal\text{'}\}$$
$$S(F) = \{R_F^j |\ 1 \le j \le m,\ label(R_F^j) = \text{`}suspicious\text{'}\}$$
$$A(F) = \{R_F^j |\ 1 \le j \le m_i,\ label(R_F^j) = \text{`}anomalous\text{'}\}$$

where, $label(\ldots)$ returns the NSA label of a feature range. In the following description, we denote $\Omega(F) = N(F) \cup S(F) \cup A(F)$.

*Example 1.* Let us assume that there are three features $F_1$, $F_2$ and $F_3$, $FV = \{F_1, F_2, F_3\}$. The example feature instances are listed in Table 1, in which 'A', 'B', 'C' and 'D' represent four different intrusions, and 'N' represents normal behaviors. The feature ranges for every feature are listed in (b).

**Table 1.** A simple example

(a) Example instances.

| $F_1$ | $F_2$ | $F_3$ | STATUS |
|---|---|---|---|
| TCP | 1 | 0.01 | N |
| ICMP | 2 | 0.04 | N |
| NETBIOS | 6 | 0.10 | C |
| TCP | 4 | 0.08 | A |
| UDP | 5 | 0.06 | B |
| UDP | 8 | 0.14 | D |
| NETBIOS | 6 | 0.10 | N |
| UDP | 7 | 0.02 | N |
| UDP | 8 | 0.14 | B |

(b) Example feature ranges.

| FEATURE | INDEX | FEATURE RANGES | STATUSES | NSA LABEL |
|---|---|---|---|---|
| $F_1$ | $R_{F_1}^1$ | TCP | N,A | 'suspicious' |
| | $R_{F_1}^2$ | ICMP | N | 'normal' |
| | $R_{F_1}^3$ | UDP | N,B,D | 'suspicious' |
| | $R_{F_1}^4$ | NETBIOS | N,C | 'suspicious' |
| $F_2$ | $R_{F_2}^1$ | [1,2] | N | 'normal' |
| | $R_{F_2}^2$ | [3,5] | A,B | 'anomalous' |
| | $R_{F_2}^3$ | [6,6] | N,C | 'suspicious' |
| | $R_{F_2}^4$ | [7,7] | N | 'normal' |
| | $R_{F_2}^5$ | [8,8] | B,D | 'anomalous' |
| $F_3$ | $R_{F_3}^1$ | [0.01,0.05) | N | 'normal' |
| | $R_{F_3}^2$ | [0.05,0.09) | A,B | 'anomalous' |
| | $R_{F_3}^3$ | [0.09,0.14] | N,C,D | 'suspicious' |

## 2.2 Compound Features

The concept of NSA labels and subspace partitioning can easily be extended to more than one feature. For convenience, we will refer to a single feature as an *atomic feature* and a combination of multiple features as a *compound feature*. We formally define a compound feature as follows:

**Definition 1 (compound feature).** *For any two features $F_1$ and $F_2$, a compound feature $F_{12}$ is defined as a subset of the cartesian product of $\Omega(F_1)$ and $\Omega(F_2)$, such that each element in this set actually represents some audit trails.*

*In other words, if the ordered pair* $(a, b) \in \Omega(F_{12})$*, then the compound feature range* $(a, b)$ *must identify some training audit trails.*

$$\Omega(F_{12}) = \{(a,b) | a \in \Omega(F_1), b \in \Omega(F_2), (a,b)\ identifies\ some\ audit\ trails\}$$

*For convenience,* $F_{12} = F_1 \times F_2$.

*Example 2.* Using the example instances in Section 2.1, for $F_{23} = F_2 \times F_3$,

- $R^1_{F_{23}} = R^1_{F_2} \times R^1_{F_3}, N \Rightarrow$ '*normal*';
- $R^2_{F_{23}} = R^2_{F_2} \times R^2_{F_3}, A, B \Rightarrow$ '*anomalous*';
- $R^3_{F_{23}} = R^3_{F_2} \times R^3_{F_3}, N, C \Rightarrow$ '*suspicious*';
- $R^4_{F_{23}} = R^4_{F_2} \times R^1_{F_3}, N \Rightarrow$ '*normal*';
- $R^5_{F_{23}} = R^5_{F_2} \times R^3_{F_3}, B, D \Rightarrow$ '*anomalous*';

and, for $F_{123} = F_1 \times F_{23}$ (i.e., the signature base in USAID[3]),

- $R^1_{F_{123}} = R^1_{F_1} \times R^1_{F_{23}}, N \Rightarrow$ '*normal*';
- $R^2_{F_{123}} = R^2_{F_1} \times R^1_{F_{23}}, N \Rightarrow$ '*normal*';
- $R^3_{F_{123}} = R^1_{F_1} \times R^2_{F_{23}}, A \Rightarrow$ '*anomalous*';
- $R^4_{F_{123}} = R^3_{F_1} \times R^2_{F_{23}}, B \Rightarrow$ '*anomalous*';
- $R^5_{F_{123}} = R^4_{F_1} \times R^3_{F_{23}}, N, C \Rightarrow$ '*suspicious*';
- $R^6_{F_{123}} = R^3_{F_1} \times R^4_{F_{23}}, N \Rightarrow$ '*normal*';
- $R^7_{F_{123}} = R^3_{F_1} \times R^5_{F_{23}}, B, D \Rightarrow$ '*anomalous*';

Intuitively, similar to atomic features, the feature ranges of the new compound feature $F_{12}$ can also be labeled as normal, suspicious and anomalous ones, i.e., they also have the NSA labels. Furthermore, just like the atomic feature space, the feature space of a compound feature can also be partitioned into three feature subspaces, i.e., $\Omega(F_{12}) = N(F_{12}) \cup S(F_{12}) \cup A(F_{12})$.

Since the compound feature built from two atomic features shows the same property as any of its component atomic feature, it can be treated like an atomic feature to build higher order compound features. Using this recursive procedure, the feature vector $FV$ can be converted into an equivalent n-order compound feature $F_{1...n}$ with subspaces $N(F_{1...n})$, $S(F_{1...n})$ and $A(F_{1...n})$.

### 2.3 Variable-Length Signatures

As stated above, for *any* (atomic/compound) feature, there are normal, suspicious and anomalous feature ranges in its feature space. In other words, the feature can distinguish some behaviors falling into normal and anomalous feature ranges as normal or anomalous, which is the task of intrusion detection. Therefore, in intrusion detection, the special characteristic of the feature can be utilized to reduce the number of features in the detection phase.

**Definition 2 (Signature).** *Assuming that there exists a feature vector* $FV = \{F_1, F_2, \ldots, F_n\}$*, the feature ranges of every feature are determined beforehand. A* `signature` *is a feature range of* $F_{1...n}$*. In other words, the signature is the combination of feature ranges of all features in the feature vector.*

*Example 3.* In the example of Section 2.1, one signature is $(ICMP, [1, 2], [0.01, 0.05])$. The total number of possible signatures in this example is 4*5*3=60.

**Definition 3 (Variable-Length Signature).** *Given a feature vector* $FV = \{F_1, F_2, \ldots, F_n\}$ *with labelled feature ranges for every feature, a* `variable-length signature` *is a feature range that is constituted by the first* $i$ *features* $(i \leq n)$ *from* $FV$*, such that its NSA label is either 'normal' or 'anomalous'. The signature is called variable as the number of features involved in it can vary from 1 to* $n$*.*

*Example 4.* As in above Example 3, for the same signature $(ICMP, [1, 2], [0.01, 0.05])$, considering that the NSA label of the feature range $(ICMP)$ is 'normal', $(ICMP)$ is a '*normal*' variable-length signature. $(TCP, [3, 5])$ is also an '*anomalous*' variable-length signatures of the intrusion 'A'.

Among the relations between signatures and variable-length signatures, a 'normal' or 'anomalous' signature will include at least one variable-length signature, but a 'suspicious' signature does not include any variable-length signature. Thus, any variable-length signature reduces this redundancy in its corresponding signature without sacrificing the distinguishability.

## 3 Model Redundancy in the Behavior Model

In the behavior model, *model redundancy* is the behavior information that can be discarded without loss of signature distinguishability. There are two ways to reduce model redundancy of the behavior model in our methodology. First, due to the overlapping distinguishability among features, some features can be discarded in all signatures. Secondly, for some signatures, some features can be further discarded to form a variable-length signatures.

### 3.1 A Mechanism for Reducing Model Redundancy

Roughly speaking, the model redundancy in signatures can be reduced as follows. First the significance of every feature is evaluated and a sorted feature vector is identified from the original feature vector. Using the sorted feature vector, the signature building procedure constructs a so-called NSA signature base, in which the model redundancy is reduced. Lastly, a detection mechanism is designed to evaluate the usefulness of model redundancy in the behavior model. In addition, two labelled datasets are needed in our evaluation: a training audit trails and a test audit trails. The two datasets are labelled correctly with corresponding statuses (e.g., 'normal', 'Nimda', 'MSBlast').

### 3.2 Feature Subspaces Extraction

**Step 1: feature value collection.** In the labeled training audit trails, the feature values are collected for every feature. The statuses (*normal and/or intrusions*) of every feature value are also collected and stored in its status list.

Based on its status list, every feature value is assigned an NSA label. Note that this step is applied to nominal, discrete and continuous features, but the following two steps are only applicable to discrete and continuous features.

**Step 2: feature value clustering.** The objective of this step is to form initial feature ranges for every feature by clustering the neighboring feature values. For a discrete feature, two feature values $x_1$ and $x_2$ are *neighboring* if $|x_1 - x_2| = 1$[1]. For a continuous feature, two feature values $x_1$ and $x_2$ are neighboring if $|x_1 - x_2| \leq \delta$. If several neighboring feature values have the same NSA label, they will be combined to form an *initial feature range*. As a special case, if, for a feature value, its neighboring feature values have different NSA labels from itself, it forms an initial feature range whose upper and lower bounds are both equal to the feature value itself. Every initial feature range thus formed inherits the NSA label of the feature values falling within it.

**Step 3: feature range generalization.** However, under most scenarios, the initial feature ranges of a feature will not cover all of its feature space. Any feature subspace outside the labelled feature ranges is named as an *uncovered subspace*. Comparing to the neighboring definition of feature values, the neighboring of feature ranges is defined as follows. Two feature ranges are *neighboring* if there is no other feature range between them. Thus, the uncovered subspace between any two neighboring feature ranges is processed as follows: if the two feature ranges have the same NSA label, a new feature range will be formed to cover the two feature ranges as well as the uncovered subspace; otherwise, the uncovered space is divided and allocated to these two defined feature ranges using *a predefined splitting strategy*. The NSA labels of the initial feature ranges will be inherited by the newly extended or combined feature ranges.

In this paper, an uncovered space will be shared by its neighbors equally. Ultimately, all the *known* feature space of every feature will be covered by well-defined feature ranges.

### 3.3 A Sorted Feature Vector

**Notations.** Given a compound feature $F_{1...i}$, whose normal, suspicious and anomalous feature subspaces are $N(F_{1...i})$, $S(F_{1...i})$ and $A(F_{1...i})$ respectively, and an atomic feature $F_{i+1}$. Basically, it is possible that a feature range labelled 'anomalous' is caused by several intrusions, but the feature range due to a single intrusion is extremely valuable for identifying that intrusion, and is thus highly desirable. Therefore, $A(F_{1...i})$ is further split into two feature subspaces $An(F_{1...i})$ and $A1(F_{1...i})$. Specifically, if the feature range in $A(F_{1...i})$ is caused by only one intrusion, it will classified as $A1(F_{1...i})$, otherwise, as $An(F_{1...i})$. In summary, there are four feature subspaces for $F_{1...i}$: $N(F_{1...i})$, $S(F_{1...i})$, $An(F_{1...i})$, and $A1(F_{1...i})$, and for $F_{1...i+1}$: $N(F_{1...i+1})$, $S(F_{1...i+1})$, $An(F_{1...i+1})$, and $A1(F_{1...i+1})$ respectively.

[1] If the distance is larger than 1, there is an uncovered space $|x_1 - x_2 - 1|$.

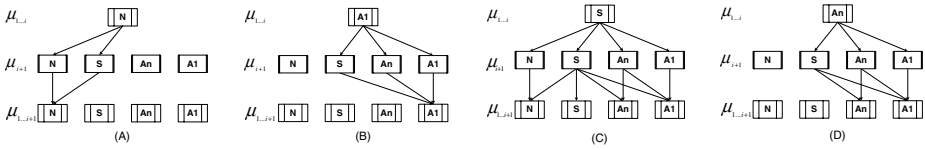



**Fig. 1.** Four compounding scenarios

**Measuring and Sorting Features.** In Figure 1, the four compounding scenarios are illustrated to design a measure for feature selection. As indicated in (A), the compounding between $N(F_{1...i})$ and feature subspaces of $F_{i+1}$ will result in feature ranges in $N(F_{1...i+1})$. Thus, there is no increment of the distinguishability for intrusion detection. Similarly, (B) will not lead to any benefit either. In contrast, the compounding scenarios (C) and (D) will increase the distinguishing ability of $F_{1...i+1}$ as some of their resultant feature ranges fall within $N(F_{1...i+1})$ or $A1(F_{1...i+1})$. As a result, to increase the distinguishability of $F_{1...i+1}$, the increment due to compounding scenarios (C) and (D) should be maximized.

Considering that our ultimate objective is to reduce the computation overhead in the detection phase, it is useful to consider frequency information of every feature range. For this reason, we assume that the training audit trials have the same frequency distribution to the test audit trails, thus the detection computation will be reduced. With respect to the compounding operation between $F_{1...i}$ and $F_{i+1}$, the following notations are defined.

- $\#S2N(F_{1...i}, F_{i+1})$ is the number of instances falling into $S(F_{1...i})$ and $N(F_{1...i+1})$.
- $\#S21(F_{1...i}, F_{i+1})$ is the number of instances falling into $S(F_{1...i})$ and $A1(F_{1...i+1})$.
- $\#n21(F_{1...i}, F_{i+1})$ is the number of instances falling into $An(F_{1...i})$ and $A1(F_{1...i+1})$.

$newIdentify(F_{i+1}|F_{1...i}) = \#S2N(F_{1...i}, F_{i+1}) + \#S21(F_{1...i}, F_{i+1}) + \#n21(F_{1...i}, F_{i+1})$

The feature evaluation (Algorithm 1) works as follows. Initially, $FV_{sort}$ is empty. In $i$-th step, all features outside $FV_{sort}$ are evaluated by $newIdentify$, and the feature with the maximal value will be appended to $FV_{sort}$ as the last element. Obviously, the feature with maximal distinguishability is selected.

**Algorithm 1** Feature sorting and selecting for intrusion detection

**Require:** Feature Vector $FV$, Number of Features $n$, Training audit trails $\Sigma_{tr}$;
1: Selecting one feature $F_{max}$ with the maximal size of instances in $\Sigma_{tr}$ falling into $N(F_k)$ and $A1(F_k)$ $(1 \leq k \leq n)$;
2: Inserting $F_{max}$ into $FV_{sort}$;
3: **for** $i = 2$ to $n$ **do**
4: maxIdentify = -∞; maxIdentifyFeature = -1;
5: **for** $j = 1$ to $n$ **do**
6: **if** $F_j \in FV_{sort}$ **then** continue;
7: Building the compound feature by compounding $F_j$ and the features in $FV_{sort}$;
8: Calculating $newIdentify(F_j|F_{FV_{sort}})$ from $\Sigma_{tr}$;
9: **if** newIdentify>maxIdentify **then** maxIdentify=newIdentify; maxIdentifyFeature=j; **endif**
10: **end for**
11: Inserting $F_{maxIdentifyFeature}$ into $FV_{sort}$;
12: **end for**
13: Output $FV_{sort}$;

### 3.4 Building NSA Signature Base

The NSA signature base building process starts with the first feature in $FV_{sort}$, and includes one additional feature in each step in the sorted order. Finally, after compounding with the last feature, the compounding feature ranges in $S(F_{FV_{sort}})$ and $An(F_{FV_{sort}})$ are left as signatures in the signature base. Algorithm 2 describes it in detail.

**Algorithm 2** Building NSA signature base

**Require:** $FV_{sort}$ and its size $m$, Training audit trails;
NSABase=$\Phi$;
**for** $i = 1$ to $m$ **do**
  Building the compound feature $F_{1...i}$ in $FV_{sort}$;
  **for** each feature range in $N(F_{1...i})$ and $A1(F_{1...i})$ **do**
    **if** it does not match the signatures in NSABase **then**
      Inserting it into NSABase as a variable-length signature;
    **end if**
  **end for**
**end for**
Inserting $S(F_{1...m})$ and $An(F_{1...m})$ to NSABase;
Output NSABase;

**Table 2.** Several example entries in the NSA signature base

| VARIABLE-LENGTH SIGNATURES | NSA | INTRUSIONS OR NORMAL |
|---|---|---|
| [94,A] | A | portsweep |
| [1069,N] | N | normal |
| [2,S]+[214,S] | A | buffer_overflow |
| [0,S]+[182,S]+[84,N] | N | normal |
| [0,S]+[86,S]+[0,S] | A | smurf |
| [0,S]+[277,S]+[239,S]+[14,S] | A | back |

Table 2 shows several example entries in the NSA signature base. In every entry, the feature range is denoted by its index, and its NSA label is paired with the index using a pair of square brackets. The compounding of the feature ranges is denoted by '+'. For instance, in '[2,S]+[214,S]', '[2,S]' refers to the second range of the first feature in $FV_{sort}$ (which is '*source bytes*', please see $FV_{sort}$ in the appendix), and 'S' indicates that its NSA label is 'suspicious'. The whole expression, along with its NSA label and status, constitutes a complete variable-length signature "[2,S]+[214,S]=>A:buffer_overflow".

Obviously, if there exist '*suspicious*' signatures in the NSA signature base (i.e., $S(F_{1...m}) \neq \Phi$ in Algorithm 2), it implies that the feature vector is not enough to completely distinguish the intrusive and normal behaviors in the audit trails. Because of the ambiguity in the suspicious signatures, they will lead to detection errors (*false alarms or false negatives*) for intrusion detection.

**Definition 4 (Signature Variation).** *In the signature base of a computing resource, if (variable-length) signatures A and B have the same status list, A is called the signature variation of B, and vice versa.*

It is obvious that the signature variations of an intrusion indicate the existence of intrusion variations, which lead to the well-known detection difficulty in SID techniques. In general, the number of signature variations in the signature base indicates the diversity of the corresponding intrusions.

### 3.5 Detection Mechanisms via Variable-Length Signatures

In the detection phase, the feature ranges of every feature in the sorted feature vector and the NSA signature base are used to detect anomalies in the test or real-time audit trails. First, the feature instance is extracted from the audit trails. Then, a corresponding variable-length signature is generated incrementally by mapping of the feature values one at a time. The signature thus generated is compared with the variable-length signature base, and if a match is found, the instance is identified by the matched entry and the remaining features are not used. If there is no such match using all features, the instance is '*anomalous*'.

### 3.6 Mimicry Attacks

The threat from mimicry attacks (introduced by Wagner et al.[6]) can be explained as follows. In USAID[3], a mimicry attack achieve its goal by mimicking all the feature ranges of a 'normal' signature. In contrast, a mimicry attack to variable-length signatures can be designed more easily than USAID since the short 'normal' variable-length signatures will be the preference to attackers. In other words, variable-length signatures become more vulnerable than USAID.

## 4 Experiments

We have chosen a typical dataset for network intrusion detection from KDD CUP 1999 contest, in which every record is an instance of a specific feature vector collected from the audit trails. This is because the dataset meets the requirements of our methodology: *labeled audit trails and intrusion-specific feature vector*. For a detailed description of the datasets, please refer to '*http://www-cse.ucsd.edu/users/elkan/clresults.html*'. As the precision of a continuous feature is 0.01 in the dataset, we let $\delta = 0.01$.

### 4.1 Experimental Results

In our following sections, we will focus on mining the sorted feature vector, building the NSA signature base, and evaluating the model redundancy. To save space, we would refer the reader to [3] for the details of the feature ranges.

**Mining a Sorted Feature Vector.** Figure 2 illustrates the percentage of identified instances in the audit trails with more features being added into the sorted feature vector. Obviously, the curve converges to a stable value as more features

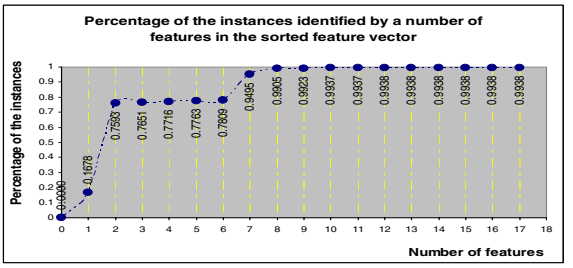



**Fig. 2.** The distinguishability tendency in mining the sorted feature vector

are included into the sorted feature vector, i.e., more instances in the training audit trails are identified by these features. The stopping criterion for feature selection is defined when, for any feature $F_k$ in our applied feature vector, $newIdentify(F_k|F_{FV_{sort}}) = 0$. That is, there are no instances in the training audit trails will be identified by adding a further feature. Using this criterion, (m=)17 features are selected into the sorted feature vector, i.e., (n-m=)24 features are discarded before continuing with further experiments (for details, c.f. the sorted feature vector in the appendix).

**Building the NSA Signature Base.** With respect to the sorted feature vector and the feature ranges for every feature, every instance in the labeled audit trails is evaluated as a variable-length signature. Then, these variable-length signatures

**Table 3.** The statistics of NSA signature base

| ITEM | $N(\dots)$ | $S(\dots)$ | $An(\dots)$ | $A1(\dots)$ | TOTAL |
|---|---|---|---|---|---|
| Size of NSA base with 41 features | 60371 | 43 | 15 | 2779 | 63208 |
| Size of NSA base with variable features | 952 | 43 | 15 | 1072 | 2082 |
| Average length of variable-length signatures | 5.79 | 17 | 17 | 5.53 | 5.97 |

constitute the NSA signature base, and thus the model redundancy is reduced in comparison with the NSA signature base with constant-length signatures. Table 3 lists the statistics of the variable-length signatures in the NSA signature base. In summary, the storage space of NSA signature base using variable-length signatures (i.e., the model redundancy) is cut down by 99.52% in comparison with constant-length signatures. At the same time, the non-empty signature set $S(\dots)$ and $An(\dots)$ indicates that the information in the feature vector is not enough to distinguish all the behaviors in the training audit trails.

Table 4 summarizes the signature variations in the NSA signature base. It is clear that the same intrusion can lead to multiple variable-length signatures, and the number of these signature variations indicate the diversity of the intrusion and thus the difficulty to detect these intrusions. At the same time, it is worth noting that the most diverse behavior is the 'normal' behavior, which has the largest number of 'normal' signature variations. This phenomenon can

**Table 4.** Number of signature variations in the training audit trails

| NAME / LENGTH | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| normal | 141 | 214 | 28 | 29 | 62 | 37 | 67 | 4 | 76 | 265 | 1 | 17 | 5 | 2 | 0 | 0 | 8 |
| neptune | 1 | 0 | 23 | 45 | 0 | 147 | 3 | 43 | 9 | 0 | 0 | 7 | 3 | 0 | 1 | 1 | 0 |
| buffer-overflow | 1 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| portsweep | 2 | 0 | 84 | 48 | 7 | 66 | 2 | 37 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 6 |
| warezclient | 3 | 19 | 0 | 1 | 0 | 2 | 7 | 0 | 3 | 37 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| back | 19 | 8 | 3 | 10 | 3 | 2 | 0 | 0 | 12 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 |
| pod | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| teardrop | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| rootkit | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| smurf | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| phf | o | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| warezmaster | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| multihop | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| satan | 0 | 8 | 19 | 102 | 1 | 43 | 1 | 12 | 1 | 13 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| imap | 0 | 3 | 0 | 1 | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ftp-write | 0 | 5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ipsweep | 0 | 1 | 2 | 1 | 6 | 12 | 20 | 0 | 2 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| nmap | 0 | 0 | 52 | 0 | 0 | 1 | 3 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| guess-passwd | 0 | 0 | 1 | 1 | 8 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| land | 0 | 0 | 0 | 1 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 0 |
| load module | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| spy | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| perl | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

be explained as follows. In the normal usage of a computing resource, there are many tasks to do with different objectives, and different tasks lead to different signatures. However, for an intrusion, the objective of the task is simple comparatively to attack and destroy a computing resource, so the intrusive task is not complicated as normal tasks. Therefore, an intrusion behavior will cause fewer signatures than 'normal' behavior.

**Evaluating Model Redundancy.** To evaluate the significance of model redundancy for intrusion detection, three experiments are done using *constant-length signatures with 41 features* (**SIG-41**), *constant-length signatures with the selected 17 features* (**SIG-17**), and *variable-length sigantures with the sorted feature vector* (**varSIG**) respectively. In SIG-41, all the model redundancy is included in the NSA signature base, but some model redundancy due to overlapping distinguishability among features are eliminated in SIG-17, and lastly, the model redundancy in the signatures are further eliminated in varSIG. In Table 5[2], they are compared with respect to four efficiency parameters: (1) FAR: false alarm rate; (2) DR-Known: detection rate for known intrusions; (3) DR-New: detection rate for new intrusions; (4) Num-FR: number of feature ranges in detecting an instance.

**Table 5.** Efficiency parameters for intrusion detection

| PARAMETERS | FAR (%) | DR-Known (%) | DR-New (%) | Num-FR |
|---|---|---|---|---|
| SIG-41 | 1.451 | 99.779 | 98.184 | 41 |
| SIG-17 | 1.409 | 99.778 | 97.901 | 17 |
| varSIG | 0.431 | 95.928 | 55.172 | 3.205 |

[2] Similar to USAID with constant-length sigantures [3], two new intrusions, namely, snmpgetattack and mailbomb, are eliminated from the detection results.

In Table 5, the negligible difference between detection efficiency (i.e., the same detection rate and false alarm rate) for SIG-41 and SIG-17 indicates that our feature selection is effective, and that the model redundancy due to the overlapping distinguishability among features can be discarded. Even though the detection rate of varSIG is lower than SIG-41 and SIG-17, it is still encouraging because it achieves much lower false alarm rate and significantly lights computation overhead (i.e., smaller Num-FR).

The reason for lower detection rate in varSIG can be explained as follows. For every constant-length signature in the NSA signature base, there is one corresponding variable-length signature, in which some feature ranges are discarded for compactness. However, some intrusions will only cause anomalies in the discarded feature ranges. For example, assuming that a constant-length signature is "[0,N]+[14,S]+[23,S]=>N:Normal" and the corresponding variable-length signature is "[0,N]=>N:Normal". If the signature of an instance is "[0,N]+[25,S]+[23,S]", it will be detected as 'normal' by the variable-length signature but as 'anomaly' by the constant-length signature. For this reason, although it leads to intensive computation in the detection phase, the model redundancy in a signature, which can be reduced in a variable-length signature, is critical to enhance the efficiency in detecting novel intrusions. Conversely, the significance of model redundancy is rooted in the incompleteness of the behavior model (e.g., novel intrusions).

## 5 Related Work

Even though feature selection is well known as a critical step for intrusion detection [2], most research on AID techniques utilize expert knowledge to select the features manually [4], and there are only a few reported work relating to feature evaluation for intrusion detection [5] [2]. In [5], feature ranking is done by evaluating the whole performance for intrusion detection. In our methodology, feature evaluation is done by maximizing the feature distinguishability between normal and anomalous signatures only in the training phase. In MADAM ID [2], the statistical patterns are first mined from the network audit data, and then these patterns are used as guidelines to select features for intrusion detection. Unfortunately, the statistical patterns may not reflect the audit trails completely, and there is no guarantee that all feature distinguishability will be used in the detection phase. However, in this paper, as the stopping criteria for the feature selection is that the distinguishing ability will not change even with more features, all feature distinguishability in the feature vector will be included in the sorted feature vector.

## 6 Conclusions and Future Work

In this paper, we try to evaluate the usefulness of model redundancy in the behavior model for intrusion detection. To achieve it, the concept of variable-length signatures and a feature selection methodology are proposed to reduce the redun-

dancy in the behavior model. Our preliminary experimental results show that the model redundancy due to the overlapping distinguishability among features are insignificant for intrusion detection. Even though the model redundancy cut down by variable-length signatures is critical in detecting novel intrusions, it can lead to many benefits: lowering the false alarm rate, compacting the behavior model and significantly lightening the detection computation overhead.

There are still several problems left unsolved, and they will be addressed in our future work. (1) Adjusting the degree of redundancy in the signature to enhance the detection performance; (2) Whether the discarded features will evolve into critical ones in the future or under other environments; (3) Selecting the features considering the cost of a feature to be changed in mimicry attacks.

## References

1. H. Debar, M. Dacier, and A. Wespi. A revised taxonomy for intrusion detection systems. *Annales des Telecommunications*, 55(7–8):361–378, 2000.
2. W. Lee and S.J. Stolfo. A framework for contructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security*, 3(4):227–261, Nov. 2000.
3. Zhuowei Li and Amitabha Das. Unifying anomaly-based and signature-based intrusion detection. Technical Report CAIS-TR-2004-005, CAIS, Aug. 2004.
4. M.V. Mahoney and P.K. Chan. Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks. In *SIGKDD 2002*, July 23–26 2002.
5. S Mukkamala and A H. Sung. Identifying significant features for network forensic analysis using artificial intelligent techniques. *International Journal of Digital Evidence*, 1(4):1–17, Winter, 2003.
6. David Wagner and Paolo Soto. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 255–264, 2002.

**The Sorted Feature Vector in Our Experiments**

$FV_{sort}$ = {Source Bytes, Service, Flag, REJ Error Rate, Destination Host Service REJ Error Rate, Same Service Rate, Destination Host Service Different Host Rate, Different Service Rate, Service Different Host Rate, Destination Bytes, Protocol Type, Service SYN Error Rate, Destination Host Service SYN Error Rate, Logged In, Land, SYN Error Rate, Duration}.

# An Open Approach for Designing Secure Electronic Immobilizers

Kerstin Lemke, Ahmad-Reza Sadeghi, and Christian Stüble

Horst Görtz Institute,
Ruhr-Universität Bochum, Germany
{lemke, sadeghi, stueble@crypto.rub.de}

**Abstract.** The automotive industry has developed electronic immobilizers to reduce the number of car thefts since the mid nineties. However, there is not much information on the current solutions in the public domain, and the annual number of stolen cars still causes a significant loss. This generates other costs particularly regarding the increased insurance fees each individual has to pay.

In this paper we present a system model that captures a variety of security aspects concerning electronic immobilizers. We consider generic security and functional requirements for constructing secure electronic immobilizers. The main practical problems and limitations are addressed and we give some design guidance as well as possible solutions.

**Keywords:** Electronic Immobilizer, Transponder, Motor Control Unit, RFID, Mafia Attack, Distance Bounding, Trusted Computing.

## 1 Introduction

Since the mid nineties, authorities, insurance companies and automotive manufacturers have put much effort in decreasing the number of car thefts in Europe by using electronic immobilizers.[1] An immobilizer system allows the owner of an ignition key to start the car engine. Certainly, improvements have been achieved against car theft through deployment of electronic immobilizers (see, e.g., [16, 1, 12]) but also due to better co-operation between authorities in different countries. However, skilled and determined thieves can still overcome the electronic immobilizer systems ([16]), e.g., through applying advanced attacks such as manipulating the control software of the engine just by using the diagnostic interface.[2] Further, [16] addresses several organizational weak points: the development and production of electronic immobilizers is not sufficiently secured and the trade of diagnostic devices (including the technical details for electronic immobilizers) cannot be sufficiently controlled due to the annulment of the 'group exemption ordinance'.

[1] For instance Germany is one of the European countries with a high number of stolen cars. This number was 144.057 in 1993 and is reduced to 57.402 in 2002 ([16]).

[2] e.g., around 200 diagnostic devices are currently missing in Germany [4].

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 230–242, 2005.

Unfortunately, there is not much technical information about immobilizers publicly available, and details on the current solutions are rarely known, or only some insights are given.[3] As the value loss of stolen cars is large, and this leads to other high costs particularly regarding the additional insurance fees each one of us has to pay, it is worth to reconsider and improve the security of electronic immobilizers.

Based on cryptographic and security measures this contribution aims at providing an "open" approach starting from the functional and security requirements on electronic immobilizer systems down to implementation issues. We point out some practical problems, give design rules and discuss some solutions and open issues concerning electronic immobilizers.

## 2 System Model

The general model with its components, involved principals, the interfaces between these components and the possible channels to these principals is illustrated in Figure 1. The principals involved are the vehicle manufacturer $M$, the car owner $O$, workshops $W$ (approved by the vehicle manufacturer), control authorities $A$, insurance companies $I$ as well as trusted third parties.

The electronic immobilizer is embedded in the vehicle's electronics, and consists of three components: The *transponder* $T$, which is integrated in the *ignition key* of the car, proves its identity towards the *Motor Control Unit* (MCU) that controls the motor engine. The *ignition lock* mainly acts as an interface (e.g., a contactless reader) between transponder and motor control unit, but it can implement some auxiliary functions like a mechanical lock. The communication between the reader and the transponder is radio frequency (RF) based. The transponder obtains its power by the inductive coupling with the RF field that is produced by the reader.

In the following we only briefly consider the involved parties and the trust relations among them. These aspects and the infrastructure required are not the subject of this contribution since our focus concerns the functional and security aspects of electronic immobilizers.

### 2.1 Trust Relationships

The trust relationships between these parties are very different due to their different interests, and can be very complex. The interests of these parties are manifold: both manufacturers and insurances may tolerate a certain threshold on the number of stolen cars. Beyond this threshold, insurances may react just by adjusting their loss risk, manufacturers may decide to invest into an improved development of electronic immobilizers to decrease the costs for car insurance or for publicity reasons.

[3] e.g., by Texas Instruments [12]. Their solution is based on RFID technology and implements a mutual authentication where the underlying cryptographic algorithm used is a proprietary stream cipher.

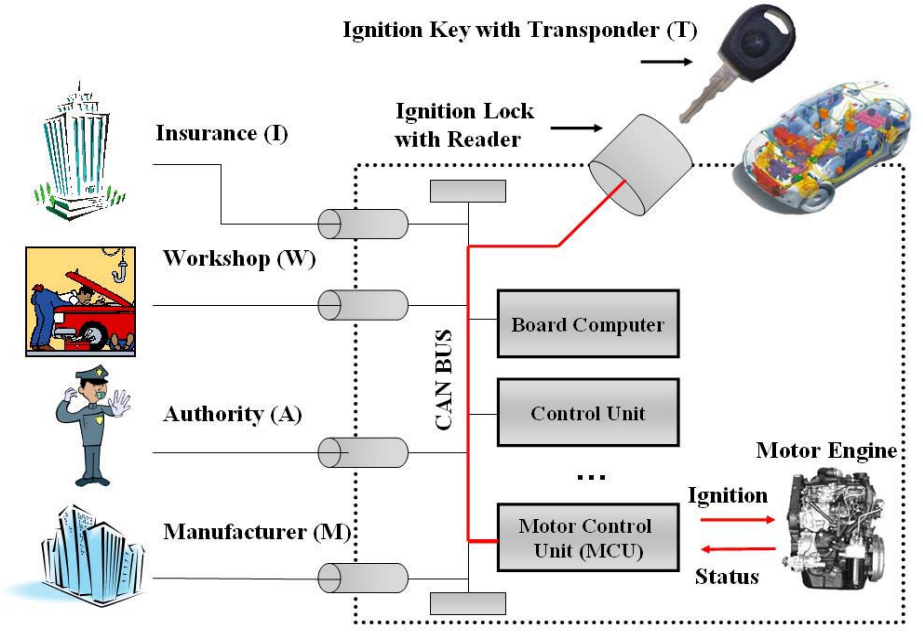



**Fig. 1.** Infrastructure of the system model

Car owners expect an optimal car theft protection system, including both technical and organizational measures. Car owners and workshops are able to physically access the car components during its operation. One generally assumes that the skills (knowledge and tools) of the workshop employees allow more specific attacks. Manufacturers may be mostly trusted by car owners, while workshops may not. Control authorities are not driven by self-interest and are assumed to play according to the rules to minimize car thefts.

In general, indirect relations (involving a third party) may arise among two principals which lead to more complex relationships. One may consider only two levels of trust, namely full, meaning that a principal is trusted by all other parties, or partial, meaning that this principal is partially trusted by other with respect to certain actions.

The control authority $A$ is the principal who is fully trusted by the other involved principals, but $A$ trusts the other ones only partially. All other trust relations are considered to be only partially.

Primarily, malicious actions are imaginable on behalf of the owner and the workshop. Car thefts can also be made easier because of information leakage at the manufacturer. The car owner is the 'weakest' principal involved who risks to be accused both for the modification of components and/or the co-operation with car thieves. The infrastructure and the corresponding transactions (protocols) should therefore guarantee that an honest owner holds an evidence for having behaved legally. Further relationship exists between subsequent owners of the car. Also here, the trust relationship is considered to be partially.

As mentioned before these aspects have impact on the security of electronic immobilizers, however, they belong to infrastructural and organizational requirements and are not considered further in this contribution.

### 2.2 Assumptions

Our discussions are based on the following assumptions:

**Separation:** To keep the system model simple, we assume that the central locking system and the electronic immobilizer are implemented independently, without any interaction, i.e., the corresponding circuits are decoupled.[4]

**No Biometrics and no PIN Entry Devices:** The ownership of the ignition key is sufficient for the authentication. We do not consider biometric measures or PIN entry devices since (i) they require (costly) security devices and (ii) they do reduce user-friendliness, e.g., if the car owner wants to lend her car to friends.

**Organization:** Users are responsible for taking care of the ignition keys as well as the corresponding paper documents. The manufacturers provide a key management infrastructure. We further assume that the identifying data and the secret keys involved are already generated and programmed in the non-volatile memory of both the transponder and the motor control unit (e.g., in a secure production environment).

**Physical Access:** Towing a vehicle cannot be prevented by any electronic immobilizer!

## 3 Requirement Analysis

A variety of attacks can be mounted for an unauthorized initiation of the ignition process. Possible threats include cloning or simulating transponders[5] or exploiting any weaknesses in the implementation of security mechanisms or when updating the motor control unit and/or the transponder. Moreover, organizational threats against the transponders and the motor control units are of high importance: critical organizational functions concern, e.g., when users order the replacement of transponders or when new motor control units are to be installed (e.g., in workshops). Frauds can also occur during the development and key initialization.

We denote the set of all vehicles with $\mathcal{C}$ and the set of all transponders with $\mathcal{T}$. We call a transponder $T$ *valid*, if there is an approved mapping between $T \in \mathcal{T}$ and the corresponding vehicle $C \in \mathcal{C}$ where the approval is done by a trusted party (such as the manufacturer) or a trusted component certified by this party.[6]

[4] Nevertheless, there are obviously tendencies to integrate both systems ([2]).

[5] this means being able to construct an device with identical functionality (including the secret initialization data of the target device).

[6] One may desire procedures that do not require trust in manufacturers. However, in practice manufacturers may not be willing to accept this strategy.

A simple example is a list signed by the trusted party which contains the identification data (ID) of each transponder $ID_T$ and the ID of the vehicle $ID_C$ to which $T$ is assigned by the underlying mapping. We denote the set of valid transponders by $\mathcal{T}_{valid}$. Informally, the main requirements to be fulfilled by an immobilizer system are:

*Correctness*: A valid transponder $T \in \mathcal{T}_{valid}$ can always invoke the ignition process of the corresponding car.
*Security*: For a transponder $T^* \notin \mathcal{T}_{valid}$ it shall be infeasible to invoke the ignition process.

To be able to achieve the security objective mentioned above in practice, a variety of technical and organizational building blocks have to be deployed each having own requirements. In the following we will briefly consider these aspects.[7]

### 3.1 Security Requirements

In this section we consider the generic security requirements most of which are well-known.

**Protocol Requirements:** Typically, an authentication protocol has to be provided between the transponder and the motor control unit. Security aspects concern protection against active and passive attacks such as eavesdropping the communication between the transponder and the control unit for offline analysis, oracle attacks on the control unit, masquerading and replay attacks, and man in the middle attacks. A type of man in the middle attack is called *mafia fraud* [5] which is of particular concern in the context of wireless systems used for authentication and will be detailed in Section 4.1.

Note that to authenticate the motor control unit, a *mutual authentication* scheme between transponder and motor control unit is reasonable. However, to achieve this in existing vehicles we are faced with the following main problems: Firstly, it is feasible for a skilled adversary to connect a fake MCU to the CAN (Controller Area Network) bus in parallel to the original MCU with the goal to bypass the authentication mechanism later on. To make this hard, the link between MCU and the motor engine has to be separately secured. Secondly, there exists no *trusted path* between the human user and ignition key (e.g., an user interface) yet that can signal to the car owner the result of the authentication (or attestation) protocol.

**Evaluation:** There should be a possibility to verify the correctness of the applied protocols as specified by the immobilizer specification, e.g., by means of emulators checking the communication on the CAN bus. This would increase the trust of users in the underlying immobilizer systems.

[7] Note that the security requirements should remain fulfilled under different implementations, e.g., when software updates of the motor control unit are done by the manufacturer or if test functions are invoked.

**Implementation Requirements:** Further, technical requirements concern protection against attacks that exploit implementation weaknesses such as inherent leakage (e.g., side-channel attacks [13, 14]), forced leakage (e.g., fault analysis attacks [6]), and vulnerabilities of the logical or physical construction.

**Organizational Requirements:** These security requirements concern the life cycle issues, i.e., secure manufacturing, secure initialization (e.g., creation of individual data and cryptographic keys), secure distribution (e.g., transponder maintenance), and secure removal (e.g., destroying of cryptographic keys and components). An important aspect in this context is the requirement that car owners can prove that they are not cheating, e.g., by being able to prove the number of valid transponders even if the car is stolen. Moreover, it should be feasible to detect a complete replacement of the electronic immobilizer system to counteract a typical today's scenario of vehicle theft where a vehicle is first towed to a garage and subsequently the motor control unit is replaced by another one, that was earlier installed in a junk car.

### 3.2 Usability and Safety Requirements

Next, we list additional requirements of immobilizer systems starting from presumed functional requirements of the automotive industry, caused by safety and usability reasons:

**Time Constraints:** The execution time of the authentication must be short. This is obvious since the owner is not willing to wait for the engine to start.

**Resource Constraints:** The resources (e.g., hardware) are constrained. This is more critical for the transponder.

**Maintenance:** It should be possible to maintain the transponder on behalf of the owner. This includes the cases where the owner wants to block a transponder, e.g., in case it is lost, or add a new one.

**Functional Separation:** The security functions of the immobilizer should not have impact on safety aspects, e.g., a successful authentication of the transponder should be valid until the motor is turned off (a running motor should not halt for safety reasons).

**No Failure Counters:** Failed authentication attempts shall not lead to a denial of service.

## 4 Solutions, Open Issues and Limitations

Based on the requirements of Section 3, we now discuss important aspects to be considered when implementing immobilizer systems.

### 4.1 Authentication Protocol

Due to the functional requirements on the constrained devices (especially restrictions on the execution time) the use of symmetric cryptography is more efficient than protocols based on asymmetric cryptography.

As mentioned in Section 3.1 the physical link between the motor control unit and the motor engine has to be specially secured. The idea is that an adversary needs more efforts to detach the motor control unit. To make the separation hard for the adversary, a possible solution is to weld the MCU to the engine. However, it is also imaginable that the MCU consists of two parts, one part is hard wired with the engine and the other part is exchangeable.

For the mutual authentication a *trusted path* between the human user and ignition key (e.g., a user interface) that can signal to the car owner the result of the authentication (or attestation) protocol has to be established. Here, a small light-emitting diode on the ignition key might be a solution. Another solution might be the use of the user interface provided by the board computer, however, this implies the assumption that the display cannot be manipulated, which cannot always be guaranteed.

The ISO/IEC 9798-2 three-pass mutual authentication protocol ([3]) using random challenges is proposed as the basic authentication protocol (see also [7]).

Possible cryptographic algorithms for the encryption function include block ciphers, as Triple-DES and AES, and stream ciphers. The cryptographic algorithms Triple-DES and AES are available for direct use, both for encryption and for message authentication codes. A hardware implementation of AES on an RFID based chip is, e.g., presented in [9].

**Preventing Mafia Fraud Attacks:** Authentication schemes are used in many applications, but as already observed in [5] mafia fraud attacks cannot be prevented *only* by cryptographic mechanisms. The following scenario demonstrates the mafia fraud: consider a car which is parked next to the house of the car owner and the transponder is located inside the house, e.g., near the entrance. A thief gains mechanical access to the ignition lock and inserts a relaying device instead of the ignition key. The relaying device establishes a radio link which is directed towards the owner's house. Once, the transponder is activated by this radio link, the authentication protocol works as specified which leads to a start of the motor engine.

Here, the adversary does not own the transponder, but the adversary establishes a radio link to the transponder. The adversary makes use of the identity of the transponder, without awareness of the car owner.

**Preventing the Activation of the Transponder:** Mafia fraud attacks are caused by the RF-based activation of the transponder which does not require a human interaction. Therefore, mafia fraud attacks can be blocked if the transponder cannot be activated by the RF field anymore.

One possible solution is to include an ON/OFF switch on the ignition key which allows the car owner to set the transponder in a non-responsive mode. In the non-responsive mode, the transponder does not answer to any requests. Here, the car owner is responsible to care that the transponder is set to a non-responsive mode while not in use. Alternatively, the ignition lock could be used

for a mechanical unlocking of the transponder so that the car owner does not need to care about it.

**Distance Bounding Protocol:** An upper limit on the distance between two physical entities that are involved in a wireless protocol can be determined by precise timing measurements. Electromagnetic waves propagate with the speed of light $c$, which is approximately $c = 3 \cdot 10^8$ m/s. The spatial extension $\Delta r$ of an electromagnetic field after a time $\Delta t$ is given as $\Delta r = c \cdot \Delta t$. A location which is 3 m away from the origin is reached after 10 ns.

As the transponder and motor control unit exchange two messages, two electromagnetic waves propagate in opposite direction. In real life, additional delays have to be considered for the processing in semiconductor devices: at minimum one clock cycle is passed before the answer can be sent back.

In [8] the authors propose distance bounding protocols. The basic idea is as follows: A series of rapid bit exchanges takes place between the involved parties where the number of the bits depends on the security parameter specified. In the corresponding protocol the verifying party $V$ challenges the proving party $P$, who has access to secret keys, by sending random bits. $P$ has to reply immediately after receiving these bits. The delay time for replies enables $V$ to compute an upper bound on the distance to $P$. Some precautions should be taken to guarantee that the responses received by $V$ at the bit exchange originally stem from $P$, e.g., by a prior commitment by $P$. A modified distance bounding protocol should counteract mafia fraud attacks also in case that the random number generator of the transponder is weak.

The suitability of distance bounding strongly depends on high clock rates at the bit exchange sequence. Automotive immobilizers typically work in the frequency range of 100 kHz ([10]). Using clock frequencies of 100 kHz, it is not worth to implement the Distance Bounding protocol since with the time used for one clock cycle is 10 $\mu$s corresponding to a granularity of distance measurements of 3000 m. At frequencies of 13,56 MHz and above, the embedding of Distance Bounding becomes reasonable, at least for hardware based authentication protocols which can minimize the processing delay times.

### 4.2 Securing the Motor Control Unit

Here, our primary security aim is to prevent the disclosure and modification of secret initialization data of the motor control unit. Further, substitutions of motor control units should be detectable by control authorities later on.

**Physical Security:** We assume that the core of the motor control unit is a high-performance microcontroller which does not include hardware security mechanisms. In this case, it is recommended to embed the MCU into a tamper-responsive envelope. Note, that 'mal-function' of the MCU is a consequence of tampering if the module is encapsulated. An alternative, but still demanding approach, is the development of a secure 'tamper resistant' high-performance microcontroller.

Since tamper-responsive envelopes are costly, one may be satisfied by using 'only' a small tamper-resistant component that securely stores secrets as long as they are not used. For instance, the *Trusted Platform Module* (TPM) [11] suggested by the trusted computing group[8] (TCG) may be used. The TPM contains an unique certified key, called the endorsement key, that can be used to identify the TPM and thus the motor control unit. Using the TPM, one can also "bind" encrypted content to a specific TPM. This function is called *sealing*, allowing the realization of a secure update function of the control unit software. The *remote attestation* function provided by TPMs allows remote parties to verify the software configuration of the motor control unit using a cryptographically secure hash function. This allows the involved principals to verify the integrity of the installed software of the motor control unit.

Note that an add-on of a TPM requires a secure link between the TPM and each relevant control unit. Further, note that a complete exchange of the TPM and its associated components cannot be prevented either. Nevertheless, the use of a TPM causes higher efforts for the exchange of all associated components. A possible disadvantage might be the complexity that is induced by the TPM.

**Interface Security:** Attack scenarios as the manipulation of the software with the diagnostic interface are enabled if the design allows to bypass the authentication, either by exploiting flaws or by providing test interfaces which can jeopardize the security of the system. These kinds of attacks can be prevented by a careful system design. Software updates need to be verified by the motor control unit (or by the associated TPM) whether they were originated by the manufacturer before the software is modified. An access to test functions shall only be granted after a successful mutual authentication, e.g., with the valid transponder.

**Auditing:** As a complete exchange of the motor control unit (which is e.g., swapped out from a wreck of the same type) is hardly to prevent mechanisms should be in place which allow the control authorities to determine whether the MCU was originally fitted in the vehicle or not.

A possible solution is an authentication protocol between the control authority and the MCU that transfers one or multiple unique vehicle identification numbers. With this information either the originality is confirmed or the original place of installation can be revealed. However, note that in practice numbers such as the chassis number can still be manipulated.

### 4.3 Securing the Transponder

The primary aim is to prevent the disclosure and modification of secret initialization data of the transponder.

Transponders include an IC which is optimized for low power constraints. In [12] it was shown that transponders can include EEPROM memory and use it for the long-term storage of initialization data.

[8] www.trustedcomputinggroup.org

It is obvious that transponders should include security mechanisms to counteract both logical and physical attacks. The complexity of the logical functionality of transponders is quite small so that logical protection is definitively manageable, particularly, software updates are typically not foreseen. Regarding physical attacks, the transponders should be equipped with passive protection mechanisms to make tamper attempts sufficiently difficult.

### 4.4 Replacement of Transponders

The ownership of the ignition key shall authorize a principal to start the engine. However, when an ignition key is lost, the owner has to be provided with technical and/or organizational means to block the lost one and obtain and initialize a new one (with new cryptographic keys). There are several solutions imaginable, e.g., those which require a secure channel to the manufacturer or to accredited workshops, and those which do not.

**Maintaining Transponders by Infrastructure:** In case of an infrastructure maintained by the manufacturer the car owner is provided with a new ignition key if the car owner possesses the original paper documents. The initialization of the ignition key can be done by the manufacturer or at authorized workshops. In the latter case, we assume a secure cryptographic link between the transponder and the initialization center.

**Maintaining Transponders by Car Owners:** Today, it is very costly for car owners to loose a key because only certified workshops can do the replacement. The possibility for car owners to add new transponders and to remove old (e.g. lost) ones independent of the manufacturer would therefore increase both security and usability. In the following discussion, we are assuming that the non-volatile memory of the transponders can be rewritten and that a symmetric key scheme is used.

We propose a solution where the MCU is the central unit that initializes blank transponders (e.g., 'duckling principle' [15]) and that provides appropriate interfaces to authenticate the car owner. As discussed in Section 3, it is essential to ensure that the information how many valid keys currently exist is counted in a secure way, to ensure that owners cannot deceive insurances or buyers of their car. Thus, the MCU cannot be used to store this value, since this information would become unavailable in the case that a car is stolen. Instead, we propose to store the number of valid keys redundantly by all keys. Although this solution requires all transponders to participate this protocol, it has the benefit that the number of valid transponders can be controlled if at least one valid transponder is available.

To prevent that car owners can create secret copies of a transponder, confidentiality of the initialized transponder key has to be guaranteed. One solution is to transmit the cryptographic authentication key in an encrypted form. Therefore, blank transponders have to be shipped with an initial secret key that has to be known by the MCU, requiring some kind of key infrastructure.

Although the proposed solution is on the one hand more flexible and improves the privacy of car owners, it requires on the other hand more complex handling by the car owner. Moreover, the MCU has to provide an interface to perform the authentication of car owners.

But the most important issue is whether the automotive industry is willing to hand over this maintenance function to car owners, since if a manufacturer independent maintenance function is available, the manufacturer and the control authority cannot monitor the personal order of transponders anymore.

### 4.5 Further Implementation Issues

**Random Number Generation:** The random number generator should generate an unpredictable sequence of bits (even if the adversary has recorded the previous sequences). A common implementation choice is a pseudo-random generator that is based on a cryptographic cipher and uses two secrets: the key and an initialization value.

**Inherent and Forced Leakage:** The potential vulnerability of a cryptographic implementation towards inherent and forced leakage cannot be completely assessed by evaluating the design only. Practical tests should be conducted to examine the susceptibility of the implementation to passive and active side channel attacks. Appropriate defenses for the cryptographic implementation include the use of internal random numbers to de-correlate the inherent leakage of the cryptographic device from the secret data processed. Additionally, a desynchronisation in time is helpful. Fault Analysis can typically be averted by an internal verification of the result to avoid the output of faulty cryptograms. For further details we refer to the various contributions within the side channel related literature. Note, that an encapsulation as suggested in Section 4.2 makes leakage attacks more difficult, as the microcontroller cannot directly be accessed.

### 4.6 Movement and Positioning Systems

In Section 2.2 it was stated that towing of a vehicle cannot be prevented by an electronic immobilizer. Because of this, adversaries can tow the vehicle to a garage first before they replace components of the vehicle. There already exist sensors (e.g., Hall sensors) which measure the mechanical movement inside the gear of a vehicle. In combination with mobile communication systems (as GSM) alarm events can be signaled to the owner. Additionally, GPS can yield detailed information on the current location of the vehicle. Care should be taken that these sensors cannot easily be detected and disabled or removed before the vehicle is towed.

## 5 Conclusion

We started an open approach for designing electronic immobilizers. Herein, we presented and discussed a model, the security and functional require-

ments as well as solution ideas for constructing secure electronic immobilizers. We pointed out some of the main practical problems and limitations when deploying electronic immobilizers and made some suggestions for implementation. Mainly we considered the aspects of the motor control unit and the transponder which is integrated in the ignition key, but we also propose ideas for the key management by the car owner. A complete physical exchange of an electronic immobilizer system cannot be prevented. However, for the future detection of complete exchanges a cryptographic protocol for control purposes should be foreseen.

## References


1. http://www.secureyourmotor.gov.uk.
2. http://www.verkehrsunfallforensik.de/pdf/68_Wegfahrsperren.pdf.
3. *ISO/IEC 9798-2: Information Technology – Security Techniques – Entity Authentication – Part 2: Mechanisms using symmetric encipherment algorithms*. International Organisation for Standardization, 1999.
4. Die neue Strategie der Autodiebe. Frankfurter Allgemeine Zeitung, Nr. 40, Seite T1, 2004.
5. Thomas Beth and Yvo Desmedt. Identification Tokens — Or: Solving the Chess Grandmaster Problem. In A.J. Menezes and S.A. Vanstone, editors, *Advances in Cryptology – CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 169–176. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1991.
6. Eli Biham and Adi Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97*, volume 1294 of *LNCS*, pages 513–525. Springer-Verlag, 1997.
7. Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003.
8. Stefan Brands and David Chaum. Distance-Bounding Protocols. In T. Helleseth, editor, *Advances in Cryptology – EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 344–359. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1994.
9. Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong Authentication for RFID Systems Using the AES Algorithm. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *LNCS*, pages 357–370. Springer-Verlag, 2004.
10. Klaus Finkenzeller. *RFID-Handbook*. Wiley & Sons LTD, 2003.
11. Trusted Computing Group. TPM main specification. http://www.trustedcomputinggroup.org, Nov 2003. Version 1.2.
12. Ulrich Kaiser. Theft Protection by means of Embedded Encryption in RFID Transponders (Immobilizer). ESCAR conference, Cologne, Germany, November 2003.
13. John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Side Channel Cryptanalysis of Product Ciphers. *Journal of Computer Security*, 8(2/3):141–158, 2000.
14. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *LNCS*, pages 388–397. Springer-Verlag, 1999.

15. Frank Stajano and Ross Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Security Protocols—7th International Workshop*, volume 1796 of *Lecture Notes in Computer Science*, pages 172–194, Cambridge, United Kingdom, 2000. Springer-Verlag, Berlin Germany.
16. W. Thönnes and S. Kruse. Electronical driving authority - how safe is safe? -. VDI Berichte Nr. 1789, 2003.

# An Empirical Study on the Usability of Logout in a Single Sign-On System

Mikael Linden[1] and Inka Vilpola[2]

[1] CSC, the Finnish IT Center for Science,
P.O. Box 405, FI-02101 Espoo, Finland
Mikael.Linden@csc.fi
http://www.csc.fi/
[2] Tampere University of Technology, Institute of Software Systems,
P.O. Box 553, FI-33101 Tampere, Finland
Inka.Vilpola@tut.fi
http://www.cs.tut.fi/

**Abstract.** Single sign-on (SSO) has shown to be a successful paradigm in a network environment where a large number of passwords would otherwise be required. However, the SSO paradigm leaves the practices of logging out of services undetermined. In this study, the users' subjective satisfaction in the current implementation of login and logout was examined with both quantitative and qualitative methods. The study was carried out in a university using SSO in its intranet. The main result of this study is that when a multiservice environment uses SSO for user authentication, a single logout should also be used instead of expecting users to separately log out from each service.

## 1 Introduction

Authentication means the verification of a user identity in an information system. In a typical organisation, users have access to several independent services, each of them requiring separate credentials (for example, username and password) for user authentication. According to [1], users spend a considerable amount of time trying to recall the multiple username/password pairs. The helpdesk workload caused by forgotten passwords is also significant.

Single sign-on (SSO) means that users authenticate themselves only once and are logged into the services they subsequently use without further manual authentication [2]. As single sign-on increases efficiency in an organisation and reduces helpdesk calls, it has been a feature that IT managements in organisations have been calling for. Because SSO increases efficiency and user satisfaction, it also improves the usability of the system applying it [3]. Nowadays, a multitude of commercial and open source authentication products are available for taking care of single sign-on.

Authentication creates a security context which ensures (for example, by virtue of a symmetric encryption key) the authenticity of the message exchange between the user

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 243 – 254, 2005.

and the service during its life time. The service may provide the user an option to tear down this security context by providing an explicit logout service. Having pressed the logout button, the user again becomes unauthenticated for the service in question. In that sense, logout is an inverse operation for login.

In a single sign-on environment, logout becomes problematic. When a user clicks the logout button in a service, does she expect that she is logged out from this particular service or from all the services that she has been using during the single sign-on session? This study focused on services used in the web, as a web browser has become a de-facto interface for services provided by different organisations in the Internet. Both the intranet single sign-on scenario (in which the services are provided by a single organisation) and the federated identity scenario (in which single sign-on covers services provided by different organisations) were studied. The study also investigated how the users expect to be indicated whether they are anonymous or authenticated in a service in the SSO environment.

In the next section, previous research on single sign-on systems and usability studies of security systems in general are introduced. After that the context and methods used in our study are described and the results and analysis are presented. Finally the paper is concluded and guidelines presented for implementing logout in information systems that use SSO.

## 2 Previous Research

This chapter introduces the taxonomy of SSO systems and the concept of federated identity. It also presents some notes on the dilemma of security and usability in SSO.

### 2.1 Single Sign-On Systems

Pashadilis et al [2] and De Clercq [4] have presented architectures and systems for single sign-on not only in the web but in information systems in general. In his taxonomy, Pashadilis presents four categories for single sign-on systems. These categories are summarised in Table 1. The corresponding categories by De Clercq are presented in parentheses.

**Table 1.** The four categories for single sign-on systems by Pashadilis et al [2]. De Clercq's corresponding categories are in parentheses [4]

| | Local SSO systems | Proxy-based SSO systems |
|---|---|---|
| Pseudo-SSO systems | Local pseudo-SSO systems (secure client-side credential caching) | Proxy-based pseudo-SSO systems (secure server-side credential caching) |
| True SSO systems | Local true SSO systems (public key infrastructure based SSO systems) | Proxy-based true SSO systems (token-based SSO systems) |

In pseudo-SSO systems, no modifications are necessary for the actual service; instead, there is a pseudo-SSO component between the user and the service. The component authenticates the user and provides her cached credentials to the service. In a true SSO system, there is no cache for user credentials. Instead, the services are modified to trust assertions made by a specific Authentication Service Provider that takes care of the actual authentication of the user.

In a local SSO system, single sign-on is implemented by a component installed in the user's workstation, whereas in the proxy-based mode, there is a specific server dedicated for authentication between the user and the service. In addition to the categories presented in Table 1, De Clercq presents a fifth category for credential synchronisation systems, in which user credentials are synchronised between the services. As this architecture does not actually provide the user single sign-on but just the same username and password for several services, it is not included in Table 1.

Several case studies describe single sign-on deployments. Volchkov [5] has studied the single sign-on deployment of a Swiss bank. Anchan [6] has implemented the principle of one username and password in a school in Florida. Taylor [7] has presented an architecture for sharing biological specimen and health indicator databases in Australia. However, logout has seldom been discussed in the articles. Also, neither Pashadilis nor De Clercq has presented how logging out should be handled in their architectures.

### 2.2 Federated Identity

As a user typically uses services provided by different organisations, it has become necessary to build models for using a single set of credentials and single sign-on also in services across organisational boundaries. A federation is “an association of organisations that come together to exchange information as appropriate about their users and resources in order to enable collaborations and transactions” [8]. In a federation, a user has an Identity Provider that acts as her home organisation and is responsible for maintaining her identity and authenticating her. The Identity Provider uses federating software to pass the user's identity or attributes to the Service Provider, which provides the actual service the user is accessing. Several federating softwares have been developed mostly for the web, such as Liberty [9], Shibboleth [10] and WS-Federation [11]. Security Assertion Mark-up Language (SAML), which is a widely used building block of federating softwares, has covered the logout problem by specifying a dedicated message exchange for single logout in the federation.

Microsoft has also developed its proprietary Passport system for cross-organisation access. Microsoft Passport has a single logout feature; when the user presses logout in one of the services, the user is logged out of all the services she has been using [12, p. 14]. Oppliger has studied the newest version of the Passport, .NET Passport, and describes its design in his article [13]. Kormann [14] has pointed out a usability problem in one of the services utilising the Microsoft Passport. In the service, there were two logout buttons, one for the specific service and the other for the entire Microsoft Passport federation. Kormann concluded that it may be too difficult for an end user to understand the difference between the two logout buttons.

### 2.3 Usability Versus Security

Usability has been defined as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [3]. Security and usability can be seen as counterparts as the passwords load users' memories and the authentication dialogues interrupt their primary tasks. To reduce the users' memory loads companies have applied the single sign-on technique [1].

Anchan et al. [6] list a few advantages of the SSO paradigm. They also state the memorability of a single password but add that need for support or help desk calls related to passwords are reduced by using SSO. Applying SSO actually improves security as the need to write down the passwords is reduced. At the same time, user experience is also improved. Productivity increases when the time needed for authentications is reduced.

User experience can also be affected by providing the user feedback on the system status. Jakob Nielsen recommends that the visibility of the system status, for example, the user being authenticated, should be one of the user interface design principles [15]. Changes in the system status should be informed to the user in an appropriate way.

Security related tasks, such as logging out of services, are not necessarily the primary focus from the end-users' point of view, remind Smetters et al [16]. Therefore, the usability of logging in and out of a system is hard to evaluate with traditional usability testing related to the users' natural tasks within the system. Smetters et al. suggest that recording, observing and interviewing should be the methods for the usability testing of security applications. The interviews should, in particular, focus on ensuring the positive user experience of secure application usage and in the situations when the end-user is forced to make choices in terms of security.

## 3 Research Context and Methods

The object of this research was to find out the users' subjective opinion (that is, the user experience) on using the current SSO system in a university's intranet. Special attention was paid to the logout in the SSO system and moving between services in the intranet and outside of the intranet. The need for the indication of the user being authenticated and the preferred feedback to present it were also studied. Two major groups were first defined to be the target user groups of the research: undergraduate students in the university and the university's personnel.

Two research methods were used to get both qualitative and quantitative data. The qualitative data was collected with two focus groups and, based on the results, the quantitative data was gathered by tests using a structured inquiry.

### 3.1 Background and Research Context

The university's intranet is formed of multiple web-based services provided by the university's administration and institutes. To authenticate the users, the intranet uses a

Pubcookie single sign-on system (version 3.0), an open source product developed by the University of Washington [17]. The user interface indicates the user being authenticated depending on the service. For example, in the exam enrolment service the user's name and student number are shown. Not all the users are able to use all the services in the intranet; instead, the services have several different user groups with different privileges. Altogether there are 11 500 active students, staff and faculty members who have a username and password for the intranet.

As the users log in to the university's intranet (and, thus, to the single sign-on system), they are able to use the various services without further authentication. But when a user logs out of any of the services, she still remains logged in to the other services she has been using during the single sign-on session. In other words, even after having logged out, the user can still browse (for example, by following a link, typing the URL to the browser or pressing the back button in the browser) the other services in which she has not pressed the logout button. This can be harmful unless the users are fully aware of this feature, because if a user leaves the browser open, the next user has access to the other services with the previous user's identity. In the university's workstations, security has been ensured by forcing the user to log out of the workstation after use. In public places, like libraries or conferences, the risk can be realised.

### 3.2 Focus Group Sessions

The objective of the focus group sessions was to reveal potential usability issues related to the SSO system in the university's intranet. The focus group method was chosen in order to have an open discussion between the end users and get authentic opinions and experiences of system usage. The purpose was to design the questions of the quantitative inquiry according to the findings in the focus groups.

Focus group is a qualitative method where 6 to 9 users from a targeted user group have a moderated discussion [18]. The moderator has a manuscript on how the discussion is kept on track. The moderator also takes care that no single participant dominates the discussion nor surrenders from the discussion. In focus groups, some kind of stimulus material is usually presented to initiate the discussion.

Two focus group sessions were arranged, one for both user groups. The manuscript was the same for both groups (Table 2), but the context presented in the stimulus material differed slightly depending on the user group.

The stimulus material in both focus groups consisted of four pictures (Figure 1). The pictures were shown as a slideshow one at a time. The scenario was about two researchers visiting a conference in a European city. One of the researchers wants to check his email auto-reply settings in the home university's intranet using a conference PC. The other researcher uses the same service after the first researcher has logged out from some other service within the intranet. To her big surprise, she finds that the first user is still logged in to the auto-reply service. The story was much alike for undergraduate students, but the intranet service was different.

The focus group sessions were held in a meeting room where all the participants and the authors could sit around a table. Materials were projected so that the partici-

pants could clearly see them. Both undergraduate students and personnel were invited through a news group, and a total of 8 undergraduate students and 8 employees participated in the groups. All participants were given a movie ticket as a reward for their participation. On the basis of the researchers' notes, the main conclusions of each subject of discussion were drawn.

**Table 2.** Contents of the manuscript used in the focus group sessions

| Phase | Material and questions |
|---|---|
| Welcoming | Opening words and advice to shut down mobile phones<br>Introduction to the focus group method<br>Permission to record the discussion asked in writing |
| Start-up questions | A list of the intranet services was delivered on paper to the participants<br>The participants were asked to circle with a pen the services they intuitively connect with the university's intranet<br>The same list of services was projected on the wall and the discussion was led to the user identification required in order to get to the services<br>1. Special situations that had occurred while logging in<br>2. Frequency of password request<br>3. Rationale of password request<br>The place where the services were most often used; the university, the home, a public place |
| Stimulus | Stimulus material presentation<br>Discussion related to the presented scenario |
| Usage scenarios | The same list of services as during the start-up questions was again projected on the wall<br>Every question followed the same pattern: first the participants were asked what they expected to happen, then an expert explained what really happens and finally the participants were asked again how they think it should work. The questions were about:<br>1. Logging in to the intranet,<br>2. Moving from one service to another within the intranet,<br>3. Moving away from the university's intranet,<br>4. Logout from intranet services<br>5. Using services outside the university with the same SSO<br>Finally, there was some discussion about what can be defined as intranet |
| Thank you | Possible questions<br>Rewarding participants |

**Fig. 1.** Pictures forming the visual part of the scenario

### 3.3 Quantitative Test Session with Structured Inquiry

The objective of the quantitative test session and structural inquiry was to confirm some of the results of the focus group sessions. The test section included stimulation of the same type which was presented in the focus group scenario. After the stimulative exercises, students answered the questionnaire. The target user group was now only the students at the university as they were seen as the critical mass of those using the university's intranet services. The structured inquiry consisted of two background questions, 11 statements to agree or disagree with by a five point Likert scale and one multiple choice question.

The quantitative test sessions were held in the university's computer classroom, where workstations are available for all students. The invitation was published on the university's bulleting boards and in the intranet. All participants were given coffee and a donut in the university's cafeteria as a reward. The results were taken to a spreadsheet to conduct a statistical analysis.

# 4 Results and Analysis

In this chapter, the results of the quantitative study are presented and analysed and the main conclusions of the study are drawn.

## 4.1 Results of the Quantitative Tests

The results of the quantitative research are summarised in Table 3 and Chart 1. In total, 58 answers could be accepted in the research. 79 percent of the answers were

**Table 3.** Results of the quantitative research

| Questions grouped by theme | Mean | Stdev |
|---|---|---|
| **SSO concept** | | |
| 1. When I log in, I should be informed about all the services that are covered by the single sign-on system. | 3,76 | 1,0 |
| 3. The services that utilise single sign-on need not be linked to each other. Instead, they may be completely separated from each other. | 3,12 | 1,14 |
| 4. Instead of utilising single sign-on, the service should ask the password again when I enter another personal service. | 3,03 | 1,24 |
| 5. All the services in a single sign-on system should have a similar layout. | 2,48 | 1,16 |
| **Logout** | | |
| 2. When I use several services having single sign-on, logging out of one of the services should log me out of all of the services. | 4,16 | 1,15 |
| **Indication about being authenticated** | | |
| 7. My name or username should always be present on the web pages in a single sign-on system. | 3,86 | 0,93 |
| 9. I want to be aware of whether I am anonymous or an identified user in a service. | 4,5 | 0,63 |
| **Services provided by other organisations** | | |
| 6. In my opinion, services in a single sign-on system should not be limited to services provided by one organisation. | 3,61 | 0,93 |
| 8. If I browse to a service provided by another organisation, such as to the study grants service provided by the national Social Insurance Institution, I should re-enter my password. | 2,91 | 1,19 |
| 10. If the single sign-on system covers several organisations, I should be notified when I am moving to a service provided by another organisation. | 3,6 | 0,97 |
| 11. It does not matter what organisation actually provides the service in a single sign-on system. | 2,97 | 1,15 |

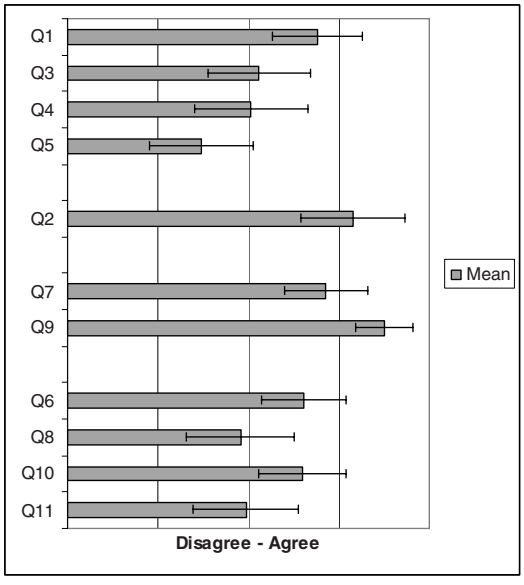



**Chart 1.** Table 3 being visualised. The scale for mean is the Likert scale (1-5), where 3 means "I cannot say"

given by male and 21 percent by female students. 24 percent of the participants were first year students, 36 percent second year students, 16 percent third year students, 12 percent fourth year students and 12 percent had studied more than four years. The participants represent the population of the university well. More research will be needed for generalising the results for other types of organisations.

### 4.2 Single Sign-On Should Imply Single Logout

Question number 2: "When I use several services having single sign-on, logging out of one of the services should log me out of all of the services." was strongly agreed with by the users (mean 4,16, stdev 1,15).

The undergraduate students in their focus group session had no consensus over whether there should be single sign-on or separate login for each service. However, they were united that if there is single sign-on there should also be single logout, or, alternatively, if there are separate logins there should also be separate logouts from every service. One of the students even stated that the current paradigm has a bug, because despite of the single sign-on, the system leaves the user logged in to the other services when the user logs out of one of the services.

In the personnel's focus group session, the participants argumented why the current logout paradigm, single sign-on without single logout, can be misleading. The university's intranet is formed of multiple services and thus there is no clear beginning or end where to logout. The university's personnel also claimed that they have not seen clear instructions about logout even when they received their username and password for the university's intranet. The personnel presented a pragmatic solution to remind

the user about logging out of the system; if the single logout is not implemented, each service should be opened in its own browser window. This would remind the user of the logout of every service that has been opened.

### 4.3 Users Need to Be Informed If They Are Not Anonymous in a Service

The quantitative test showed that users want to be aware of whether they are anonymous or authenticated users in a service (question 9., mean 4,5, stdev 0,63). They also required that their names or usernames should always be present on a web page in a single sign-on system (question 7., mean 3,86, stdev 0,93). The results clearly indicate that the users, while browsing between services inside and outside the single sign-on system, feel uncomfortable if they do not know if a particular service knows who they are.

The undergraduate students thought that there is no need for an indication while moving from personal to public sites. Instead, they were concerned for the inconsistent behaviour of the browser's Back function; users are able to re-enter the personal web page by pressing the browser's Back button even once they have logged out of the service. The same phenomenon was mentioned in the personnel's focus group.

### 4.4 Users Do Not Resist the Idea of Having Services Provided by Other Organisations in the Single Sign-On System

The quantitative test results did not indicate a strong agreement or disagreement on whether the university's single sign-on system could also cover services outside the university (questions 6., 8., 10. and 11.), i.e, services utilising federated identity. Test users slightly agreed on that "Services in a single sign-on system shall not be limited to the services provided by one organisation." (question 6., mean 3,61, stdev 0,93).

The undergraduate students' focus group stated that if the service outside the university is very sensitive, like health care services, the password should be asked again. Again, the logout of the service outside the organisation should log the user out of all of the services provided by the organisation. The university personnel's focus group had different opinions about the need for a new login to the services provided by another organisation. One of them suggested that services from other organisations should open in a new browser window.

It seems that services provided by an external organisation could be included in the organisation's single sign-on system. The need for re-entering the password might depend on the service, but at least it should not always be required. If the external service is part of the university's single sign-on system, the same logout paradigm should be applied.

## 5 Guidelines and Conclusions

The single sign-on paradigm does not dictate the implementation of logout in a single sign-on system. This might lead to insecure situations, in which a user has logged out of one of the services in the single sign-on system but remains logged in to others without being aware of it. Guidelines based on user studies are needed in order to achieve usable and secure login/logout systems.

Based on our research, we present guidelines on how to implement logout in a multiservice environment using SSO:

1. *The logout paradigm should follow the login.* If single sign-on is applied then single logout from all of the services should also be applied. Mixed solutions can be confusing for the users and thus cause insecurity.
2. *Users should be fully aware of whether they are anonymous or authenticated users in a service.* This should be applied as long as the user is logged in to the services.
3. *The single sign-on system may cover services provided by different organisations.* Independently of the service provider, the same login and logout manners should be applied.

The guidelines should be applied both to organisational single sign-on systems and to services utilising federated identity.

## Acknowledgements

We would like to thank research assistant Janne Tervonen, who arranged both the qualitative and the quantitative test sessions.

## References

1. Sasse, M.A., Brostoff, S., Weirich, D.: Transforming the 'weakest link' – a human/computer interaction approach to usable and effective security. BT Technol J Vol 19 No 3 (2001)
2. Pashalidis, A., Mitchell, C.: A Taxonomy of Single Sign-On Systems. In: Safavi-Naini, R., Sebarry, J. (eds): Proceedings of the 8th Australasian Conference on Information Security and Privacy. Lecture Notes in Computer Science, Vol. 2727. Springer-Verlag, Berlin Heidelberg New York (2003) 249–264
3. ISO/IEC (1998b) 9241-11 Ergonomic requirements for office work with visual display terminals (VDT)s - Part 11 Guidance on usability, ISO/IEC 9241-11: 1998
4. De Clercq, J.: Single sign-on architectures. In: Davida, G., Frankel Y., Rees, O. (eds): Proceedings of Infrastructure security, International Conference, InfraSec 2002. Lecture Notes in Computer Science, Vol. 2437. Springer-Verlag, Berlin Heidelberg New York (2002) 40–58
5. Volchkov, A.: Revisiting Single Sign-on. A Pragmatic Approach in a New Context. IEEE IT Professional, Vol. 3 No 1 (2001) 39–45
6. Anchan, D., Pegah, M.: Regaining Single Sign-On Taming the Beast. In: Proceedings of SIGUCCS'03 Conference. ACM Press, New York (2003) 166–171
7. Taylor, K., Murty, M.: Implementing Role Based Access Control for Federated Information Systems on the Web. In: Johnson, C., Montague, P., Steketee, C. (eds): Australasian Information Security Workshop 2003. Australian Computer Society Inc, Sydney (2003) 87–95
8. The InCommon Federation. The InCommon Glossary. Available in http://www.incommonfederation.org/glossary.cfm (visited 1/2005)
9. Liberty Alliance Project. Liberty ID-FF Protocols and Schema Specification version 1.2. Piscataway, New Jersey (2003)
10. Internet2/MACE. The Shibboleth project. http://shibboleth.internet2.edu/ (visited 1/2005)

11. Web Services Federation Language. IBM, Microsoft, VeriSign (2003)
12. Microsoft .NET passport review guide. Microsoft corporation (2004)
13. Oppliger, R: Microsoft .NET Passport and Identity Management. Information Security Technical Report, Vol. 9 No 1 (2004) 26–34
14. Kormann, D., Rubin, A.: Risks of the Passport single signon protocol. Computer Networks, Vol. 33 Issues 1-6 (2000) 51–58
15. Nielsen, J.: Ten Usability Heuristics. Available in http://www.useit.com/papers /heuristic/heuristic_list.html (visited 1/2005)
16. Smetters, D.K., Grinter, R.E.: Moving from the Design of Usable Security Technologies to the Design of Useful Secure Applications. In: New Security Paradigms Workshop '02. ACM Press, New York (2002) 82–89
17. University of Washington. Pubcookie: open-source software for intra-institutional web authentication. http://www.pubcookie.org/ (visited 1/2005)
18. Nielsen, J.: Usability Engineering. Academic Press, San Diego (1993) 214–216

# Secure Software Delivery and Installation in Embedded Systems*

André Adelsbach, Ulrich Huber, and Ahmad-Reza Sadeghi

Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
Andre.Adelsbach@nds.rub.de
{sadeghi, huber}@crypto.rub.de

**Abstract.** Increasingly, software (SW) in embedded systems can be updated due to the rising share of flashable electronic control units (ECUs). However, current SW installation procedures are insecure: an adversary can install SW in a given ECU without any sender authentication or compatibility assessment. In addition, SW is installed on an all-or-nothing base: with the installation, the user acquires full access rights to any functionality. Concepts for solving individual deficiencies of current procedures have been proposed, but no unified solution has been published so far.

In this paper we propose a method for secure SW delivery and installation in embedded systems. The automotive industry serves as a case example leading to complex trust relations and illustrates typically involved parties and their demands. Our solution combines several cryptographic techniques. For example, public key broadcast encryption enables secure SW distribution from any provider to all relevant embedded systems. Trusted computing allows to bind the distributed SW to a trustworthy configuration of the embedded system, which then fulfills a variety of security requirements. Finally, we outline the management of flexible access rights to individual functionalities of the installed SW, thus enabling new business models.

## 1 Introduction

Control unit hardware (HW) and SW in embedded systems used to be tied together as one single product and rarely changed once the system had been shipped. Nowadays, HW and SW in an electronic control unit (ECU) have become separate products. SW can be updated or upgraded after shipment and add customer value due to the ubiquitous use of flashable ECUs. Examples are the ECUs in a modern car where updates can increase the engine performance and reduce emission levels.

Current procedures for installing SW in an embedded ECU are insecure—details about the deficiencies will be given in Sect. 2. Historically, these deficiencies didn't matter because SW installation was focused on warranty-based

* A full version of this paper containing further details is available at [1].

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 255–267, 2005.

replacement of defective SW. The system owner was informed in costly recalls and received the SW updates free of charge. Recently, a paradigm shift has taken place: value-added SW components can be distributed to interested owners and new business models allow the extraction of revenues even after shipment.

The secure delivery of SW to embedded systems and the management of the corresponding digital rights differ from any existing DRM system known to the authors. First, the distribution currently necessitates a skilled intermediary between SW provider and user because the installation process relies on system-specific equipment which is only available to maintenance personnel. For example, a SW update in a vehicle ECU is usually installed via a manufacturer-specific diagnostic tester that is only available to maintenance providers.[1] Second, different classes of such intermediaries exist: depending on their equipment and capabilities, maintenance providers usually have different installation rights. In the automotive example, an uncertified garage might not be granted the right to install SW for safety-relevant ECUs such as the airbag ECU. Third, a newly developed SW component is not necessarily compatible with any target ECU and the SW of all other ECUs in the embedded system. For example, an average compact-class vehicle contains 40 ECUs while high-end and luxury class vehicles can have up to 70 ECUs.[2] Secure SW installation must therefore fulfill a variety of requirements regarding security and usability. Last, new business models for embedded systems will induce new requirements. Due to the high value of the embedded system and the potential consequences of system failure, non-repudiation will be an important requirement.

We propose a procedure for secure SW delivery and installation in embedded systems. We combine a variety of different cryptographic techniques to build such a procedure. The main contribution of our proposal is the secure installation procedure itself based on Public Key Broadcast Encryption (PKBE) and Trusted Computing. Another contribution is a requirement model for all parties that participate in a typical distribution and installation setting. To the authors' knowledge, neither a suitable procedure nor a general requirement model have been previously published although individual requirements have been proposed [9, 2, 10]. The use of the PKBE scheme proposed in [11] has several advantages in this particular setting.[3] First, it enables *efficient* one-way communication from SW providers to a potentially large, but selected set of embedded systems, even though they have to be considered *stateless* receivers containing a fixed set of secret keys which can't be updated. Specifically, the length

[1] Maintenance providers such as dealers, garages and road service teams carry out the SW installation as the car owner lacks the necessary equipment and skills [2].

[2] The Volkswagen Phaeton has 61 ECUs [3]. In addition, each OEM usually has different car models with differing ECU configurations. The ECU configuration of a particular model changes during the production life cycle due to an update of HW or SW components [3, 4]. The compatibility of a SW component does not only depend on the target ECU hardware, but also on other ECUs in the vehicle [5, 6, 7].

[3] Broadcast encryption was first introduced in [12]. Several improvements were proposed, e.g., in [13]. We refer to the public key broadcast encryption scheme of [11].

of the message header does not grow with the number of intended receivers as in the case of a standard Public Key Infrastructure (PKI).[4] Second, the proposed PKBE scheme allows the revocation of an unbounded number of receivers. Even if a large number of receivers has been compromised or is to be excluded, messages can still be broadcast to the remaining receivers. Last, it gives non-discriminatory access to the broadcast channel. The public key property allows any (not necessarily trusted) party to broadcast to any chosen set of receivers. Specifically, the manufacturer of the embedded system can't exclude any SW provider from the broadcast channel or otherwise prevent competition.[5]

Trusted Computing is the enabling technology for an embedded system to become a trusted receiver of broadcast messages. Based on minimum additional hardware and cryptographic techniques such as attestation and sealing, an embedded system can be trusted to be in a particular configuration. The assessment of the compatibility of a particular SW component with the embedded system can be based on this configuration. In order to avoid discrimination of certain SW providers, we suggest the use of property-based attestation [14].

## 2 Related Work

Several types of embedded systems exist and specific literature on each type is available. However, we consider a modern vehicle to be the most challenging example, namely due to the specific qualities of SW distribution and installation as outlined in Sect. 1. In particular, the high number of ECUs and their variants leads to a complex assessment of compatibility. Therefore, we focus on automotive literature and add an example from the field of IT security. A typical procedure for installing SW in an automotive ECU is described in [15]. It is performed by a so called flashloader, a standard SW environment that allows for in system re-programming of ECUs. Current installation procedures rarely apply any cryptographic techniques [15, 16, 6].

A framework for international automotive SW installation standards is introduced in [16]. However, it doesn't consider any DRM or security aspects. A proposal for "end-to-end security" of SW installation in vehicles is made in [17]. However, the signing of the SW component by "an authorized party" is the only protective measure, which provides only a partial solution[6] to the requirements that we will introduce in Sect. 4. An extended discussion of related work is presented in the full version of this paper [1].

[4] If standard PKI was used, the message header length would be $O(|\mathcal{U}|)$ where $\mathcal{U}$ is the set of intended receivers. In the PKBE scheme from [11], this length is only $O(r)$ where $r$ is the number of revoked or excluded receivers.

[5] Non-discrimination is also important on the receiving end: for instance, the European Commission Regulation 1400/2002 prevents discrimination of independent maintenance providers. The manufacturer must give them access to necessary material and information, e.g., spare parts, technical information and diagnostic equipment.

[6] For example, it does not prevent discrimination of independent SW providers as the vehicle manufacturer is assumed to take over the role of the authorized party.

# 3 Model

## 3.1 Roles and Objects

The following roles (cf. Fig. 1) will be used throughout this paper:

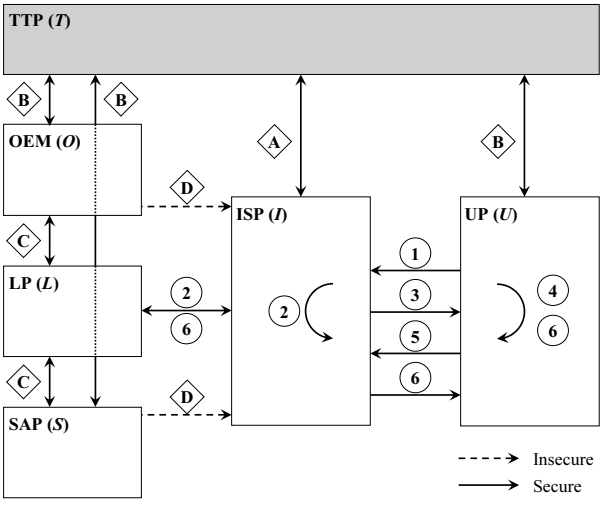



**Fig. 1.** Installation procedure in six steps

($O$) **OEM:** The Overall Equipment Manufacturer (OEM) develops, assembles and delivers the embedded system to users. For this, $O$ cooperates with suppliers that develop and/or manufacture components for the embedded system. The initial SW components at shipment time may be either from $O$ or from his suppliers. Automotive examples are car manufacturers such as Daimler Chrysler, Ford, GM or Toyota.

($S$) **SAP:** SW Application Programmers (SAPs) develop SW components for the embedded system. They may either be (i) suppliers that participate in developing and/or assembling the embedded system or (ii) independent application programmers that develop SW components (updates and/or upgrades) and distribute them after shipment. Automotive examples are suppliers such as Bosch, Delphi, Denso, Siemens and Visteon.
We use the term *"SW provider"* as a synonym for "OEM or any SAP".

($I$) **ISP:** The Installation Service Providers (ISP) maintain the embedded system, i.e., mechanical parts, ECU HW and SW. As part of their maintenance services, they install updates and/or upgrades of SW components. They have equipment that is necessary for the installation procedure and capabilities that allow them to correctly install SW components. Automotive examples are car dealers, garages and road service teams. The installation rights of $I$ are modeled as clearance levels. Each SW component requires a minimum

clearance level $\text{Clear}_{\min}$. $I$ can have any clearance level in $\{1, 2, \ldots, m\}$. If $I$ has clearance level $i$, it may install any SW with $\text{Clear}_{\min} \leq i$. The highest level $m$ permits the installation of any SW. Without clearance level, no SW may be installed.[7]

($L$) **LP:** The License Provider (LP) distributes licenses for SW components that the SW providers $O$ and $S$ have developed. Prior to distribution of a license, $L$ needs to establish terms and conditions with the SW providers in which the model for sharing license revenues is detailed.[8] To the authors' knowledge, automotive examples don't exist yet, but might be established as spin-offs of OEMs and SAPs.

($U$) **UP:** The User Platform (UP) is manufactured by $O$ and purchased by the user. The user is interested in SW for $U$ and willing to pay for it if it offers a perceivable value-added. We define $U$'s configuration as the collective information on each SW (and implicitly HW) component that is installed in $U$. The obvious automotive example for $U$ is a car. We assume $U$ to have an internal communication network over which all its components denoted by $\{u_0, u_1, \ldots, u_n\}$ are connected. In the implementation of an embedded system, they correspond to ECUs. $u_0$ is assumed to be the trusted computing base and provides a central installation and license service. $u_0$ is the only component capable of distributing new SW to the other components $u_i$, $1 \leq i \leq n$. Due to cost constraints, we cannot assume the $u_i$ to be high-performance components, i.e., their computational resources are limited, especially related to cryptographic techniques. The SW distribution from $u_0$ to the $u_i$ is performed over an internal communication network to which all components are connected.[9]

($T$) **TTP:** The Trusted Third Party (TTP) has two different certification tasks: first, $T$ creates SW certificates for $O$ and $S$. These certificates confirm the properties of each newly developed SW component. With SW properties we mean characteristic features of SW such as functionality, interfaces, supported protocols, memory and processor requirements, necessary environment, etc. Second, $T$ creates clearance level certificates which certify $I$'s right to install specific SW components. In the automotive example, this role is currently taken over by $O$. This implies a trust model in which each $S$ must trust $O$. However, an independent $T$ becomes necessary if $O$ is not fully trusted and discrimination of any $S$ should be avoided. An independent $T$ might evolve out of safety standards authorities such as the NHTSA[10] in the USA or the Euro NCAP[11] in Europe.

---

[7] Other models for installation rights can easily be integrated into our proposal. For the purpose of this paper, clearance levels serve as an example.

[8] The discussion of licensing models, e.g., pay-per-installation or pay-per-usage, is out of the scope of this paper.

[9] In the automotive example, this holds for communication networks ("data busses") such as CAN, LIN and MOST.

[10] National Highway Traffic Safety Administration, `http://www.nhtsa.dot.gov/`

[11] `http://www.euroncap.com/`

## 4 Security Requirements

We consider the security requirements of each party separately. The following terms will be used in this section: when the installation results in *success*, we mean the execution of a complete installation. A *complete installation* includes the installation of a legal SW component and the delivery of a legal license. A *legal SW component* is a SW component whose properties have been certified by $T$ and committed by the SW provider. A *legal license* is a license which was legally generated by $L$ and legally acquired by $U$. With *failure* we mean that no SW is installed, i.e., $U$'s configuration does not change. A *legal $I$ for a specific SW* is defined to be an $I$ with an authentic clearance level certificate from $T$ with a clearance level sufficient for the SW. A *legal $U$* does neither request illegal nor incompatible SW nor involve an illegal $I$.

### 4.1 OEM Requirements

**(OCR) Correctness:** The result of the installation procedure must be success if and only if all involved parties behave according to the specified protocol.

**(OPE) Policy Enforcement:** $O$ requires enforcement of following policies:

- **(OPE1) Rights Enforcement:** After acceptance by $L$, the terms and conditions of $O$ should not be circumvented.
- **(OPE2) Compatibility Enforcement:** An installation will result in *success* only if the SW and $U$ are compatible. *Compatibility* of a SW and $U$ means that the SW properties are conform to and suitable for $U$'s configuration. For example, this implies that the SW must run correctly on $U$ and may not have inconsistent interfaces.
- **(OPE3) ISP Clearance Enforcement:** Only a legal $I$ may install SW.[12]

**(OCF) Confidentiality:** No party except $O$ and the trusted component $u_0$ of $U$ may be capable of reading SW developed by $O$ *prior to* installation.[13] This is meant to protect the intellectual property contained in $O$'s SW. However, we only consider conditional access to the SW.[14]

**(OI) Integrity:** The installed SW component must be integer.

### 4.2 SAP Requirements

$S$ shares all requirements with $O$, but has an important additional requirement:

**(SND) Non-discrimination:** The identity of $S$ may neither influence $S$'s ability to send over the broadcast channel nor the result of the installation procedure.

[12] For example, this protects $O$ from warranty claims of the user when the user pretends that $O$ and $I$ have colluded to install SW with an illegal clearance level certificate.

[13] This also excludes $I$ from reading the cleartext. However, $I$ will still be necessary in most installation procedures because $I$ has the necessary skill set, installation equipment, maintenance area, spare parts, etc.

[14] Complementary measures, e.g., fingerprinting, are out of the scope of this paper.

For example, when $S_1$ and $S_2$ have each developed a legal SW with the same properties, $S_1$ may not be *technically* preferred in the installation procedure.[15]

### 4.3 ISP Requirements

**(ICR) Correctness:** This requirement is identical to the requirement OCR.

**(INR) Non-repudiation:** After each installation procedure, $I$ must be able to prove the origin and the result of the installation to any honest party.

**(ICE) Clearance Enforcement:** This requirement is identical to OPE3. For example, this justifies $I$'s effort to obtain a clearance level certificate.

**(IND) Non-discrimination:** A legal $I$ must be able to install *any* SW component which $U$ requests and which is at or below his clearance level. For example, the SW provider may not be able to separate ISPs with identical clearance level into subgroups and exclude individual subgroups from the SW distribution.

**(IFP) Frame-Proofness:** If no installation has occurred, $I$ may not be wrongly accused of treachery, e.g., of having installed SW.

### 4.4 License Provider Requirements

**(LNR) Non-repudiation:** A licensee cannot deny the receipt of a legal license.[16]

### 4.5 User Requirements

**(UCR) Correctness:** This requirement is identical to the requirement OCR.

**(UNR) Non-repudiation:** After the installation procedure, $U$ must be able to prove the result, i.e., either success or failure, to any honest party.

**(UIO) Installation Origin:** No SW installation may be performed without request by $U$.

**(UA) Authenticity:** The installed SW component and the license must be authentic, i.e., as requested by $U$ and sent by the SW provider and $L$ respectively.

## 5 Proposed Solution

This section provides a summary of the proposed installation procedure (cf. Fig. 1) that consists of a setup period (Phases A–D) and the actual installation (Steps 1–6). The protocols for these two parts will be detailed in Sect. 5.2.

We first give an overview of installation procedure: in the setup period, the system parameters, e.g., security parameters of the cryptographic schemes, are chosen. Each $I$ applies for a specific clearance level and is certified by $T$. This

[15] However, non-technical influence of $O$ on the user cannot be prevented, e.g., when $O$ advertises for $S_1$'s products.

[16] For example, $U$ cannot receive a legal license and later refuse payment, pretending he never received the license.

certification is performed once and repeated only if a new $I$ joins the system or existing certificates expire. In parallel, a SW provider who has developed a new SW component submits it to $T$ and requests certification of the SW properties. After certification, the SW provider establishes terms and conditions with $L$. Both steps need to be done for each new component.[17] Finally, the SW component is distributed to each $I$ via the broadcast channel. The actual installation starts with an installation request by $U$. $I$ checks if it has the necessary clearance level and, if so, obtains a license from $L$. After delivery of SW and license to $U$, $u_0$ checks if the SW, the license and $I$ are legal (for definitions see Sect. 4). If so, $u_0$ instructs the target component $u_i$ to install the SW. $u_0$ then confirms the successful installation to $I$ and awaits $I$'s acknowledgment. After receiving the acknowledgment, $u_0$ instructs $u_i$ to use the SW.

### 5.1 Conventions, Building Blocks and Message Formats

- $\mathsf{ID}()$ is a function that maps a principal or an object to a unique identifier.
- $\mathsf{Hash}()$ is a cryptographic hash function.
- $(\mathsf{GenKey_A}(), \mathsf{Sign}(), \mathsf{Verify}())$ denotes the key generation, signing and verifying of a digital signature scheme. $\sigma_X \leftarrow \mathsf{Sign}(k_X^{\text{sign}}; M)$ means the signing of the message $M$ with $X$'s signing key $k_X^{\text{sign}}$, resulting in the signature $\sigma_X = (M, \text{Sig}(M))$. $ind \leftarrow \mathsf{Verify}(k_X^{\text{test}}; \sigma_X)$ means the verification of $\sigma_X$ with the key $k_X^{\text{test}}$. The result of the verification is the Boolean value $ind$.
- $(\mathsf{GenKey_P}(), \mathsf{Reg}(), \mathsf{Enc_P}(), \mathsf{Dec_P}())$ is a tuple that denotes the key generation, user registration, encryption and decryption of a PKBE scheme. $\mathsf{GenKey_P}()$ is used by $T$ to set up all the parameters of the scheme, e.g., the set of all public keys $\mathcal{K}^{\text{enc}}$ which is available to any party. $\mathsf{Reg}()$ is used by $T$ to compute the set of secret keys $\mathcal{K}_U^{\text{dec}}$ to be delivered to a user $U$. $\mathsf{Enc_P}(\mathcal{K}^{\text{enc}}, \mathcal{U}; M)$ is used by a (not necessarily trusted) sender to encapsulate a message $M$ with the set of public keys $\mathcal{K}^{\text{enc}}$ in such a way that only the unrevoked users $\mathcal{U}$ can recover it. $\mathsf{Dec_P}(\mathcal{K}_U^{\text{dec}}; C)$ is used by a user $U$ to decipher $C$ with his private key set $\mathcal{K}_U^{\text{dec}}$ and returns $M$ if and only if the user is unrevoked, i.e., $U \in \mathcal{U}$.
- $(\mathsf{GenKey_S}(), \mathsf{Enc_S}(), \mathsf{Dec_S}())$ is a tuple that denotes a symmetric encryption scheme for key generation, encryption and decryption. The shared key of $X$ and $Y$ is denoted $k_{X,Y}$ (for details on sharing the key, cf. [1]).
- $\mathsf{MAC}(k_{X,Y}; M)$ is a function that calculates the Message Authentication Code (MAC) of message $M$ under the shared key $k_{X,Y}$ of $X$ and $Y$.
- $\text{Clear}(I)$ denotes the clearance level of the ISP $I$. $\text{Clear}_{\min}(s)$ denotes the minimum clearance level that is required for an ISP to install the SW $s$.
- $\mathsf{Comp}(U; P_1^s, P_2^s, \ldots)$ denotes a compatibility check function that returns `true` iff the requested SW $s$ and $U$ are compatible (cf. Sect. 4.1). The compatibility check $\mathsf{Comp}()$ is computed by $u_0$ based on the properties $P_i^s$ of $s$ (see Sect. 3.1 on p. 259). For this purpose, $u_0$ interprets those properties and

[17] However, a SW provider and $L$ might establish more general terms and conditions which cover a whole set of SW components.

derives requirements for $U$ such as interfaces, protocols, minimum memory and processing power, etc. If $U$ fulfills all of the requirements, $\mathsf{Comp}()$ returns `true` which confirms compatibility of $U$ and $s$. If any requirement is unfulfilled, $\mathsf{Comp}()$ returns `false`.[18]

- $\mathsf{Target}(U; P_1^s, P_2^s, \ldots)$ denotes a function which returns the target component $u_i$, $u_i \in \{u_1, \ldots, u_n\}$ on which the SW $s$ is to be installed.
- $\mathcal{R}_U = \{r_1^U, r_2^U, \ldots\}$ denotes the set of rights that $U$ asks for when it sends an installation request. An example for $r_i^U$ is a one-year validity period.
- $\mathsf{right}(\mathcal{R}_U(\sigma), i)$ denotes a separator which returns the right $r_i^U$ of $\mathcal{R}_U = \{r_1^U, r_2^U, \ldots\}$ where $\sigma$ is a signature on $\mathcal{R}_U$ or on a string containing $\mathcal{R}_U$.
- $instr_{u_i} \leftarrow \mathsf{install}(\mathsf{ID}(u_i), \mathsf{ID}(s), s_{\text{enc}}^{u_i})$ is an order from $u_0$ to $u_i$ to install $s_{\text{enc}}^{u_i}$.
- $\widetilde{instr}_{u_i} \leftarrow \mathsf{use}(\mathsf{ID}(u_i), \mathsf{ID}(s); p_1, p_2, \ldots)$ denotes an order from $u_0$ to $u_i$ to use the SW $s$ with the input parameters $(p_1, p_2, \ldots)$. For example, if $p_i \in \{0, 1\}$ represents a functionality of $s$, then this functionality is activated for $p_i = 1$ and deactivated for $p_i = 0$.[19] $u_0$ derives the parameters from the rights $\mathcal{R}_U$ granted in the license.

### 5.2 Protocols

**Setup.** The setup period consists of four phases A–D (cf. Fig. 1):

**Phase A:** Each ISP applies for certification of a particular clearance level.[20] The result of the certification process is the clearance level certificate $\zeta_I$, more precisely $\zeta_I \leftarrow \mathsf{Sign}(k_T^{\text{sign}}; \mathsf{ID}(I), k_I^{\text{test}}, \text{Clear}(I))$.

**Phase B:** $T$ sets up the PKBE scheme, publishes the public keys and provides each $U$ with its private keys. In addition, every party distributes its test key, e.g., using $T$ to certify and distribute the public keys. Each SW provider (either $O$ or $S$) encrypts any newly developed SW component $s$ for later distribution via the broadcast channel (1). He computes a hash $h \leftarrow \mathsf{Hash}(s)$, generates the SW signature $\sigma_{O|S}^{\text{SW}} \leftarrow \mathsf{Sign}(k_{O|S}^{\text{sign}}; h)$ and sends the property certification request $(s, \sigma_{O|S}^{\text{SW}})$ to $T$.[21] With $O|S$ we mean any of the two roles $O$ and $S$

$$s_{\text{enc}} \leftarrow \mathsf{Enc_P}(\mathcal{K}^{\text{enc}}, \mathcal{U}; s). \tag{1}$$

$T$ verifies the SW signature $\texttt{true} \stackrel{?}{=} \mathsf{Verify}(k_{O|S}^{\text{test}}; \sigma_{O|S}^{\text{SW}})$ and the hash value $h \stackrel{?}{=} \mathsf{Hash}(s)$. If both are valid, $T$ generates the SW property certificate $\zeta_s$ for the

[18] For example, `false` might be the result when $U$ and $s$ have inconsistent interfaces.
[19] In the automotive example, the functionality might be additional horsepower.
[20] Many certification models are possible, but we omit their discussion here. One example is a joint definition of clearance level requirements by $T$, $O$ and $S$, possibly including spokespersons of $I$ and official authorities.
[21] Care has to be taken in order to avoid security vulnerabilities when signing an encryption [18].

SW $s$ in (2). For example, the SW provider may submit the claimed properties of his SW which $T$ then verifies:[22]

$$\zeta_s \leftarrow \mathsf{Sign}(k_T^{\text{sign}}; \mathsf{ID}(s), P_1^s, P_2^s, \ldots), \quad P_1^s := \text{Clear}_{\min}(s),\ P_2^s := \mathsf{Hash}(s). \tag{2}$$

We use $\text{Clear}_{\min}(s)$ and $\mathsf{Hash}(s)$ as the first two properties because this simplifies the notation: both properties need to be certified by $T$.

**Phase C:** During this step, terms and conditions between the SW providers and $L$ are negotiated and committed. Afterwards $L$ can independently create licenses for any $U$ of the form:

$$\gamma_L \leftarrow \mathsf{Sign}(k_L^{\text{sign}}; \texttt{license}, \mathsf{ID}(U), \mathsf{ID}(s), \mathcal{R}_U). \tag{3}$$

**Phase D:** The SW provider signs the property certificate in order to commit to the properties of $s$. Finally, he broadcasts the encrypted SW component together with his signature $\sigma_{O|S}^{\text{comm}} \leftarrow \mathsf{Sign}(k_{O|S}^{\text{sign}}; \zeta_s)$.

**Installation of a SW Component.** After this setup phase, the installation procedure for a specific SW component can start:

1. In the first step, $U$ sends a signed installation request $\sigma_U^{\text{req}}$ to $I$. The request contains the identifier of the requested SW $s$ and the desired rights $\mathcal{R}_U$

$$\sigma_U^{\text{req}} \leftarrow \mathsf{Sign}(k_U^{\text{sign}}; \rho_U), \quad \rho_U = (\mathsf{ID}(U), \mathsf{ID}(s), \mathcal{R}_U) \text{ and } \mathcal{R}_U = \{r_1^U, \ldots\}. \tag{4}$$

2. $I$ verifies $U$'s signature $\texttt{true} \stackrel{?}{=} \mathsf{Verify}(k_U^{\text{test}}; \sigma_U^{\text{req}})$ and its own clearance level $\mathsf{Clear}(I) \stackrel{?}{\geq} \mathsf{Clear}_{\mathsf{min}}(s)$. If both are valid, $I$ obtains a license for $U$ from $L$ of the form (3) and signs the installation package $(\gamma_L, s_{\text{enc}})$ for $L$ and $I$ (5):

$$\sigma_I^{\text{inst}} \leftarrow \mathsf{Sign}(k_I^{\text{sign}}; \gamma_L, s_{\text{enc}}). \tag{5}$$

   If at least one condition remains unfulfilled, $I$ sends a signed denial to $U$.
3. In case of success, $I$ sends the tuple $(\sigma_I^{\text{inst}}, \zeta_I, \zeta_s, \sigma_{O|S}^{\text{comm}})$ to $U$ where $\zeta_I$ represents his clearance level certificate, $\zeta_s$ the SW properties certificate and $\sigma_{O|S}^{\text{comm}}$ the SW provider's commitment to $\zeta_s$.
4. The trusted component $u_0$ verifies that the SW $s$ was indeed requested (6), $I$ possesses an authentic clearance level certificate $\zeta_I$ (7), $I$ has the necessary clearance level (8), the SW property certificate $\zeta_s$ is authentic (9), the delivered SW component is identical to the SW component referred to in the property certificate (10), $s$ and $U$ are compatible (11), the SW provider has made a commitment to $\zeta_s$ (12), $I$ has added his signature $\sigma_I^{\text{inst}}$ (13), $I$ has delivered a legal license (14), which grants the requested rights $\mathcal{R}_U$ (15):

$$\texttt{true} \stackrel{?}{=} \exists\ \rho_U \text{ for } \mathsf{ID}(s) \tag{6}$$

[22] In a different trust model, $O$ might be the party that certifies SW properties. This would significantly reduce the workload on $T$. However, it would require all $S$ to trust $O$ or result in dispute if $O$ denied fair evaluation.

$$\texttt{true} \stackrel{?}{=} \mathsf{Verify}(k_T^{\text{test}}; \zeta_I) \tag{7}$$

$$\mathsf{Clear}(I) \stackrel{?}{\geq} \mathsf{Clear}_{\mathsf{min}}(s) \tag{8}$$

$$\texttt{true} \stackrel{?}{=} \mathsf{Verify}(k_T^{\text{test}}; \zeta_s) \tag{9}$$

$$P_2^s \stackrel{?}{=} \mathsf{Hash}(s) \tag{10}$$

$$\texttt{true} \stackrel{?}{=} \mathsf{Comp}(U; P_1^s, P_2^s, \ldots) \tag{11}$$

$$\texttt{true} \stackrel{?}{=} \mathsf{Verify}(k_{O|S}^{\text{test}}; \sigma_{O|S}^{\text{comm}}) \tag{12}$$

$$\texttt{true} \stackrel{?}{=} \mathsf{Verify}(k_I^{\text{test}}; \sigma_I^{\text{inst}}) \tag{13}$$

$$\texttt{true} \stackrel{?}{=} \mathsf{Verify}(k_L^{\text{test}}; \gamma_L) \tag{14}$$

$$\mathsf{right}(\mathcal{R}_U(\gamma_L), i) \stackrel{?}{=} \mathsf{right}(\mathcal{R}_U(\rho_U), i) \quad \forall i. \tag{15}$$

If all conditions are fulfilled, $u_0$ finds the appropriate subset of the PKBE scheme and decrypts $s_{\text{enc}}$ with the corresp. private key: $s \leftarrow \mathsf{Dec_P}(\mathcal{K}_U^{\text{dec}}; s_{\text{enc}})$ . Then $u_0$ determines the target ECU $u_i$ in (16) and re-encrypts $s$ for $u_i$ with a symmetric key $k_{u_0,u_i}$ shared only with $u_i$.[23] Subsequently, $u_0$ invokes $u_i$ to install the SW component by sending the tuple $(instr_{u_i}, mac_{u_i})$ in (17). The message $instr_{u_i}$ provides $u_i$ with the encrypted SW via $U$'s internal communication network. The MAC $mac_{u_i}$ confirms the authenticity of $instr_{u_i}$ while $mac_{u_0}$ is $u_i$'s confirmation to $u_0$ that $s$ was successfully installed:

$$u_i \leftarrow \mathsf{Target}(U; P_1^s, P_2^s, \ldots) \quad \text{with} \quad u_i \in \{u_1, \ldots, u_n\} \tag{16}$$

$$s_{\text{enc}}^{u_i} \leftarrow \mathsf{Enc_S}(k_{u_0,u_i}; s) \tag{17}$$

$$instr_{u_i} \leftarrow \mathsf{install}(\mathsf{ID}(u_i), \mathsf{ID}(s), s_{\text{enc}}^{u_i}) \tag{18}$$

$$mac_{u_i} \leftarrow \mathsf{MAC}(k_{u_0,u_i}; instr_{u_i}) \tag{19}$$

$$mac_{u_0} \leftarrow \mathsf{MAC}(k_{u_0,u_i}; \mathsf{ID}(s)). \tag{20}$$

5. After the installation, $U$ confirms the result of the installation request $\rho_U$. For this purpose, $u_0$ uses the indicator $ind \in \{\texttt{true}, \texttt{false}\}$ where `true` represents success and `false` represents failure. $u_0$ adds the signature $\sigma_U^{\text{conf}}$ and sends $(\rho_U, \gamma_L, ind, \sigma_U^{\text{conf}})$ to $I$, where $\sigma_U^{\text{conf}} \leftarrow \mathsf{Sign}(k_U^{\text{sign}}; \rho_U, \gamma_L, ind)$
6. $I$ verifies the confirmation in (21) and forwards $U$'s confirmation to $L$. $I$ also sends an acknowledgment back to $U$ (22). Within $U$, $u_0$ checks the acknowledgment (23) and, if it is authentic, $u_0$ invokes $u_i$ to use the SW component with parameters $p_1, p_2, \ldots$ (24):

$$\texttt{true} \stackrel{?}{=} \mathsf{Verify}(k_U^{\text{test}}; \sigma_U^{\text{conf}}) \tag{21}$$

$$\sigma_I^{\text{ack}} \leftarrow \mathsf{Sign}(k_I^{\text{sign}}; \sigma_U^{\text{conf}}) \tag{22}$$

$$\texttt{true} \stackrel{?}{=} \mathsf{Verify}(k_I^{\text{test}}; \sigma_I^{\text{ack}}) \tag{23}$$

[23] The generation of $k_{u_0,u_i}$ will be detailed in [1]. Meanwhile, we assume it exists.

$$\widetilde{instr}_{u_i} \leftarrow \mathsf{use}(\mathsf{ID}(u_i), \mathsf{ID}(s); p_1, p_2, \ldots) \tag{24}$$

$$\widetilde{mac}_{u_i} \leftarrow \mathsf{MAC}(k_{u_0,u_i}; \widetilde{instr}_{u_i}). \tag{25}$$

After receiving and verifying the instruction $(\widetilde{instr}_{u_i}, \widetilde{mac}_{u_i})$, $u_i$ uses the new SW component $s$ with parameters $(p_1, p_2, \ldots)$. $u_0$ stores all licenses and periodically checks if any of them has expired. When a license expires, $u_0$ tells $u_i$ to execute the SW with different parameters. For example, the new parameters might instruct $u_i$ to stop using the SW or switch off some functionality, e.g., the additional horsepower in the automotive example. In addition, $u_0$ indicates the need for a new license to the user.
If the installation failed, $U$ uses the old platform configuration.

## 6 Assumptions, Security Analysis and Implementation

Due to space constraints, we present these sections in the full paper [1].

## 7 Conclusion

In this paper we have proposed a procedure for secure SW delivery and installation in embedded systems. It integrates installation service providers as intermediaries between SW provider and embedded system and categorizes them into separate clearance levels. Compatibility of the SW component and the target system is checked prior to installation. The fulfillment of a variety of requirements and the introduction of an elementary license system allows any SW provider to establish new business models that are currently not supported. The SW provider's intellectual property is protected and a variety of digital rights is supported. From the embedded system owner's point of view, the procedure prevents installation of illegal SW and supports warranty claims against the SW provider in case of defective SW with unambiguous evidence. Public Key Broadcast Encryption enables efficient communication with embedded system on an insecure one-way channel. The use of Trusted Computing concepts induces the necessary trust in the embedded system.

## References

1. Adelsbach, A., Huber, U., Sadeghi, A.R.: Secure software delivery and installation in embedded systems. Full version, (`http://www.prosec.rub.de/publications`)
2. Heinisch, C., Simons, M.: Loading flashware from external interfaces such as CD-ROM or W-LAN and programming ECUs by an on-board SW-component (SAE Technical Paper Series 2004-01-0678). [20] URL `http://www.sae.org/`.
3. Heinrich, A., Müller, K., Fehrling, J., Paggel, A., Schneider, I.: Version management for transparency and process reliability in the ECU development. [19] 219–230

4. Schmitt, M.: Software-update, configuration and programming of individual vehicles on the aftermarket with an intelligent data-configurator. [19] 1021–1046
5. Alminger, H., Josefsson, O.: Software handling during the vehicle lifecycle. [19] 1047–1055
6. Huber, M., Weber, T., Miehling, T.: Standard software for in-vehicle flash reprogramming. [19] 1011–1020
7. Oeftiger, U.: Diagnose und Reparatur elektronisch unterstützter Fahrzeuge. [8]
8. Euroforum, ed.: Jahrestagung Elektronik-Systeme im Automobil, Fachtag Design – Test – Diagnose elektronischer Systeme, Munich (2004)
9. BMW Car IT: Das Potenzial von Software im Fahrzeug. Press report, BMW Group, URL `http://www.bmw-carit.de/pdf/plakate.pdf` (2002)
10. Stölzl, S.: Software products for vehicles. [19] 1073–1088
11. Dodis, Y., Fazio, N.: Public key broadcast encryption for stateless receivers. In Feigenbaum, J., ed.: Digital Rights Management Workshop. Volume 2696 of Lecture Notes in Computer Science., Springer Verlag (2003) 61–80
12. Fiat, A., Naor, M.: Broadcast encryption. Advances in Cryptology—CRYPTO '93 Proceedings, Lecture Notes in Computer Science **773** (1994) 480–491
13. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. Advances in Cryptology—CRYPTO '01 Proceedings, Lecture Notes in Computer Science **2139** (2001) 41–62
14. Sadeghi, A.R., Stüble, C.: Property-based attestation for computing platforms: Caring about properties, not mechanisms. (2004)
15. Daimler Chrysler AG: Functional specification of a flash driver version 1.3. Specification, Herstellerinitiative Software, URL `http://www.automotive-his.de/download/HIS\%20flash\%20driver\%20v130.pdf` (2002)
16. Dallmayr, C., Schlüter, O.: ECU software development with diagnostics and flash down-loading according to international standards (SAE Technical Paper Series 2004-01-0273). [20] URL `http://www.sae.org/`.
17. Müller, M.: IT-Security in Fahrzeugnetzen. Elektronik Automotive (2004) 54–59 ISSN: 1614-0125.
18. An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: EUROCRYPT '02, Springer-Verlag (2002) 83–107
19. VDI Society for Automotive and Traffic Systems Technology, ed.: Electronic Systems for Vehicles. In VDI Society for Automotive and Traffic Systems Technology, ed.: Electronic Systems for Vehicles, VDI Berichte 1789, Congress, Baden-Baden, Germany, VDI Verlag GmbH Düsseldorf (2003)
20. Society of Automotive Engineers (SAE), ed.: SAE World Congress. In Society of Automotive Engineers (SAE), ed.: 2004 SAE World Congress, Detroit, Michigan, March 8–11, 2004, Detroit, Michigan (2004) URL `http://www.sae.org/`.

# A Restricted Multi-show Credential System and Its Application on E-Voting

Joseph K. Liu[1] and Duncan S. Wong [2,⋆]

[1] Department of Information Engineering,
The Chinese University of Hong Kong Shatin, Hong Kong
ksliu@ie.cuhk.edu.hk
[2] Department of Computer Science,
City University of Hong Kong Kowloon, Hong Kong
duncan@cityu.edu.hk

**Abstract.** A multi-show credential system allows a user to unlinkably and anonymously demonstrate the possession of a credential as many times as the user desires. In some applications, this could be too flexible to be useful. In this paper, we propose a restricted version of such a system. The restricted multi-show credential system only allows the user to demonstrate his possession of a credential once in a given period of time. This time period can also be quantified to a sequence of discrete events. That is, each credential can only be shown once in each event. However, the same credential can still be shown anonymously in another event without being linked. On its applications, we propose a restricted multi-show credential based e-voting system. The e-voting system has the following desirable properties. (1) Simplicity: each user only registers once when he first joins the system and no additional registration/setup phase is needed for the user before casting a vote in each subsequent voting event. (2) Flexibility: the set of eligible voters can be different for different voting events with no additional overhead. (3) Unlinkability: the voters among different voting events cannot be linked. (4) Efficiency : The system maintains the same order of efficiency no matter a voting event is "yes/no" type, "1-out-of-n" type or even "t-out-of-n" type. Furthermore, we show how to extend the e-voting system into an electronic questionnaire system.

## 1 Introduction

A credential system, introduced by Chaum [10], allows a user to obtain a credential from an organization and demonstrate the possession of the credential to a verifier anonymously. That is, the verifier cannot get the identity of the user, even if the verifier colludes with the organization which issues the credential.

⋆ The author's work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. 9040904 (RGC Ref. No. CityU 1161/04E )).

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 268–279, 2005.

There are two types of credential systems: the one-show credential system and the multi-show credential system. Possession of a one-show credential can only be demonstrated once. If it is 'double-spent', it can be detected or anonymity of the owner would be compromised. Possession of a multi-show credential can be demonstrated for an arbitrary number of times without being linked and the anonymity of the owner would not be compromised. Camenisch and Lysyanskaya [5] proposed the first practical multi-show credential system in 2001. Their system allows a user to unlinkably demonstrate possession of a credential as many times as necessary. In some applications however, this reusability property may be too flexible to be useful, especially in applications related to e-voting.

E-voting systems have been studied for more than twenty years. Since the first e-voting system [9] proposed in 1981, a number of systems with various functional and security properties have been proposed. A popular definition of a secure e-voting system is given in [13]. There are seven requirements: completeness, soundness, privacy, unreusability (detecting double voting), eligibility, fairness and verifiability. Completeness implies all valid votes should be counted correctly. Soundness implies all invalid votes should not be counted. Privacy means all votes should be kept secret. Unreusability prevents any voter to vote twice or more Eligibility means no unauthorized entity can vote. Fairness implies nothing can affect the result. Verifiability ensures the voting result can be publicly verified. Recent results pointed out some new requirements. One is receipt-freeness [2, 15]. A receipt-free e-voting system prevents a voter from proving to others that he has cast a particular vote. Another requirement is non-transferability [8]. In most of the e-voting systems, the voting right can be transferred because the authentication document is irrelevant to the voter. A non-transferable e-voting system ensures that the transfer of the voting right is equivalent to the transfer of all the information privately owned by the voter.

A detailed survey of different types of e-voting system can be found in [18, 17].

### 1.1 Our Contributions

We propose a *restricted* version of an anonymous multi-show credential system. The restricted credential system only allows a user to demonstrate his possession of a credential once in a given period of time. If we fix the duration of the period of time and synchronize the periods for all users, we would be able to quantify a sequence of synchronized fix-length periods of time to a sequence of discrete events and identify each event uniquely. In this way, the restricted credential system allows each credential to be shown for only once in each event. However, the same credential can still be shown anonymously in another event without being linked. Hence our system still allows a user to demonstrate the possession of a credential in as many events as necessary. To our best knowledge, it is the first time of introducing this concept.

The motivation of proposing the restricted multi-show credential system is from its promising application on e-voting systems. Comparing with previous e-voting systems, a restricted multi-show credential based e-voting system has advantages in the following aspects. First, the system does not need a trusted

third party (TTP) to protect voters' privacy. The mechanism from the restricted multi-show credential system facilitates the privacy protection. Second, the system only needs each user to register once. After this one-time registration, the user can cast a vote in multiple voting events. Hence the system is more user-friendly. Without a TTP and requiring only one-time registration for each voter, e-voting system becomes very effective and easy to maintain. Third, the system does not leak any information on who has voted and who has not, and voters among different voting events are unlinkable. Fourth, the performance of the system is almost the same no matter the votes are "yes/no" type, 1-out-of-$n$ type or even $t$-out-of-$n$ type. In addition to these, our proposed system satisfies all the desirable properties of a secure e-voting system [13, 2, 15, 8]. We also show how to modify our proposed e-voting system into an electronic questionnaire system.

The remaining of the paper is organized as follows. In Sec. 2, we present a restricted multi-show credential system. In Sec. 3, an e-voting system based on the restricted multi-show credential system is presented. It is followed by an electronic questionnaire system in Sec. 4. Finally we conclude the paper with some remarks in Sec. 5.

## 2 A Restricted Multi-show Credential System

A credential system has *users*, *organizations*, and *verifiers* as types of entities. A user $U$ obtains a credential from an organization $O$ (the credential issuer) after identifying herself using a pseudonym. $U$ then demonstrates his possession of a credential granted by $O$ to a verifier $V$ or another organization. For more details, we refer the readers to the paper written by Camenisch and Lysyanskaya [5]. In the following, we introduce a set of proof of knowledge notations and techniques that are used in our construction.

### 2.1 Proof of Knowledge of Discrete Logarithms and Representations

In the description of the proof of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms, we use the notations introduced by Camenisch and Stadler [6].

$$PK\{(\alpha, \beta, \gamma) : y = g^{\alpha}h^{\beta} \wedge \tilde{y} = \tilde{g}^{\alpha}\tilde{h}^{\gamma} \wedge (u \leq \alpha \leq v)\}$$

denotes a zero-knowledge Proof of Knowledge of integers $\alpha, \beta$ and $\gamma$ such that $y = g^{\alpha}h^{\beta}$ and $\tilde{y} = \tilde{g}^{\alpha}\tilde{h}^{\gamma}$, where $u \leq \alpha \leq v$ and $y, g, h, \tilde{y}, \tilde{g}, \tilde{h}$ are elements of some group $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$.

The convention is that Greek letters denote quantities the knowledge of which is being proved, while all other parameters are known to the verifier. Using this notation, a proof-protocol can be described by just pointing out its aim while hiding the details. (The implementation details of the proof-protocol can be found in [3, 4, 7, 14].)

By using the standard method of hashing due to Fiat and Shamir [12] and assuming that each hash function behaves like a random oracle [1], such a proof-protocol can be turned into a secure signature scheme or an undeniable proof of knowledge scheme (that is, cannot be simulated [11]). We use the notation

$$UPK\{(\alpha) : y = g^{\alpha}\}(v)$$

to denote an undeniable proof of knowledge for a particular event with a flag $v$ which will be explained later.

In this paper, we apply $PK$ and $UPK$ to a group of quadratic residues modulo a composite $n$, that is, $G = QR_n$. The prover needs to convince the verifier that the elements he presents are indeed quadratic residues. This can be done by executing $PK\{(\alpha) : y^2 = (g^2)^{\alpha}\}$. The quantity $\alpha$ is defined by $\log_{g^2} y^2$, which is the same as $\log_g y$ in case of $y \in QR_n$.

### 2.2 Security Requirements

Besides all the relevant desirable properties of a multi-show credential system stated in [5], that is, correctness, unforgeability and anonymity, a restricted version requires two new notions on the area of linkability: *inter-event unlinkability* and *intra-event linkability*. Correctness means that an honest verifier always accepts if the credential is granted by an honest organization. Unforgeability means that it is negligible to generate a valid credential of an organization if the secret key of the organization is not known. Consistency of credentials [5] should also be ensured. Anonymity means that no entity can compute the identity of the owner from a valid credential, even the secret key of the credential issuer is given. In the following, we describe the additional two properties for a restricted multi-show credential system in more detail.

1. **Inter-event Unlinkability:** A credential system is unlinkable among different events if a credential owner presents his credential only once in each of the events. No one can figure out there is a particular person who has presented his credential at different event with non-negligible probability, even the secret key of the credential issuer is given.
2. **Intra-event Linkability:** A credential is linkable if the credential is presented by its owner for more than once in a particular event. Everyone (knowing public information) can determine if two shows of a credential are corresponding to the same credential if the two shows are within the same event. However, it is not necessary to revoke the anonymity of the credential owner. We only require the detection of double-spending of a credential within the same event.

### 2.3 System Parameters and Issuer Key Generation

The system parameters of our restricted multi-show credential system follows that of the credential system by Camenisch and Lysyanskaya [5]. Let the length of all the RSA moduli be $\ell_n$. Define integer intervals $\Gamma = ]-2^{\ell_\Gamma}, 2^{\ell_\Gamma}[$, $\Delta =$

$]-2^{\ell_\Delta}, 2^{\ell_\Delta}[$, and $\Lambda = ]2^{\ell_\Lambda}, 2^{\ell_\Lambda+\ell_\Sigma}[$ where $\ell_\gamma = 2\ell_n$, $\ell_\Delta = \epsilon(4\ell_n+3)$, $\epsilon > 1$ is a security parameter, and $\ell_\Lambda > \ell_\Sigma + \ell_\Delta + 4$.

The organization $O$, which issues credentials, chooses randomly $\ell_n/2$-bit primes $p'$ and $q'$ such that $p = 2p'+1$ and $q = 2q'+1$ are prime and $n = pq$. It picks $a, b, d, g, h, z \in_R QR_n$ where $QR_n$ defines the group of quadratic residues modulo $n$. It also picks a hash function $H : \{0,1\}^* \rightarrow QR_n$. For security analysis, $H$ is considered to be a random oracle [1]. It stores the secret key $SK = (p, q)$ and publishes the public key $PK = (n, a, b, d, g, h, z, H)$. Besides $H$, all the parameters in $PK$ are assumed to have been verified properly using methods mentioned in [5]. The parameter $\ell_\Lambda$ should be chosen such that computing discrete logarithms in $QR_n$ with $\ell_\Lambda$-bit exponent is hard.

### 2.4 Generation of a Pseudonym

A user $U$ establishes a pseudonym $N_{(U,O)}$ and a validating tag $P_{(U,O)}$ with organization $O$. For enforcing the constraint of event, which will be explained shortly, we extend the formation of $P_{(U,O)}$ and the corresponding protocol steps in such a way that $P_{(U,O)}$ will commit not only $U$'s secret key $x_U \in \Gamma$ and $s_{(U,O)} \in \Delta$ but also an additional parameter $t_{(U,O)} \in \Delta$, i.e. $P_{(U,O)} = a^{x_U} b^{s_{(U,O)}} z^{t_{(U,O)}}$. Similar to the formation of $s_{(U,O)}$, $t_{(U,O)}$ is also constructed from contributions of both $U$ and $O$ and only $U$ knows the value of $t_{(U,O)}$. For completeness, we describe the modified protocol as follows. For those who are familiar with the original protocol of [5] and have figured out the actual construction from the idea above, the following can safely be skipped.

1. $U$ chooses a value $N_1 \in \{0,1\}^k$, and values $r_1, t_1 \in_R \Delta$ and $r_2, r_3, r_4 \in_R \{0,1\}^{2\ell_n}$. $U$ sets $C_1 = g^{r_1}h^{r_2}$, $C_2 = g^{x_U}h^{r_3}$, $C_3 = g^{t_1}h^{r_4}$, and sends ($N_1$, $C_1$, $C_2$, $C_3$) to $O$.
2. $U$ serves as the prover to show $O$ that

$$PK\{(\alpha,\beta,\gamma,\delta,\epsilon,\zeta) : {C_1}^2 = (g^2)^\alpha (h^2)^\beta \wedge {C_2}^2 = (g^2)^\gamma (h^2)^\delta \wedge {C_3}^2 = (g^2)^\epsilon (h^2)^\zeta\}$$

3. $O$ chooses a random $N_2$, $r, t \in_R \Delta$ and sends $(N_2, r, t)$ to $U$.
4. $U$ sets her pseudonym $N_{(U,O)} := N_1 || N_2$. $U$ computes $s_{(U,O)} = (r + r_1 \bmod (2^{\ell_\Delta+1}-1)) - (2^{\ell_\Delta}-1)$, $t_{(U,O)} = (t + t_1 \bmod (2^{\ell_\Delta+1}-1)) - (2^{\ell_\Delta}-1)$, and sets her validating tag $P_{(U,O)} = a^{x_U} b^{s_{(U,O)}} z^{t_{(U,O)}}$ and sends $P_{(U,O)}$ to $O$.
5. $U$ computes $\tilde{s} = \lfloor \frac{r_1+r}{2^{\ell_\Delta+1}-1} \rfloor$, $\tilde{t} = \lfloor \frac{t_1+t}{2^{\ell_\Delta+1}-1} \rfloor$, picks $r_5, t_6 \in_R \{0,1\}^{\ell_n}$, sets $C_4 = g^{\tilde{s}}h^{r_5}$, $C_5 = g^{\tilde{t}}h^{r_6}$, and sends $(C_4, C_5)$ to $O$.
6. $U$ shows the $P_{(U,O)}$ is formed correctly by showing to $O$ that

$$PK\{(\alpha,\beta,\gamma,\delta,\mu,\eta,\epsilon,\zeta,\ \rho\ ,\phi,\vartheta,\xi,\psi,\omega) : C_1^2 = (g^2)^\alpha (h^2)^\beta \wedge C_2^2 = (g^2)^\gamma (h^2)^\delta \wedge C_3^2 = (g^2)^\mu (h^2)^\eta \wedge C_4^2 = (g^2)^\epsilon (h^2)^\zeta \wedge C_5^2 = (g^2)^\rho (h^2)^\phi \wedge \frac{C_1^2 (g^2)^{r-2^{\ell_\Delta}+1}}{(C_4^2)^{2^{\ell_\Delta+1}-1}} = (g^2)^\vartheta (h^2)^\xi \wedge$$

$$\frac{C_3^2(g^2)^{r-2^{\ell_\Delta}+1}}{(C_5^2)^{2^{\ell_\Delta+1}-1}} = (g^2)^{\psi}(h^2)^{\omega} \wedge P^2_{(U,O)} = (a^2)^{\gamma}(b^2)^{\vartheta}(z^2)^{\psi} \wedge$$
$$\gamma \in \Gamma \wedge \vartheta \in \Delta \wedge \psi \in \Delta\}$$

7. $O$ stores $N_{(U,O)}$, $P^2_{(U,O)}$ and $P_{(U,O)}$.
8. $U$ stores $N_{(U,O)}$, $P^2_{(U,O)}$, $P_{(U,O)}$, and $s_{(U,O)}$.

### 2.5 Generation of a Credential

User $U$ identified by $(N_{(U,O)}, P_{(U,O)})$ can communicate with organization $O$ for getting a pair $(c_{(U,O)}, e_{(U,O)}) \in \mathbb{Z}_n^* \times \Lambda$ such that $P_{(U,O)}d = c_{(U,O)}^{e_{(U,O)}}$. The protocol is identically the same as Protocol 2 of [5–Sec. 4.4]. We skip the details.

### 2.6 Showing a Credential in an Event

Suppose user $U$ wants to prove to a verifier $V$ the possession of a credential issued by $O$ in a single event without revealing the actual values of the possession, i.e. $(P_{(U,O)} = a^{x_U} b^{s_{(U,O)}} z^{t_{(U,O)}},\ c_{(U,O)},\ e_{(U,O)})$, where $P_{(U,O)}d = c_{(U,O)}^{e_{(U,O)}}$. The technique is the same as Protocol 3 of [5–Sec. 4.4]. The actual protocol is modified slightly to accommodate the change of the formation of $P_{(U,O)}$. In the following, we assume that $U$ and $V$ are connected with an anonymous channel [9, 19].

1. $V$ publishes a string EventInfo containing the event information such as the date or the name of the event. In general, it can be an arbitrary string provided that it is unique among all the events in the credential system.
2. $U$ computes $v = H(\mathsf{EventInfo})$, which we call it an *event ID*. Assume that computing the discrete logarithm of $v$ to the base of any other event ID in $QR_n$ is hard.
3. $U$ selects $r_1, r_2 \in_R \{0,1\}^{2\ell_n}$, computes $A = c_{(U,O)}h^{r_1}$, $B = h^{r_1}g^{r_2}$, $E = v^{t_{(U,O)}}$, and sends $(A, B, E)$ to $V$.
4. $V$ checks whether $E$ is in its database. If not, stores $E$ in the database. Otherwise, terminates the process with failure.
5. $U$ proves to $V$ that

$$UPK\{(\alpha,\beta,\gamma,\varphi,\delta,\epsilon,\zeta,\xi) : d^2 = (A^2)^{\alpha}(\frac{1}{a^2})^{\beta}(\frac{1}{b^2})^{\gamma}(\frac{1}{z^2})^{\varphi}(\frac{1}{h^2})^{\delta} \wedge$$
$$B^2 = (h^2)^{\epsilon}(g^2)^{\zeta} \wedge 1 = (B^2)^{\alpha}(\frac{1}{h^2})^{\delta}(\frac{1}{g^2})^{\xi} \wedge$$
$$E^2 = (v^2)^{\varphi} \wedge\ \beta \in \Gamma \wedge \gamma \in \Delta \wedge \varphi \in \Delta \wedge \alpha \in \Lambda\}(v)$$

*Remark:* The purpose of $v$ is to identify each event. It ensures that each event has a distinct event ID. If $U$ participates in two events with event IDs $v$ and $v'$ $(v \neq v')$, he computes $E = v^{t_{(U,O)}}$ and $E' = v'^{t_{(U,O)}}$ respectively. Based on the discrete log assumption above, it is hard to find an $\alpha \in \mathbb{Z}_n^*$ such that $v = v'^{\alpha}$. The purpose is to prevent anyone from being able to trace/link a user who participates in both events.

We skip the description of the protocol of showing a credential with respect to a pseudonym due to the similarity of the technique as above and that of [5–Sec. 4.6].

## 3 An E-Voting System

We now propose an e-voting system which is based on the restricted multi-show credential system described above. The system also uses ring signature as a primitive and assumes that each voter has a tamper resistant hardware device. The restricted multi-show credential system provides a mechanism to detect double voting and the ring signature scheme provides anonymous eligibility checking of voters. The tamper resistant hardware device is used to ensure that the system is receipt-free. Hence if this property is not important in the target application, the requirement of having tamper resistant hardware devices can be removed. In the following, we first give a brief review on ring signature and tamper resistant hardware devices.

**Ring Signatures.** A ring signature scheme allows a user who has a public key pair to spontaneously form a group of users by including his own public key and the public keys of all other group members, and generate a signature on a message so that any unbounded algorithm cannot determine the actual signer among the group members. The formation of the group is spontaneous in such a way that the group members can be totally unaware of being included and no group membership collaboration is needed. The scheme only assumes that the public keys of all the users are publicly known. The concept of ring signature was first found in [11], in the context of proof of knowledge protocol. It appeared as the forum of ring signature in [20].

**Tamper Resistant Hardware Devices.** A tamper resistant device has a security architecture which is used to store secret information in such a way that these information can only be accessed by internal functions of the device but cannot be retrieved from the device. Functions stored in the device can be invoked as it is considered as a probabilistic oracle in [21]. Signing and decryption are examples of such an oracle. In addition, a PIN is needed in order to carry out the oracle. This is to protect the owner against other malicious user who gets access to the device but does not know the PIN. In [16], an e-voting system based on tamper resistant devices was proposed and demonstrated to be user friendly and easy to implement.

### 3.1 System Description and Overview

The entities involved in one voting process include a User ($U$), a Voting Pass Issuing Organization ($O$), a Voting Centre ($V$) and optionally an Arbitrator ($A$). Assume that each entity has a public key pair backed by the PKI. $O$ is responsible for issuing voting passes and tamper resistant devices to users. $V$ provides a bulletin board for posting votes. Once something is posted on this board, it is assumed to have the information publicly known and cannot be altered further[1]. $A$ is a trusted authority responsible for opening each vote and the presence of $A$ is optional which will be explained shortly.

[1] Consider $V$ as a website with publicly known URL.

There are three protocols: Hardware Delivery Protocol, Voting Protocol and Vote-Open Protocol. We outline the functions of each protocol in the following.

In the Hardware Delivery Protocol, $U$ obtains a voting pass (a restricted multi-show credential) from $O$ and stores it in his tamper resistant device. For each user, the Hardware Delivery Protocol is carried out only once.

$U$ can conduct the Voting Protocol in multiple voting events. In each run of the Voting Protocol, $U$ shows his voting pass to $V$ through an anonymous channel [9, 19]. The protocol makes sure that no one can vote for twice or more in a single voting event while preserving unlinkability when $U$ votes in multiple voting events. $U$ also generates a ring signature on her voting-choice using her private key and the public keys of all other eligible voters. This is for showing that $U$ is a legitimate voter without revealing her identity. If $A$ is present, the election-choice is also encrypted using $A$'s event-based public key so that no one can obtain the vote during the voting phase.

The Vote-Open Protocol is carried out at the end of each voting event. During the protocol run, $A$ reveals its event-based private key so that all the votes are publicly verifiable. The public, or $A$, then verifies the ring signature and counts the votes.

The set of eligible voters may vary from one voting event to another. For example, each student of a university obtains his voting pass at the beginning of a school year. He can use it in an election of the student union of the university. Then he can *reuse* his voting pass in another election, say his departmental society. Yet, his votes cannot be linked, no one knows if he has voted in any of these two voting events or not, and his identity is protected every time when he casts a vote. Moreover, the student who gives his voting pass to his friend means giving the right of using his secret key to his friend.

**Optional Features.** The arbitrator $A$ can be removed if all the voters participate in the Vote-Open Protocol to reveal their vote. This will be discussed further in Sec. 3.5. Similarly, the tamper resistant device becomes optional if receipt-freeness is not necessary in the target application. To ensure receipt-freeness, it is assumed that the voter's private key is stored in his tamper resistant device and the Voting Protocol is carried out entirely by the device using its own random source which cannot be accessed from the outside. It cannot be used as a separated function. For example, ring signature generation is assumed to be bundled in the voting protocol implementation. The device is issued by an authorized organization $O$. Voter can only activate it by choosing a PIN with some initial configurations.

### 3.2 Hardware Delivery Protocol

$U$ communicates with $O$ to obtain a restricted multi-show credential. As mentioned above, it is assumed, for the assurance of receipt-freeness, that the tamper resistant device issued by $O$ (for example, the student union of the university or the Immigration Department of the government) actually carries out the protocol on behalf of $U$. In the rest of the paper, we do not make explicit distinction

between the tamper resistant device and $U$. The steps between $U$ and $O$ are the same as the restricted multi-show credential system in Sec. 2.4 and 2.5.

### 3.3 Voting Protocol

Assume that $U$ and $V$ are connected with some perfectly anonymous channel [9, 19]. Let $H_1 : \{0,1\}^* \rightarrow QR_n$ and $H_2 : \{0,1\}^* \rightarrow \{0,1\}^k$ be full-domain hash functions viewed as distinct random oracles [1]. Let $\mathsf{SetOfVote} = \{C_1, \cdots, C_N\}$ be a set of all the choices in a voting event. Let $\mathcal{L}$ denote the set of public keys of all the eligible voters. We use $\mathsf{Sign}_{Bob}(m)$ to denote the signature on a message $m$ generated by Bob using his private key and $\mathsf{RingSign}_{Bob,\mathcal{L}}(m)$ to denote the ring signature on $m$ generated using Bob's private key and the public keys in $\mathcal{L}$, which includes Bob's public key. Let $\mathcal{E}_A(m)$ denote an encrypted message $m \in \mathsf{SetOfVote}$ using the public key of an arbitrator $A$.

Suppose $U$ who has credential $(x_U, s_{(U,O)}, t_{(U,O)}, c_{(U,O)}, e_{(U,O)})$ wants to vote in a voting event. $U$ conducts the following protocol with $V$.

1. $V$ publishes a string $\mathsf{VoteInfo}$ containing voting information such as the set of candidates and the date of the vote. $\mathsf{VoteInfo}$ is assumed to be unique among all voting events in the system.
2. $U$ computes the Voting Event ID $v = H_1(\mathsf{VoteInfo})$ and gets the public keys of all the eligible voting members from $V$. $U$ then selects $s \in_R \{0,1\}^k$ and picks $m \in \mathsf{SetOfVote}$, computes $m' \leftarrow \mathcal{E}_A(m)$ and $c = H_2(s, m', \mathsf{VoteInfo})$, and sends $c$ to $V$.
3. $V$ selects $r \in_R \{0,1\}^k$, computes $c' \leftarrow \mathsf{Sign}_V(c, r)$, and publishes $c$.
4. $U$ selects $r_1, r_2 \in_R \{0,1\}^{2\ell_n}$, computes $A = c_{(U,O)} h^{r_1}$, $B = h^{r_1} g^{r_2}$, $E = v^{t_{(U,O)}}$, and sends $(A, B, E)$ to $V$.
5. $V$ checks whether $E$ is in its database. If not, stores $E$ in the database. Otherwise, rejects $U$'s connection.
6. $U$ proves to $V$ that

$$UPK\{(\alpha, \beta, \gamma, \varphi, \delta, \epsilon, \zeta, \xi) : d^2 = (A^2)^{\alpha}(\frac{1}{a^2})^{\beta}(\frac{1}{b^2})^{\gamma}(\frac{1}{z^2})^{\varphi}(\frac{1}{h^2})^{\delta} \wedge$$
$$B^2 = (h^2)^{\epsilon}(g^2)^{\zeta} \wedge 1 = (B^2)^{\alpha}(\frac{1}{h^2})^{\delta}(\frac{1}{g^2})^{\xi} \wedge$$
$$E^2 = (v^2)^{\varphi} \wedge \beta \in \Gamma \wedge \gamma \in \Delta \wedge \varphi \in \Delta \wedge \alpha \in \Lambda\}(v)$$

7. If the proof of knowledge above is passed successfully, $V$ publishes all the conversations between $V$ and $U$, computes $T \leftarrow \mathsf{Sign}_V(c, c', r, v, A, B, E)$, and sends $(T, c', r)$ to $U$.
8. $U$ verifies if $c'$ and $T$ are valid signatures, and check whether $V$ has published the conversation. If any of the checks fails, $U$ terminates the process. Otherwise, $U$ computes $R \leftarrow \mathsf{ring\text{-}Sign}_{U,\mathcal{L}}(T, c, v)$ and publishes $\{T, s, m', r, R\}$ on $V$.

The voting phase is illustrated in Fig. 1.

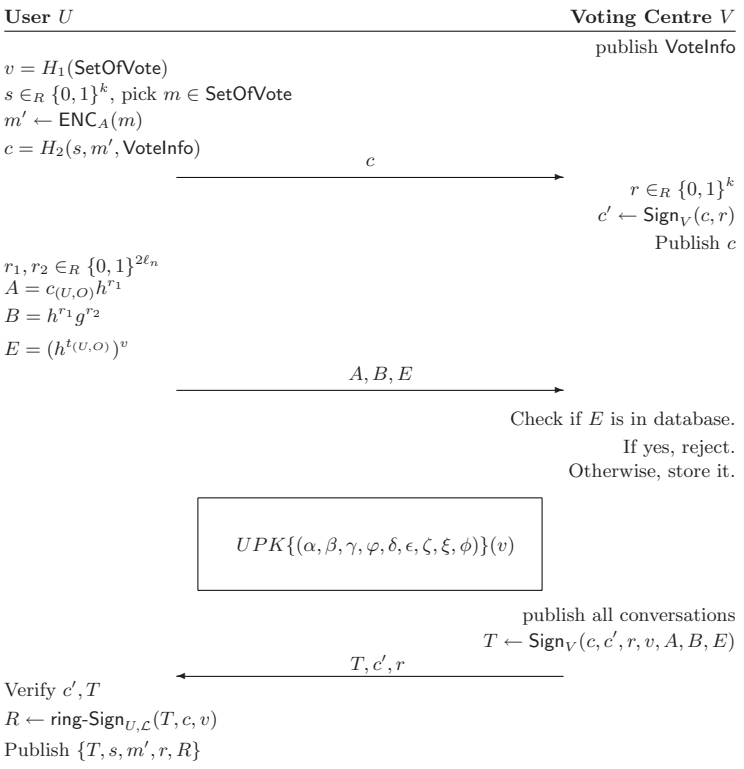



**Fig. 1.** The Voting Phase

### 3.4 Vote-Open Protocol

The vote-open phase is carried out by the arbitrator $A$. $A$ first verifies all the conversations published by $V$, all signatures $T$ and ring signatures $R$ published by the voters. It discards those which are invalid. It then publishes its private key for decryption. For each decrypted message $m$, if $m \notin \mathsf{SetOfVote}$, it discards it. Finally, $A$ counts all the valid $m$ and announces the result.

### 3.5 Removal of the Arbitrator

The above proposed system is a "vote-and-go" system. That is, a voter does not need to participate in any stage after the voting phase. This is realized with the arbitrator such that all the votes are encrypted with the public key of the arbitrator in the voting phase so that no one can get the result before the end of the voting phase. The arbitrator can be removed if the system is not required to be "vote-and-go". In this case, all voters need to participate the vote-open phase. The voting phase also needs to be modified as follows.

1. In step 2, $c$ is computed as $c = H(s, m, \mathsf{VoteInfo})$.
2. In step 8, only $R$, the ring signature tag is published.

In the vote-open phase, a voter posts $\{T, s, m\}$ to the bulletin board anonymous, pairing up with the corresponding ring signature tag $R$. The public, or any party such as the Voting Centre, verifies all the ring signatures and counts the votes. All invalid signatures and all those $m \notin \mathsf{SetOfVote}$ should be discarded.

## 4 An Electronic Questionnaire System

An electronic questionnaire (e-questionnaire) system simulates a questionnaire system in the paper world electronically. In such a system, users are asked to fill in a form which could be in multiple-choice format or fill-in-the-blank format. In many cases, users remain anonymous. However, each user cannot hand in more than one questionnaire in order to bias the result. There are many functional similarities between an e-questionnaire and e-voting system. The main difference is that an e-questionnaire system needs to allow users to input an arbitrary string while an e-voting system only allows a selection from a finite set as input. This makes most of the existing e-voting systems not suitable to be used as e-questionnaire systems.

Our proposed e-voting system can easily be extended to an e-questionnaire system. We just need to change $\mathsf{SetOfVote}$, the message to be chosen from, in Sec. 3.3, to be a set of arbitrary strings representing the overall input of the questionnaire. Note that receipt-freeness is not preserved in our e-questionnaire system since an user can put his signature in his answer to the questionnaire. But in most of the cases, it is not a requirement for a questionnaire system in the real world.

## 5 Concluding Remarks

In this paper, we proposed a restricted version of multi-show credential system. In addition, we propose a new e-voting system which is based on the restricted multi-show credential system and ring signature with a tamper resistance device. Furthermore, we show how to extend our e-voting system to an electronic questionnaire system. There are several open problems left for further research such as constructing a more efficient e-voting system with the above desired properties and without using any tamper-resistance hardware device while providing receipt-freeness.

## References

1. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.
2. J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *Proc. 26th ACM Symp. on Theory of Computing (STOC)*, pages 544–553. ACM, 1994.

3. S. Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS-R9323, CWI, April 1993.
4. S. Brands. Rapid demonstration of linear relations connected by boolean operators. In *Proc. EUROCRYPT 97*, pages 318–333. Springer-Verlag, 1997. LNCS 1233.
5. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocations (full paper). In *Proc. EUROCRYPT 2001*, pages 93–118. Springer-Verlag, 2001. LNCS 2045.
6. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Proc. CRYPTO 97*, pages 410–424. Springer-Verlag, 1997. LNCS 1296.
7. J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. Technical Report TR260, Institute for Theoretical Computer Science, ETH Zurich, March 1997.
8. R. Chan, J. Wong, and A. Chan. Anonymous electronic voting system with non-transferable voting passes. In *SEC 2000*, volume 175 of *IFIP Conference Proceedings*, pages 321–330. Kluwer, 2000.
9. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.
10. D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.
11. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proc. CRYPTO 94*, pages 174–187. Springer-Verlag, 1994. LNCS 839.
12. A. Fiat and A. Shamir. How to prove yourself: Practical solution to identification and signature problems. In *Proc. CRYPTO 86*, pages 186–194. Springer-Verlag, 1987. LNCS 263.
13. A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale election. In *AUSCRYPT91*, pages 244–260. Springer-Verlag, 1992. LNCS 718.
14. E. Fujisaki and T. Okamoto. Witness hiding protocols to confirm modular polynomial relations. In *Proc. the 1997 Symposium on Cryptography and Information Security*, pages SCS197–33D. The Institute of Electronics, Information and Communication Engineers., 1997.
15. M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *Proc. EUROCRYPT 2000*, pages 539–556. Springer-Verlag, 2000. LNCS 1807.
16. B. Lee and K. Kim. Receipt-free electronic voting scheme with a tamper-resistant randomizer. In *Proc. ICISC 2002*, pages 389–406. Springer-Verlag, 2003. LNCS 2587.
17. J. Liu, V. Wei, and D. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. Cryptology ePrint Archive, Report 2004/027, 2004. http://eprint.iacr.org/.
18. J. Liu, V. Wei, and D. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In *ACISP04*, pages 325–335. Springer-Verlag, 2004. Lecture Notes in Computer Science No. 3108.
19. C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Proc. EUROCRYPT 93*, pages 248–259. Springer-Verlag, 1994. LNCS 765.
20. R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Proc. ASIACRYPT 2001*, pages 552–565. Springer-Verlag, 2001. LNCS 2248.
21. V. Shoup and A. Rubin. Session key distribution using smart card. In *Proc. EUROCRYPT 96*, pages 321–331. Springer-Verlag, 1996. LNCS 1070.

# Recard: Using Recommendation Cards Approach for Building Trust in Peer-to-Peer Networks

Hany A. Samuel[1], Yasser H. Dakroury[2] , and Hussein I. Shahein[3]

[1] Computer and Systems Engineering DEPT., Faculty of Engineering,
Ain Shams University, Cairo, Egypt
Phone(+2012)4703541
hany@asueng.eun.eg,
[2] Computer and Systems Engineering DEPT., Faculty of Engineering,
Ain Shams University, Cairo, Egypt
ydakroury@mcit.gov.eg
[3] Computer and Systems Engineering DEPT., Faculty of Engineering,
Ain Shams University, Cairo, Egypt

**Abstract.** The peer-to-peer applications have recently seen an enormous success and spread over the Internet community which showed a dramatic change in the current client-server paradigm; that caused the appearance of some new concepts and protocols. One of the main new concepts introduced is the user anonymity which is in spite of being considered one of the main characteristics of the peer-to-peer paradigm it has introduced a serious security flaw due to the missing of trust between the participants in the system. This paper proposes an approach for peer-to-peer security, where the system participants can establish a trust relationship between each others based on their reputation gained by the participation in the system. The proposed technique relays on the concept of the recommendation cards. This paper discusses this technique and how to apply it to a peer-to-peer file sharing application.

**Keywords:** Security, Peer-to-Peer Networks, Trust Management, Reputation Systems.

## 1 Introduction

The peer-to-peer network is the network in which peers cooperate to perform a critical function in a decentralized manner[20],in peer-to-peer networks all nodes have the same roles and there is no nodes with a special responsibilities to monitor or supervise the network behavior, this scheme is so different from the ordinary client-server paradigm as each node acts as both a server and a client at the same time for example of such scheme is Gnutella[9], Napster [21].

Current peer-to-peer applications can be classified into one of these categories:

- *Instant Messaging:* a category of user applications to exchange different kinds of messaging such as MSN messenger [19], Yahoo Messenger [29].
- *Distributed Processing:* in such systems peers share the computational power of their nodes, as in SETI@Home [27].

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 280 – 292, 2005.

- *File sharing:* is currently the largest field for peer-to-peer where it allows peers to share their files (and/or storage space), there has been many developed applications and researches in that field such as Napster [21], Kazaa[16], Gnutella[9], Freenet[8], [10], [11], Chord[12].

One of the main characteristics of the peer-to-peer network is the user anonymity which caused a serious security problem that is how the peer can trust and authenticates other peers in orders to cooperate with them. Traditional security techniques (such as the presence of trusted certification authority to provide certificates) are not applicable here due to the fact that all peers are equal. This paper introduces a new technique to provide authentication and trust based on reputation in peer-to-peer environment, the contribution of the proposed technique relays on using the recommendation cards which will be discussed in details. Although the proposed technique can be applied to all the previously discussed fields of the peer-to-peer networks; this paper discusses the application of the proposed technique to the file sharing applications.

## 2 Previous Work

Mapping the trust and reputation model of the real societies into the virtual societies has seen extensive research during the last few years, one of the pioneer researches in that field was [25] that although of its complexity it is considered a main mark in that field; another interesting model was in [1] that proposed the general structure for developing trust and reputation in a distributed system; most of the later work, in that area had followed their ideas but in different application domains [5], [13], [17].

Most of the work done in the field of trust and reputation in peer-to-peer networks relay mainly on voting [7],[6],[4], that to get the votes of other peers in order to evaluate the targeted peer or the targeted resource [4], which was found to impose an overhead on both the network (due to the number of voting messages that need to be exchanged for every voting session) and the voting peers themselves as each peer needs to process the voting request by searching his own records then encrypts the result and sends it back to the requester which imposes a computational overhead locally for that peer moreover it makes that peer vulnerable to a denial of service attack by flooding him with faked voting requests. Another point to consider here is the delay encountered in the decision making due to both the delay in communication between peers over the network and the delay in computation at each peer. An important point to notice that the voting is not only performed at the first time when two peers interact but it is also performed regularly to re-evaluate the peer's reputation. One of the voting techniques vital drawbacks is not considering the votes of the peers currently disconnected from the network although their votes may change the decision dramatically; many researches [15],[24],[2] tried to cover this point by storing the complaints about each peer in some node (peer) within the network so even if the complaining peer left the network his vote is preserved when votes are required. But these techniques has also some drawbacks such as being vulnerable to the collusion between the peers who holds the voting information with the peer who will gain profit from modifying these information; also these researches relays on the existence of a special node(s) like THA(trust holding agent), bootstrap server[2] which may be re-

garded as a violation of the peer-to-peer environment concept (that all peers are equal), moreover provides a single point of failure which makes the system vulnerable to an easy denial of service attack by attacking these special peers. Another problem introduced in these solutions when a malicious peer is assigned a special role in the network like being the bootstrap server [2]. The proposed technique tries to improve the previous techniques in order to minimize the overhead discussed.

## 3 Sketch of the Approach

Before discussing the new technique the goals of the new proposed solution should be formulated precisely; it can be stated as follows:

1. No violation of the anonymity property of the participants peers.
2. Minimizing overhead (encountered in normal voting techniques) for honest peers to prove their honesty.
3. Minimizing the network overhead and the computational overhead for the peers that are not involved in the interactions.

The proposed solution is based on some basic assumption that should be cleared:

- The assumed environment is a peer-to-peer file sharing network.
- No obligations on the files being shared or their distribution.
- The network is dynamic (i.e. that new nodes (peers) arrives at anytime and any node is permitted to leave the network or fail without any warning at any time).
- No peer is assumed to have a global view of the network.
- Each peer is able to generate a public-private key pair for his communication.
- Each peer has a unique ID that is used to identify him; this ID is a secure hashing [22] of his public key; this ID is synonymous for the peer and is not related to his real ID in any way so the anonymity property for the peers is still preserved.
- Each resource (file) has a digest which is a secure hashing [22] of its name and contents, so the downloaded file can be verified for consistency.
- No separate reputation and trust for resources as in [4] due to the huge expected number of shared resources; the resources trust is gained implicitly from the trust of their holder.

The proposed idea is a modification to the simple voting system by the adoption of the recommendation cards technique; the peer, after interacting with another peer, reports his evaluation of that peer in a card and sign it with his private key (so it can be authenticated) then send this card back to the targeted peer, later when a peer wants to interact with another peer he sends the recommendation cards gained from previous interactions to that peer so an evaluation for him can be made without any of the other peers being contacted for voting ( like the normal voting system).

As opposed to some of the previous techniques where the trust and reputation context were ambiguous (that it was not determined precisely the context of the evaluation), we precisely evaluates the peer from three different points of views which can be defined as follows:

1. *Honesty:* is a measure for the peer behavior that reflects the safety degree of the contents that can be obtained through that peer; a malicious peer can send a malicious contents that may contain a virus, Trojan horse, .. etc. that peer is considered dishonest.
2. *Reliability:* is a measure for the quality of service this peer provides, that it reflects the file quality for the shared files beside the downloading capabilities of that peer (i.e. the connection speed and if disconnection happening).
3. *Sharing:* is a measure for the volume of files shared by that peer.

**Table 1.** Notation used

| | |
|---|---|
| THi,j | The honesty trust value, reflects the peer i evaluation for peer j honesty, it ranges from 0 to 1 |
| TRi,j | The reliability trust value, reflects the peer i evaluation for peer j reliability, it ranges from 0 to 1 |
| TSi,j | The sharing trust value, measured in Bytes shared by peer j for peer i, it ranges from 0 to 1 |
| CHi, CRi | The honesty credibility factor and the reliability credibility factor of peer i opinion respectively, it ranges from 0 to 1 |
| αH , αR | The honesty learning factor and the reliability learning factor respectively, it ranges from 0 to 1 |
| αCH, αCR | The credibility learning factor for honesty and reliability respectively, it ranges from 0 to 1 |
| WHi , WRi | The honesty worthiness and reliability worthiness respectively. It ranges from 0 to 1 |
| WSi | The sharing worthiness measured in Bytes shared by i. |
| DT | Expire date for the card. |
| WT | Trust weight, it ranges from 0 to 1 |
| KUi, KRi | Public key and private key for peer i respectively. |
| Ex(M) | Encryption of message M with key x |
| H(M) | Hashing for message M |
| θH, θR | Honesty trust threshold and reliability trust threshold respectively. |
| M\|X | Concatenation of M and X |

Note that the proposed technique is flexible to be used with any other evaluation criteria.

Fig. 1 shows the format of the recommendation card (review Table1 for Notation used), a brief explanation of some fields is considered next:

- *Expire Date:* is important value reflects when this card should be obsolete; the importance of this field will be clear when discussing the difficulty of revoking an issued card.
- *Trust weight:* reflects how the personal peer experience should affect the evaluation of other peers' trustworthiness.

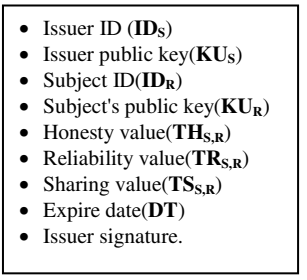



**Fig. 1.** Recommendation card format

- *Issuer signature:* EKRissuer( H ( $ID_S$ | $KU_S$ | $ID_R$ | $KU_R$ | $TH_{S,R}$ | $TR_{S,R}$ | $TS_{S,R}$ | DT) ), is the issuer digital signature for the card to ensure its validity.
- *Credibility factor:* The importance of the credibility factor rises also from the difficulty of making all the peers evaluate with the same criteria so the credibility factor plays as a weighting factor for the peers evaluation.

## 4 Recard Details

The peer that requests a file or more generally a service will be referred to as a *requester* while the peer who provides this service will be refereed to as *provider*, this notation is used only for simplifying the next discussion but is not general within the peer-to-peer context as each peer operates as both provider and requester at the same time.

When a requester searches for a resource and gets different responses from other providers, he will need to choose a trusted provider to get the resource, accordingly the provider will need to decide if that requester does worth to be given that resource or not (depending on each provider criteria).

First the requester need to evaluate each provider to choose a trusted one, so he sends a request for each provider to send his owned cards. Fig. 2 shows the scenario for the evaluation process but due to the limited space only the main points are discussed. The requester calculates the provider's worthiness from the discussed three points of views as follows:

For provider i
Remove all non valid cards.
IF ($TH_i \geq \theta_H$ AND $TR_i \geq \theta_R$) THEN

$$WH_i = W_T \times THi + (1-W_T) \times \frac{\sum_K CH_K \times TH_{K,i}}{\sum_K CH_K} \quad (1)$$

$$WR_i = W_T \times TR_i + (1-W_T) \times \frac{\sum_K CR_K \times TR_{K,i}}{\sum_K CR_K} \quad (2)$$

$$WS_i = W_T \times TS_i + (1-W_T) \times \sum_K TS_{K,i} \quad (3)$$

ELSE Neglect that provider

(Where k is the number of recommenders for that provider )

***Notice:*** the requester will not deal with the provider if his previous trust value (i.e. experienced by the requester himself) is less than the threshold value regardless of that provider's reputation (i.e. what the others tell about him).

Depending on the previous calculations a suitable provider (according to his worthiness) is selected. the selected provider will evaluate the requester which was not implemented in the previous techniques because the requester can not cheat while downloading but always the provider can cheat by giving malicious contents, but another point of view for that situation is the worthiness of the requester to access which resource according to his worthiness (i.e. that to identify his access privileges for the shared resources which will differ according to each provider criteria).

So the provider follows the same steps to evaluate the requester using calculations in (1),(2) and (3), and according to his criteria he accepts or rejects the request; in case of acceptance the session begins as shown in Fig.2 after both sides authenticate each other as follows: The requester generates a random nonce N1 and sends $EKU_{PROVIDER}(N1)$ to the provider. Then The provider generates a random nonce N2 and replies to the requester with $EKU_{REQUESTERr}(N2|N1)$, finally The requester replies with $EKU_{PROVIDER}(N2)$. Due to the random nature of N1 and N2 so the malicious peer will find the reply attack is so hard to be applied.

After the requester gets the file, he evaluates the provider then updates his records as follows:

First updating the provider record (similar to [28]).

For provider i

$$TH_i^{new} = \alpha_H \times TH_i^{old} + (1 - \alpha_H)\, e_H \quad (4)$$

$$TR_i^{new} = \alpha_R \times TR_i^{old} + (1 - \alpha_R)\, e_R \quad (5)$$

$$TS_i^{new} = TS_i^{old} + F \quad (6)$$

***Notice:*** F is the shared file size in bytes, $e_R$ and $e_H$ are the evidences for reliability and honesty respectively with value of 1 in case of successful interaction or -1 for unsuccessful interaction (according to each requester's evaluation).

Second updating the credibility of each recommender:

For each peer i in the recommenders list of that transaction

$$CH_i^{new} = \alpha_{CH} \times CH_i^{Old} + (1 - \alpha_{CH})\, e_H \quad (7)$$

$$CR_i^{new} = \alpha_{CR} \times CR_i^{Old} + (1 - \alpha_{CR})\, e_R \quad (8)$$

***Notice:*** there is no credibility for sharing; because this information can not be verified after the interaction. The requester finally constructs and sends the updated (or new) card. Notice that the provider will not update his records after the exchange simply because as a provider he can not make an evaluation for the requester.

Because any peer can generate a public-private key pair and make a faked cards that belongs to none existing peer, so only the cards from known trusted peers are accepted after being checked for consistency by checking its hashing against the is-

suer signature and checking its expire date. If the number of the accepted cards is below a threshold value a voting session is held but only the known trusted peers with high credibility are involved in it; in case that the peer is still not known then he will be assigned a default trust value and it became his responsibility is to improve it.

## 5 Attacks and Defenses

- *Denial of service:* the proposed technique helps in limiting this attack by ignoring the requests of the entrusted peers; besides including the suspected behavior (such as multiple search requests within small period of time) in updating the peer trust.
- *Free riding [3]:* that some peers does not share any useful files, it can be solved by evaluating the peer sharing worthiness (WS) and depending on this value free riders can be discovered with a good precision then a suitable action (like being given a limited access) can be taken.

The proposed technique can also be used to defend against known attacks on the reputation systems like:

- *Self replications*: a malicious peer can provide faked resources that have a good reputation (as they are also offered by trusted peers); this can easily be discovered as the faked resource digest will differ from the real resource digest (known from the trusted peers).
- *Masquerade:* the malicious peer can capture the cards of a trusted ID (as they being sent) and then pretends to be him; this can be defended easily during the authentication step.
- *Sybil attack [14]:* the malicious peer creates and control multiple peers' identities to give him a good reputation, in the proposed technique these peers' cards will be unknown so it will be discarded. Although these peers may have been interactive in the system to have a good reputation but in such case the attack's cost (to create and operate such peers) will make it impractical.
- *Corrupting peers reputation*: that malicious peers try to corrupts other honest peers reputation but this attack is easily defended within Recard because of two reasons:
    a. The honest peer can discard the cards given by the malicious peers.
    b. Even if a voting session is held, no one will give the malicious peer's opinion an effective credibility factor.

## 6 Recard Versus the Previous Systems

Most of voting systems[7],[6],[4] uses the public key encryption extensively, besides the generation of many public-private key pairs[15] which imposes a computation overhead on peers; Recard uses one public-private key pair for each peer and performs less encryption and decryption computations than other techniques.

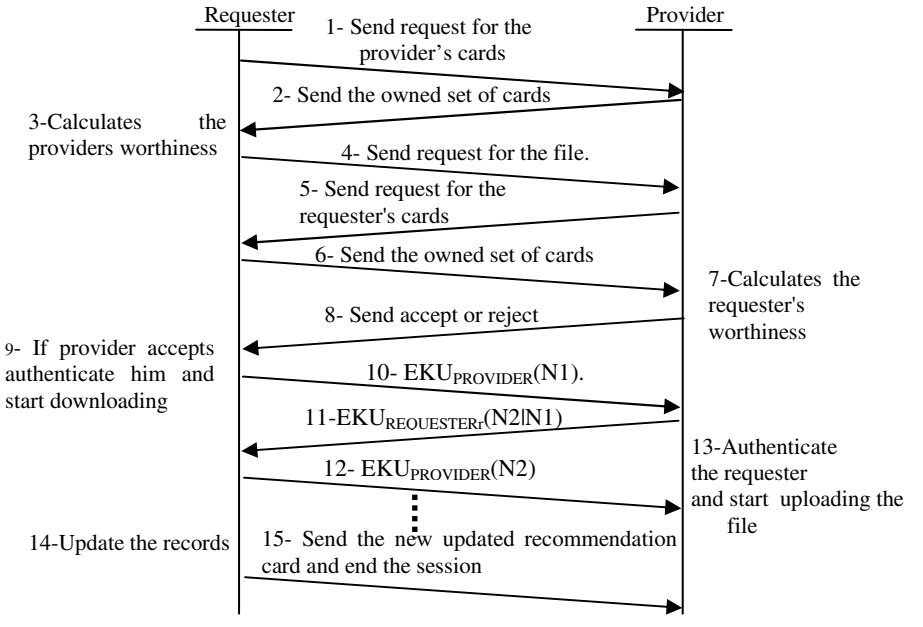



**Fig. 2.** The evaluation process handshaking

With Recard an honest peer will find no problem to prove his honesty even with all the peers he interacted with are currently out of the network. Some systems like [15], [24], [2] are storing complaints at some specific peers at the network (which can also be regarded as a violation of the concept that all the peers are equal) which makes these algorithms weak in face of the collusion of these peers with a malicious peer; moreover if these peers were down or out of the network all the complaints they store will be inaccessible (some algorithms assume the peer should transfer all his stored information to another peer before he leaves the network which open another source for overhead and inconsistency as he may just fail).In systems like [15],[24],[2] a malicious peer may try to corrupt another peer's reputation by publishing a wrong complains about that peer, this is implicitly handled in Recard that if the peer issued a bad evaluation card it can be simply discarded and not considered in next transactions .also the votes of these misbehaved peers are not considered in voting simply because they will have a bad credibility.

The persistence of Recard against the Sybil attack[14] with less overhead than before must be counted as an advantage, where previous techniques [15] used to cluster the IP addresses to find the controlled peers and recheck some votes to be sure that these votes are from real existing peers; Recard doesn't need to apply any of these techniques as only the cards from previously known trusted peers are counted; even in case of starting a voting session only the known trusted peers votes are requested (not by broadcasting). Also the inclusion of the credibility factor in the calculations of the trustworthiness permits each peer to evaluate the other peers with his own criteria which are totally independent from any other peer criteria; that solved the problem of unifying the evaluation criteria among the peers.

On the other hand one of the main drawbacks of Recard is the ability of the malicious nodes to discard the recommendations cards that contains bad evaluations but this drawback can be tolerated because if the number of valid cards received is under some threshold value a voting session is held, but with only the known peers that have good credibility factor are involved in the voting, not by broadcasting like previous techniques [7],[6],[4] which decrease the network overhead.

The revocation of the issued cards is a very hard process as the issuer peer does not know who will receive that card, even with broadcasting no guarantee that all the peers will know about the revocation of that card; the expire date appended for each card is intended to be a partial solution for that problem, this expire date is assigned according to the subject peer's trust value that the larger the trust value of that peer the longer the validity period of his card.

During the start of the system there will be no trust relationships yet, so the system will take a time to adapt (i.e. trust relationships being constructed between different peers), but this drawback is common to all the previous reputation systems in peer-to-peer networks. some other drawbacks that Recard shares with all previous reputation systems is the space required to save the trust information for each peer and the ability of the malicious peer to change his ID and start as a new user with a new ID but of course he will be rediscovered again after performing some malicious transactions.

## 7 Results and Future Work

We will assess the performance of our scheme as compared to a peer-to-peer network where no reputation system is implemented, also where a voting system is implemented.

Simulation assumptions are based mainly on the assumptions and the measurements in [18], [26] but we briefly discuss our Model and performance measure criteria.

When a query is issued by a peer, it is propagated by broadcast with hop-count horizon throughout the network (in the usual Gnutella way); peers which receive the query forward it and check if they are able to respond to it. The network consists of honest peers and malicious peers. It is assumed that the malicious peer will behave honestly most of the time (according to an honesty degree, assigned to these peers). Files are assigned probabilistically to peers at initialization, the simulation of a network proceeds in simulation cycles [18]. We have experimented with networks of different sizes and topologies and our conclusions continue to hold.

As performance measures, we are particularly interested in

1. The number of successful interaction (honest and reliable) as a percentage of the total number of interaction.
2. The number of voting sessions.

***Experiment 1:*** The network were composed of 100 peers with 30% are malicious that are assumed to get ride of the recommendation cards that state that they are malicious and keep the good cards. We used for this experiment: $\alpha CH = \alpha CR = \alpha CH = \alpha CR = 0.9$, the simulation runs until 1000 interactions are done. The simulation is repeated 10 times (for the same network configuration) and we use the mean of the results, Fig 3, Fig 4 shows how the performance was improved (measured as the percentage of

successful interactions) over the regular system that does not use a reputation system, as noted Recard performance were equal to the regular voting systems performance with respect to the percentage of the successful interactions. but as shown in Fig 5 Recard performance were much better than the normal voting system when regarded as the number of voting sessions held, as noted Recard was behaving as the regular voting sessions until the system adapted (that the peers interacted and exchanged cards so they were able to trust each others).

***Experiment 2:*** we use the same configuration as the previous experiment but with the number of peers is 50, and the simulations continues until the number of interaction reached 10000; we got results comparable to the previous experiment but the main point to note as shown in Fig 6 that the voting systems performance (measured as the number of voting session) were very bad compared with Recard which almost makes no new voting sessions as the systems adapted but the number of voting session for the regular voting system still increase nearly linearly with new interactions.

Currently different simulation experiments are applied to test the behavior of the proposed protocol regarding the previous voting protocols in the presence of different attacks. After finishing the simulation a prototype implantation of the protocol will be developed and distributed as shareware for being tested in a real network environment like the Internet. Another important research proposal is the adaption of the proposed technique into existing peer-to-peer file sharing systems such as Gnutella [9], Free Heaven [23], Freenet [8],[10],[11].

## 8 Conclusions

As have been shown Recard achieved the goals discussed before. It also introduced a new idea that peer-to-peer authentication and trust should be the problem of the two involved peers that minimize the overhead on the other peers and also on the network. It also takes into consideration the access privileges assignment based on the peer's trustworthiness and improves the performance for honest peers.

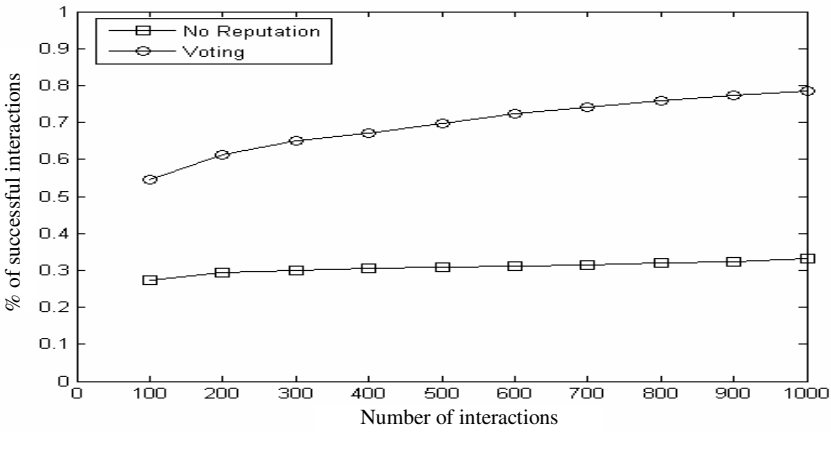



**Fig. 3.** Voting system versus No-reputation systems

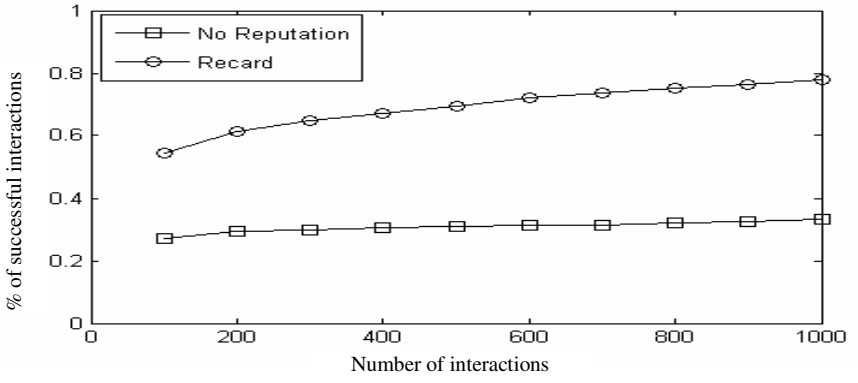



**Fig. 4.** Recard versus No-reputation systems

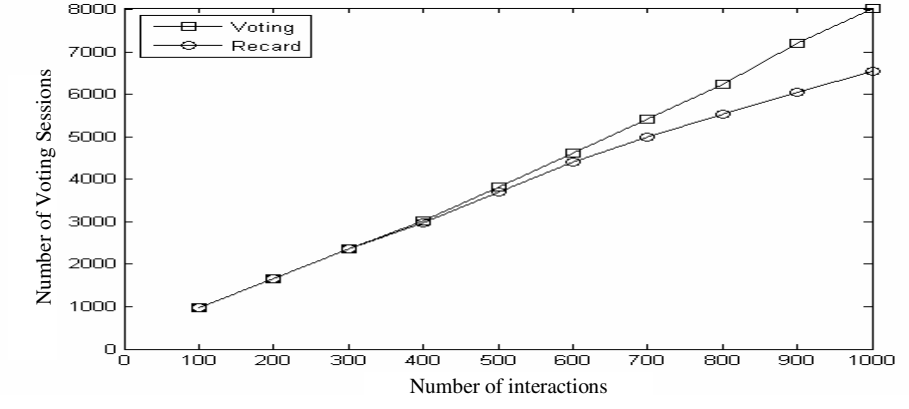



**Fig. 5.** Recard versus regular voting systems

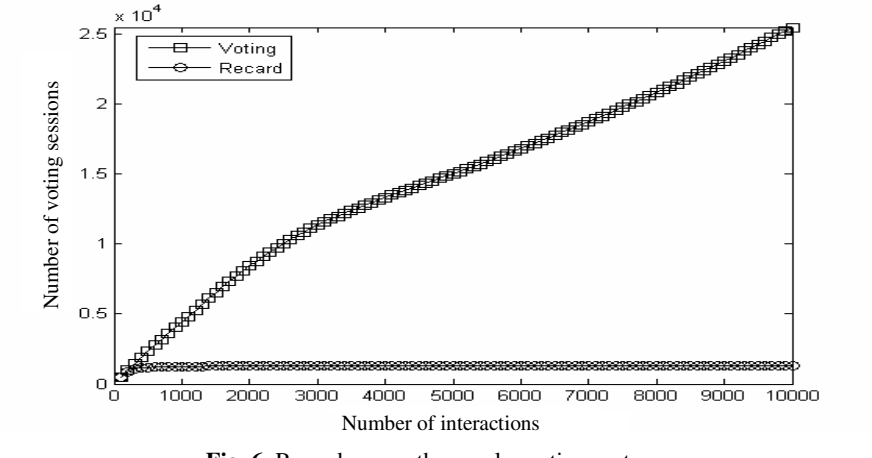



**Fig. 6.** Recard versus the regular voting systems

## References


[1] Abdul-Rahman and S. Hailes," Supporting trust in virtual communities", In proceedings of the Hawaii International Conference on System Sciences, Maui, Hawaii, Jan 4-7 2000.

[2] Aameek Singh and Ling Liu." TrustMe: Anonymous management of Trust Relationships in Decentralized p2p systems", In Proc. Of IEEE Third International Conference on Peer-to-Peer Computing (P2P'03)September 01 - 03, 2003 Linköping, Sweden.

[3] E.Adar and B. Huberman., " Free riding on gnutella ", Technical report, Xerox PARC, August 2000.

[4] E.Damiani, D.C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. "A Reputation-Based Approach for choosing Reliable Resources in Peer-to-peer networks". In Proceedings of the 9th ACM conference on computer and communication security, pages 207-216. ACM Press, 2002.

[5] F. Azzedin and M. Maheswaran," Evolving and Managing Trust in Grid Computing Systems", IEEE Canadian Conference on Electrical and Computer Engineering (CCECE '02), May 2002.

[6] F. Cornelli and E. Damiani and A. Ghorbani "Implementing a Reputation Aware Gnutella Servent". In Proceedings of the International Workshop on peer-to-peer Computing Pisa, Italy, May 24, 2002.

[7] F.Cornelli, E Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati. "Choosing reputable servents in a P2P network". In Proc. Of the Eleventh International World Wide Web Conference, Honolulu, Hawaii, May 2002.

[8] Freenet Homepage, http://freenet.sourceforge.net

[9] Gnutella Homepage, http://gnutella.wego.com

[10] Clarke "A decentralized information storage and retrieval system", Master's thesis, university of Edinburgh, 1999.

[11] Clarke, O. Sandberg, B. Wiley and T. W. Hong " Freenet: A distributed anonymous information storage and retrieval system", In proceedings of the ICSI workshop on design issues in anonymity and Unobservability (Berkeley, California, June 2000).

[12] Stoica, R. Morries, D. Karger, M. F. Kaasshoek and H. Balakrishnan. "Chord: A scalable peer-to-peer lookup service for internet applications." In Proc. of ACM SIGCOMM 2001, August 2001.

[13] J. Carter, E. Bitting and A. Ghorbani "Reputation Formalization for an information-sharing Multi-peer System", Computational Intelligence, Volume 18, Number 4, November 2002, pp 515-534.

[14] J.Douceur." The sybil attack"In First IPTPS, Cambridge, MA(USA),March 2002

[15] K. Aberer and Z. Desporovic. "Managing trust in a peer-to-peer information system". In Proc. Of the Hawaii International Conference on System Sciences, Maui, Hawaii, January 2000.

[16] Kazaa homepage, http://www.kazaa.com

[17] M. Montaner and B. Lopez "Opinion based filtering through trust". In proceedings of the 6th International workshop on Cooperative Information peers (CIA' 02), Madrid (Spain), September 18-20 2002.

[18] Mario T. Schlosser, Tyson E. Condie, Sepandar D. Kamvar.," Simulating A File-Sharing P2P Network", 1st Workshop on Semantics in Grid and P2P Networks, 20 May 2003, Budapest, Hungary.

[19] Microsoft Network Messenger Homepage, http://.messenger.msn.com

[20] MiloJicic D.S., Kalogeraki V. and Luckose R."peer-to-peer Computing", Tech Report:HPL-2002-57, available online at: http:\\www.hpl.hp.com/techreports/2002/HPL-2002-57. pdf
[21] Napster Homepage, http://www.napster.com
[22] P.C. Van Oorschot, A.J. Menezes and S.A. Vanstone. "Handbook of Applied Cryptography", CRC Press, 1996.
[23] Roger Dingledine, David Molnar and Michael J.Freedman." the Free Haven project: Distributed anonymous storage service", In proceedings of the workshop on Design Issues in Anonymity and Unobservability, July 2000.
[24] S. Kamvar, M. Schlosser, and H. Garcia-Molina. "Eigenrep: Reputation management in p2p networks". In Twelfth International World Wide Web conference, 2003.
[25] S. Marsh. "Formalizing trust as a computational concept", PhD. Thesis, university of Stirling, 1994
[26] S. Saroiu, P. K. Gummadi, and S. D. Gribble," A measurement study of peer-to-peer file sharing systems", In Proceedings of Multimedia Computing and Networking 2002(MMCN'02),San Jose, CA,USA, January 2002.
[27] SETI@HOME: THE Search for Extraterrestrial Intelligence at Home,http://setiathome.berkely.edu/
[28] Y. Wang., J. Vassileva. "Bayesian Network Trust Model in Peer-to-Peer Networks". In proceedings of second international workshop on peers and peer-to-peer computing, July 14,2003.Melbourne, Australia
[29] Yahoo Messenger Homepage, http://www.yahoo.com

# Using Trust for Restricted Delegation in Grid Environments

Wenbao Jiang[1], Chen Li[1], Shuang Hao[2], and Yiqi Dai[2]

[1] Department of Information System,
Beijing Information Technology Institute, No35 BeiSiHuanZhongLu,
Beijing, 100101, China
jiangwenbao@tsinghua.org.cn, lichen@biti.edu.cn
[2]Department of Computer Science and Technology,
Tsinghua University, 100084, China
haoshuang98@mails.tsinghua.edu.cn,
dyq@theory.cs.tsinghua.edu.cn

**Abstract.** Delegation is an important tool for authorization in large distributed environments. However, current delegation mechanisms used in emerging Grids have problems to allow for flexible and secure delegation. This paper presents a framework to realize restricted delegation using a specific attribute certificate with trust value in grid environments. The framework employs attribute certificates to convey rights separately from identity certificates used for authentication, and enables chained delegations by using attribute certificate chains. In the framework the verifier can enforce securely authorization with delegation by checking the trust values of AC chains, and judge if a delegation is a trusted delegation by evaluating the reputation value of the delegation chain. The paper discusses the way of computing trust and reputation for delegation, and describes some details of delegation, including the creation of delegation credential and the chained delegation protocol.

## 1 Introduction

Delegation is an essential tool of cooperation in distributed systems, especially in Grids that have emerged as dynamic, inter-domain, distributed computing environments [1]. Within Grids, A user must be able to delegate a service the ability to run on that user's behalf, so that the service is able to access the resources on which the user is authorized. For small ad-hoc collaborations with often only temporary existence, it is required that an entity can delegate a subset of its rights to another entity and the receiving entity can combine these rights with other delegated or own rights, so that the entities can share data, program and computational resources without the need for administrator intervention.

Delegation of rights always carries with it a risk of misuse; therefore, it is important to realize *restricted delegation*, which can minimize exposure by delegating the precise set of rights necessary for the task. Unfortunately, The conventional approach when a user must ask a service to perform some operation on her behalf is to grant *unlimited delegation*, which is to unconditionally grant the service the ability

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 293 – 301, 2005.

to impersonate the user. For delegations within a Grid, the crucial issue is the determination of those rights that should be granted by the user to the service and the circumstances under which those rights are valid. Delegating too many rights could lead to abuse, while delegating too few rights could prevent task completion [2].

The Grid Security Infrastructure (GSI) [3] is a security mechanism of the Globus Toolkit, which is widely used by Grid efforts worldwide. GSI realize delegation using *proxy certificates,* which may be used like standard X.509 identity certificates for authentication. GSI originally supported only unlimited delegation. The Community Authorization Service (CAS) [4] extends GSI delegation mechanisms by using *restricted GSI proxy certificates* [5] that allow for fine-grained control of delegated rights. The delegation mechanism using GSI proxy certificates supports impersonation, which allow entity A to grant to another entity B the right for B to authenticate with others as if it were A. This impersonation scheme is easy to integrate with many existing identity-based authorization systems. However, the impersonation scheme bears the danger of violating the "least privilege principle" [6], as it is often problematic to clearly define the minimum subset of privileges needed by the proxy. Moreover, the delegation approach using impersonation is unsuitable for some strong authorization mechanism, such as attribute-based authorization systems based on the use of Privilege Management Infrastructure (PMI) [7].

This paper focuses on the approach to realize restricted delegation based on attribute certificates with trust values, which allows for very flexible and secure delegation. The remainder of this paper is organized as follows. Section 2 introduces some definitions used. Section 3 gives a broad overview of our approach. Section 4 discusses the way of computing trust and reputation. Some details of delegation, including the creation of delegation credential and the chained delegation protocol, are described in Section 5. Finally, the conclusions are drawn in Section 6.

## 2 Definitions

A principal is a participant in a security operation; it is generally a user, a process operating on behalf of a user, a resource, or a process acting on behalf of a resource.

Delegation is the process whereby one principal grants the ability to act on its behalf to another principal. We focus here on the Delegation of rights, which assumes that a principal A has herself a set of rights, and it delegates all or a subset of them, to another principal B who can then act, instead of A, to exercise that particular set of rights. In a delegation we classify the participating principals as follows:

- The initiator, who is the originator of the delegation
- The grantor, also called the delegating principal, who delegates its rights to another principal
- The grantee, also called the delegated principal, who receives the delegation made by the grantor
- The verifier, also called "end point" or "end server", who enforces the authorization.
- The intermediary, who is a principal between the initiator and the verifier in the delegation.

Trust is an ambiguous concept that defies exact definition. This has given rise to an evident lack of coherence among researchers in the definition of trust [10,11]. For our purposes, however, we use the following definition by [10]:

*Trust is the firm belief in the competence of an entity to act as expected such that this firm belief is not a fixed value associated with the entity but rather it is subject to the entity's behavior and applies only within a specific context at a given time.*

We use trust value (TV) as a trust metric, which is a dynamic value and spans over a set of values ranging from *fully trustworthy* to *fully untrustworthy*. In the paper we adopt a percentage as a trust metric, hence TV is a value between 0 and 1, 1 denotes fully *trustworthy,* 0 denotes *fully untrustworthy.*

When evaluating the trust value of an entity, we can rely on the reputation of the entity. The definition of reputation that we will use in this paper as follows:

*The reputation of an entity is an expectation of its behavior based on other entities' observations or the collective information about the entity's past behavior within a specific context at a given time.*

## 3 Overview

Our approach, illustrated in Figure 1, uses ACs as delegation tokens, and adopts AC chains to implement chained delegation, which is similar to the delegation model in X.509 PMI [7]. Rights are securely assigned and delegated to entities by embedding the rights in attribute certificates. The grantor of the right will sign the AC to the grantee. Every AC serving as delegation token includes a trust value (TV), which denotes trust degree that the grantor assigns to the delegation. Given the PKCs of the grantor and the intermediaries in the delegation path, a verifier (resource) can check by the AC chain if the grantor and the intermediaries are authoritative and determine whether the delegation is valid.

To illustrate the scenarios of delegation in Grid environments, Figure 1 depicts a virtual organization (VO) containing three domains. Each domain has a set of entities, including users, services, resources, and so on. Hence, we have introduced an implicit hierarchy based on entities, domains, and VO.

The SOA (Source of Authority) is the root of trust within a domain, and serves as the grantor and initiator in a delegation. For a chained delegation the SOA is the initial issuer of ACs that assigns privileges to privilege holders. It authorizes the privilege holder to act as a grantor, which further delegates that privilege to other entities through the issuance of ACs that contain the same privilege (or a subset thereof). The SOA may impose constraints on the delegation that can be done. A universal restriction on delegation is that no grantor can delegate more privilege than it holds. A grantor may also further restrict the ability of downstream grantors.

Each reputation service (RS) provided by the SOA is responsible for calculating the reputation values of other domains, and maintaining a dynamic reputation metric for entities within its domain. The verifier can enforce the authorization securely by check the reliability of chained delegation using AC chain's TV. The TV is built on reputation and direct trust relation between entities. The direct trust relation is computed based on DTT (direct trust table) maintained by each entity, and the

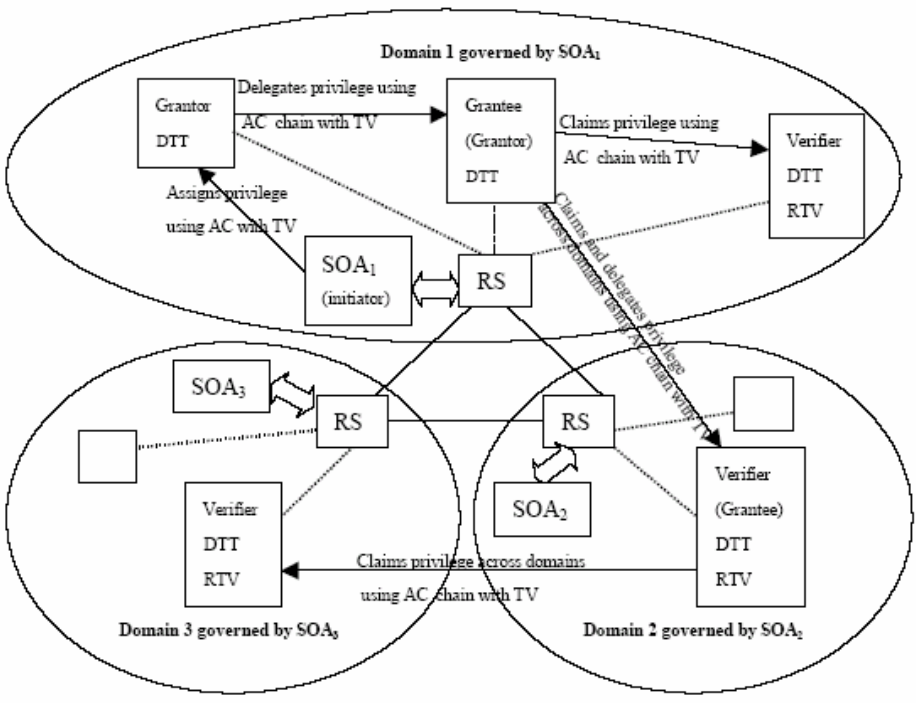



**Fig. 1.** The restricted delegation framework using trust in Grids

reputation is obtained from RS provided by SOA. When making trust-based decision to an access request, the verifier specifies an RTV (required trust value), and reject to access if the TV of delegation chain is smaller than the RTV. Similarly, the verifier may specify an RRV (required reputation value), which can be used to judge if a delegation is a trusted delegation by evaluating the reputation value of the delegation chain.

In order to realize trusted delegation, we employ a specific AC with TV for delegation, which has a different certificate structure from standard AC as defined in X.509 PMI. The main fields of our AC include:

- Issuer: the information identifying the issuer of AC, including the issuer and serial number of the issuer's PKC;
- Holder: the information identifying the holder of AC, including the issuer and serial number of the holder's PKC;
- Attribute: sets of rights (group membership, role, security clearance, or other authorization information) associated with the AC holder;
- ValidityPeriod: time periods when delegation is permissible;
- MaxPathLength: the maximum length of subsequent ACs chain in the delegation path;
- TrustValue: the trust degree that the grantor assign to the delegation
- SerialNumber: An integer value that uniquely identifies the AC within the scope of its issuer.

# 4 Computing Trust and Reputation

Azzedin et al. [10] have proposed that trust relationships in grid environments are based on a weighted combination of the direct relationship between domains as well as on the global reputation of the domains. We use the following notations as introduced in [10] to compute and envaluate trust for delegation:

- Let $D_i$ and $D_j$ denote two domains.
- Let $\Gamma(D_i, D_j, t)$ denote a trust relationship for delegation at a given time t of $D_i$ towards $D_j$ .
- Let $\Theta(D_i, D_j, t)$ denote a direct relationship for delegation at time t of $D_i$ towards $D_j$ .
- Let $\Omega(D_j, t)$ denote the reputation of $D_j$ for delegation at time t.
- Let $DTT(D_i, D_j)$ denote a direct trust table entry of $D_i$ for $D_j$. It is a table that records the trust value from the last transaction between $D_i$ and $D_j$ .

## 4.1 Computing and Evaluating Reputation

### 4.1.1 Computing the Reputation Value of Domains

The reputation value of domain $D_j$ is computed as

$$\Omega(D_j, t) = \frac{\sum_{k=1}^{n} DTT(D_k, D_j) \times R(D_k, D_j)}{\sum_{k=1}^{n} (D_k)} \quad (1)$$

where $k \neq j$, $R(D_k,D_j)$ is the recommender's trust level.Since reputation is primarily based on what domains say about another domain,the recommender's trust factor $R(D_k,D_j)$ is introduced to prevent cheating through collusions among a group of domains. Hence, $R(D_k,D_j)$ is a value between 0 and 1 and will have a higher value if $D_k$ and $D_j$ are unknown or have no prior relationship among each other and a lower value if $D_k$ and $D_j$ are allies or business partners.

### 4.1.2 Computing the Reputation Value of Entities

Similarly, the reputation value of entities $E_j$ within the domain is computed as:

$$\Omega(E_j, t) = \frac{\sum_{k=1}^{n} DTT(E_k, E_j) \times R(E_k, E_j)}{\sum_{k=1}^{n} (E_k)} \quad (2)$$

where $k \neq j$, $E_K$ and $E_j$ denote two entities within the domain. The meaning of other notations is similar to those in Formula (1).

### 4.1.3 Evaluating the Reputation Value of Delegation Chains

*1) Delegation chains within a domain*

Suppose the delegation path of a delegation chain is $A \rightarrow E_1 \rightarrow E_2 \rightarrow \ldots \rightarrow E_n \rightarrow B$, where A is the initiator, B is the verifier, and the others are the intermediaries within a domain. Then, the reputation value of the delegation chain is computed as:

$$RV\ (DC\ ,t) = \Omega\ (A,\ t) \times \prod_{j=1}^{n} \Omega\ (E_j, t) \tag{3}$$

*2) Delegation chains across domains*

Suppose the delegation path of a delegation chain is $A \rightarrow D_1 \rightarrow D_2 \rightarrow \ldots \rightarrow D_n \rightarrow B$, where A is the initiator, B is the verifier, and the others are the intermediaries across n different domains. Then, the reputation value of the delegation chain is computed as:

$$RV(DC,t) \ = \Omega\ (A,\ t)\ \times \prod_{j=1}^{n} \Omega\ (D_j,\ t) \tag{4}$$

### 4.2 Computing and Evaluating Trust

#### 4.2.1 Computing Trust Value

The trust value (TV) of an AC denotes a trust relationship for the delegation of the grantor ($E_i$ ) towards the grantee ($E_j$). Hence, the TV is computed as:

$$TV = \Gamma(E_i, E_j, t) = \alpha \text{ x } \Theta(E_i, E_j\ , t) + \beta \text{ x } \Omega(E_j\ , t) \tag{5}$$

Where $\alpha, \beta \geq 0$, $\alpha + \beta = 1$. $\Theta(E_i, E_j\ , t)$ denotes a direct relationship for delegation at time t of $E_i$ towards $E_j$, hence it can be computed as:

$$\Theta(E_i, E_j\ , t) = DTT(E_i, E_j) \tag{6}$$

$\Omega(E_j, t)$ denotes the reputation of $E_j$ for delegation at time t. It can be computed as formula (2) if $E_i$ and $E_j$ are within a domain. We take the reputation of its domain ($D_j$) as the reputation of entity ($E_j$) if $E_i$ and $E_j$ are across different domains, in this way $\Omega(E_j\ , t)$ can be compute as formula (1).

#### 4.2.2 Evaluating the Trust Value of Delegation Chains

Suppose the certificate chain of a delegation chain is $AC_1 \rightarrow AC_2 \rightarrow \ldots \rightarrow AC_n$, the trust values of certificates are as follows: $TV_1$ for $AC_1$, $TV_2$ for $AC_2$ , …, $TV_n$ for $AC_n$. Then, the overall trust value of the delegation chain, expressed as TV(DC), is computed as :

$$TV\ (DC\ ) = \prod_{j=1}^{n} TV_j \tag{7}$$

## 5 Details of Delegation

As noted above, an AC serving as an delegation credential may defined as a signed 7-tuple:

$$DT_{xy} = \langle X, Y, P_{xy}, T_{xy}, L_{xy}, TV_{xy}, N_{xy} \rangle_X$$

Where X denotes the AC's "Issuer", Y denotes the AC's "Holder", $P_{xy}$ denotes the AC's "Attribute", $T_{xy}$ denotes the AC's "ValidityPeriod", $L_{xy}$ denotes the AC's

"MaxPathLength", $TV_{xy}$ denotes the AC's "TrustValue", $N_{xy}$ denotes the AC's "SerialNumber". $<M>_X$ denotes "M signed by X's private key".

Suppose $DT_{xy}=<X, Y, P_{xy}, T_{xy}, L_{xy}, TV_{xy,} N_{xy} >_X$ and $DT_{yz} = <Y, Z, P_{yz}, T_{yz}, L_{yz}, TV_{yz,} N_{yz} >_Y$ are two ACs of a AC chain, and $DT_{yz}$ is a subsequent AC of $DT_{xy}$ in the delegation path (X→Y→Z), then: (1) $P_{yz} \leq P_{xy,}$ (2) $L_{yz} < L_{xy,}$ (3) $T_{yz} \leq T_{xy.}$ This is called three types of restrictions on the chained delegation.

### 5.1 Creating Delegation Credential with Trust Value

When a grantor (A) creates an AC with trust value, serving as an delegation credential, to a grantee (B), we can summarize the basic steps as follows:

**Step 1:** The grantor gets DTT(A,B) from her own DTT (direct trust table), then computed the direct relationship of A towards B, expressed as Θ(A, B, t), as in Formula (6).
**Step 2:** The grantor queries the grantee's reputation, expressed as Ω(B, t), from the RS in the grantor's domain.
**Step 3:** The grantor computes the Trust Value of the delegation, expressed as $TV_{AB,}$ as in Formula (5).
**Step 4:** The grantor produces the delegation credential with trust value, $DT_{AB}$, by using her PKC to sign a 7-tuple containing $TV_{AB}$ and other information as follows:

$$DT_{AB}=<A, B, P_{AB}, T_{AB}, L_{AB}, TV_{AB,} N_{AB}>_A$$

### 5.2 The Chained Delegation Protocol

Figure 2 shows an example of chained delegation, which the delegation path is SOA→A→B→C→S. It can be divided into four different delegation steps: (1) The initiator SOA assigns attributes to A, and initiates the delegation with A (SOA→A). (2) A delegates relevant privilege attributes to B (A→B). (3) B further delegates relevant privilege attributes to C (B→C). (4) C sends a request to server S (C→S). With the form of " sender → receiver: message ", these steps are described as follows:

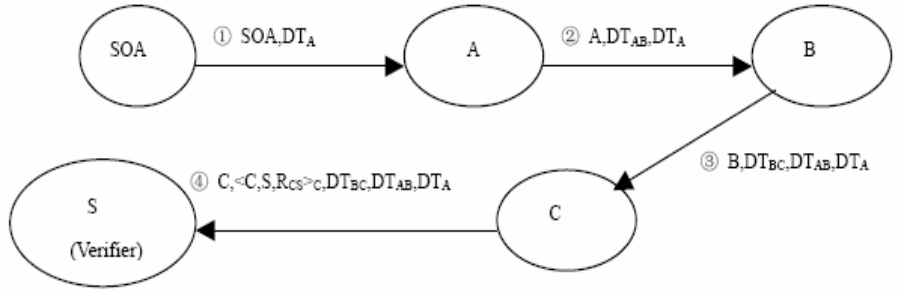



**Fig. 2.** A chained delegation process

**Step 1:** SOA→A: SOA, $DT_A$
Where
$DT_A$=<SOA, A, $P_A$, $T_A$, $L_A$, $TV_A$, $N_A$>$_{SOA}$

**Step 2:** A→B: A, $DT_{AB}$, $DT_A$
Where
$DT_{AB}$=<A, B, $P_{AB}$, $T_{AB}$, $L_{AB}$, $TV_{AB}$, $N_{AB}$>$_A$
$DT_A$=<SOA, A, $P_A$, $T_A$, $L_A$, $TV_A$, $N_A$>$_{SOA}$
Restrictions: (1) $P_{AB} \leq P_A$, (2) $0 \leq L_{AB} < L_A$, (3) $T_{AB} \leq T_A$.

**Step 3:** B→C: B, $DT_{BC}$, $DT_{AB}$ , $DT_A$
Where
$DT_{BC}$=<B, C, $P_{BC}$, $T_{BC}$, $L_{BC}$, $TV_{BC}$, $N_{BC}$>$_B$
$DT_{AB}$=<A, B, $P_{AB}$, $T_{AB}$, $L_{AB}$, $TV_{AB}$, $N_{AB}$>$_A$
$DT_A$=<SOA, A, $P_A$, $T_A$, $L_A$, $TV_A$, $N_A$>$_{SOA}$
Restrictions: (1) $P_{BC} \leq P_{AB}$, (2) $0 \leq L_{BC} < L_{AB}$, (3) $T_{BC} \leq T_{AB}$.

**Step 4:** C→S: C, <C,S,$R_{CS}$>$_C$, $DT_{BC}$, $DT_{AB}$, $DT_A$
Where
$DT_{BC}$=<B, C, $P_{BC}$, $T_{BC}$, $L_{BC}$, $TV_{BC}$, $N_{BC}$>$_B$
$DT_{AB}$=<A, B, $P_{AB}$, $T_{AB}$, $L_{AB}$, $TV_{AB}$, $N_{AB}$>$_A$
$DT_A$=<SOA, A, $P_A$, $T_A$, $L_A$, $TV_A$, $N_A$>$_{SOA}$
$R_{CS}$ denotes a request from C to S
Restrictions: (1) $P_{BC} \leq P_{AB} \leq P_A$, (2) $0 \leq L_{BC} < L_{AB} < L_A$, (3) $T_{BC} \leq T_{AB} \leq T_A$.

## 6 Conclusions

We present a framework to realize restricted delegation using a specific attribute certificate with trust value in Grid environments. The framework employs attribute certificates to convey rights separately from identity certificates used for authentication, and enables chained delegations by using attribute certificate chains. With separate credentials for privileges and identities, we can securely combine privileges from arbitrary sources and build a system that is based on the "least privilege principle", which is not supported by impersonation schemes, such as GSI proxy certificates. Furthermore, in our framework the verifier can realize secure authorization with delegation by checking the trust values of AC chains, and can judge if a delegation is a trusted delegation by evaluating the reputation value of the delegation chain.

## References

1. I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid", *Intl. J. Supercomputer Applications*, vol. 15, no. 3, pp. 200-222, 2001
2. G. Stoker, B. White, E. Stackpole, et al, "Toward Realizable Restricted Delegation in Computational Grids", In *Proceedings of the International Conference on High Performance Computing and Networking Europe (HPCN Europe 2001),* Amsterdam, Netherlands, June 2001.

3. I. Foster, C. Kesselman, G. Tsudik, et al, “A security architecture for computational grids”, *In ACM Conference on Computer and Communications Security Conference*, San Francisco, 1998, pp. 82-89.
4. L. Pearlman, V. Welch, I. Foster, et al, “A Community Authorization Service for Group Collaboration”, *In IEEE Workshop on Policies for Distributed Systems and Networks*, 2002.
5. S. Tuecke, D. Engert, and I. Foster, “Internet X.509 Public Key Infrastructure Proxy Certificate Profile”, Internet Draft, August 2001.
6. J. R. Salzer and M. D. Schroeder, "The Protection of Information in Computer Systems", *Proceedings of the IEEE*, Sept. 1975
7. ITU-T Recommendation X.509 | ISO/IEC 9594-8: Information Technology - Open Systems Interconnection - The Directory: Public-Key and Attribute Certificate Frameworks .
8. RFC3281, An Internet Attribute Certificate Profile for Authorization .
9. Morrie Gasser, Ellen McDermott, “An Architecture for practical Delegation in a Distributed System”, *1990 IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland, CA, May, 1990.
10. Farag Azzedin and Muthucumaru Maheswaran. “Evolving and Managing Trust in Grid Computing Systems”. In Canadian Conference on Electrical and Computer Engineering 2002, pages 1424–1429, May 2002.
11. A. Abdul-Rahman and S. Hailes, “Supporting trust in virtual communities,” *Hawaii Int’lConference on System Sciences*, Jan. 2000.

# Computer Vulnerability Evaluation Using Fault Tree Analysis

Tao Zhang, Mingzeng Hu, Xiaochun Yun , and Yongzheng Zhang

Computer Network and Information Security Technique Research Center,
Harbin Institute of Technology,
Harbin 150001, China
{taozhang, mzhu, yxc,zyz}@hit.edu.cn

**Abstract.** For analyzing computer system security, the system visitor could be classified into five kinds by his privilege to access system resource, and presented the model base on privilege escalation. The attacker can enhance his privilege by exploiting vulnerability, according to distribution of vulnerabilities privilege set, we could construct fault tree to reflect distinctly potential attack path, and so this method could quantificational express security state at different security policy via analyzing fault tree.

## 1 Introduction

The rapid growth of the Internet influences the economy, politics, culture and many aspects of the society. The deeper and wilder the internet application is, the more obvious and more complex the computer and network's security problem is. Hackers and virus find more ways to attack while the network technique developing contrast to the host and terminal age before. The number of vulnerabilities stated by CERT/CC [1] every year grows quickly, 117 in 1995, 419 in 1999, in 2002 the number went up to 4129, and the number of vulnerabilities stated by CERT/CC is more than 10000 till now. At the same time, more and more attack tools that aim at these vulnerabilities are exploited.

As an important aspect of network security, evaluating the computer security through the analysis of the system vulnerabilities is very important and could protect us form being hacked. The main content of this article is constructing a security evaluation method using fault tree analysis and using this method to describe its security status qualitative and quantificational. The organization of this article is follows. Section 2 introduces the related researches and the analysis of computer system is given is Section 3. Section 4 introduces the system evaluation method, and a conclusion is given in Section5.

## 2 Related Work

At the present time, there are two ways to analyze the security status of compute system:

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 302 – 313, 2005.

The first is vulnerability scanning which is a traditional way [2-5]. This method can check whether or not there are any announced vulnerabilities and any simple attack path. This technique has developed for a long time, and there are some famous ones are host scanning tools like COPS, Tripwire and network scanning tools such as Nmap, Nessus etc. The ISS also develops a security evaluation tool constituted of internet scanning tool, database scanning tool and system scanning tool. The approach based on vulnerabilities scanning and rules matching are drawing out the system character already known and summarize rule expression to build up a system character database where one character corresponds to one rule. We can match the target system's information with the rules already existed and the key point is how to generate these rules and how to ensure the accuracy of the information. This technique is just suitable to check system security qualitatively partially and cannot check a whole system. We can't conclude a quantitative describe about the whole target system's security status without thinking of the correlation of the vulnerabilities and the changes of the system status.

The other way is finding the complex attack paths or lists which can lead to changes of the system status by analyzing the security model. The evaluation technique based on model is a good chose if you want to make an effective measurement on the whole computer system. The France Scholar Rodolphe Ortalo developed a method named Privilege Graph [6], Ramakrishnan analyzed these Unix-based systems using model-checking technique [7], And Dr. Wang Yuan in Science and Technology University of China proposed a network security analysis way based on graph and achieved an original system [8]. All these are actual practices based on the model analysis. Although it is easy to build the model which can discover the hidden trouble in system, the research as far as now must be promoted in the building of model and the describing and measuring techniques because subjective elements and absence in model maturity may lead to indefinite conclusions. In practice, a correct security evaluation model needs heavy practice with statistic of security incident, support to security requirement and implements to analyze and detect vulnerability automatically.

The method in this article is building a system security model based on the works which have been finished. We analyze every system status in theory according to the privilege Escalation model, definite the describing way of the vulnerability in system, and also give the correlated path, analyze the security status of target in the way of Fault Tree. Contrast to the scanning vulnerability method, this method can describe a complete graph of the vulnerability distributing and relationship determined by the evaluated object, and this method can also pick out elements which lead to the system status changing. Contrast to the former methods which are based on model analysis, this evaluation system can present us different status changing models according to the different security strategies, and designate the hidden attacking paths show us probability that hackers attack successfully, and avoid the incalculable problems caused by the complex status transfer.

## 3 Definition of Privilege Escalation

Security evaluation mainly analyzes current computer system and needs an simple, flexible and integrate model to reduce complexity of system status space. In this paper

we present a method based on privilege escalation for security evaluation, show related conception and make definition hereinafter:

The difference between security theory and actual environment these breach computer's security function almost be classified two forms: the one is the error of software design, code, for example input validation error, design error, access validation error, and so on, all of these error be concluded in vulnerability database. The other is the conceal relation transfer brought by user change configure for convenience, file share, remote login don't need enter password etc. these condition be definite as vulnerability.

Definition 1: Vulnerability

Vulnerability is a fault produced by an error in the design, development, configuration, or using of software or module, vicious attacker may utilize this fault to unauthorized access system resource and misuse, violate the security policy, and may be produce security incident.

Definition 2: subject and object

Subject is the entity that issue access operation and effect actively, universal it is user or some process of user. $S = \{s_1, s_2 \ldots s_n\}$ is all subject set, $s_i$ (i = 1, 2 … n) is a subject.

Object is the entity that affects passive and it is carrier of information, such as file, storage, interface and so on. $O = \{o_1, o_2 \ldots o_m\}$ is all object set, $o_j$ (j = 1, 2 … m) is a object.

Definition 3: Security Policy

Security policy is the rule that make sure a subject whether have the right to access an object, and whether the activity of an user or process accord with requirement of security, such as discretionary access control policy, mandatory access control policy, integrality and etc. All security policy is $P = \{p_1, p_2 \ldots p_q\}$ and $p_i$ (i = 1, 2, …, q) is a single policy.

System users have the different access right (or privilege) to system resource. For a user, his privilege is certain, an attacker is often play a certain role of system user and usually owns corresponding lower user-level privilege, his target is the higher user-level or administrator privilege to access more resource of system. In the process of role change, his privilege escalates. This paper introduce privilege, privilege set (Pset), privilege escalation to express this concept.

Definition 4: Privilege and Privilege Set

We define a privilege as a （o, m）. o is an object, m is a set of access modes of the subject to this object and m is not null. Privilege Set is expressed by $Pset = \{(o_i, m_j) | (o_i, m_j)$ is a privilege, i=1, 2, … n, j=1, 2, …m$\}$.

Definition 5: Privilege Escalation

If a subject S owning Pset utilize some vulnerability to get a new Pset', and $\exists o', m' \neq \phi$, make $(o', m') \in Pset'$, and $(o', m') \notin Pset$, then we should present that is Privilege escalation.

In practice visitor of computer system should be classified accord with different Privilege set, many research organizations and researchers worked on this aspect, Longstaff present a taxonomy to classify all visitors of computer and divide all visitor into five classes: Remote using a common service, Trusted system, User account, Physical access, Privileged access[9]. In previous work Zhang Yong-zheng uses

Selection Decision Tree (SDT) to classify Psets of all possible users in system [13]. In this paper we classify visitor of system corresponding to privilege set, the potential privilege sets will be divided into five ranks, defined in Table 1.

**Table 1.** Classes of Privilege Set

| Pset-class | Role description |
|---|---|
| Proot | System administrator, managing all system resource such as system device, files and system processes, etc. |
| Psupuser | User which has some special privilege that's not possessed by common user , but only has partial privilege of administrator |
| Puser | Any general system user, which is created by administrator. |
| Pguest | The visitor access anonymously partial resource of system, which has Psubset of a general system user. |
| Paccess | Remote visitor which may access network services, can communicate data with services and scan system |

## 4 Fault Tree Based on Privilege Escalation Graph

### 4.1 Description of Vulnerability

This dissertation mainly studies the problems such as the exploiting, consequence and conjunction of vulnerabilities. By the analysis of several vulnerability databases, universal identification number, the name, the operation systems impacted, the published date of the vulnerability selected from these vulnerability databases as a part of the attributes of the vulnerability database in system background. Vulnerability is expressed by a septenary (BID, NAME, OS, DATE, Ppre, Pcon, AC), where Ppre is the abbreviation for Privilege-set of premise, Pcon is Privilege-set of consequence and AC is Attack Complexity.

*1) Privilege-set attribute of vulnerability*

The classification in this dissertation only studies the direct consequences of vulnerabilities (especially the vulnerabilities that have published attack scripts and programs). And the analysis for the impact of indirect consequences is solved by aftermentioned privilege-escalating graph. According to the privilege-escalating concept, an attacker should utilize vulnerabilities because this vulnerability help to his privilege-set extends. So a vulnerability should have two privilege-set attributes: Ppre and Pcon.

Ppre is a set of necessary privilege which is needed if an attacker attempts to utilize vulnerability. Only by meeting Ppre, the attacker maybe can utilize the vulnerability successfully. Pcon is a set of privileges that are produced by utilizing this vulnerability successfully and the direct privileges escalating. And Pcon indicates the compromise degree of this vulnerability in the privilege tier. Since the Ppre and Pcon distribute in the different rank, the conjunctions between vulnerabilities and a series of different means for privilege escalation is produced. Now there are 751 pieces of vulnerabilities that have been analyzed after taking samples from vulnerability database. Figure 1, 2 and 3 show the distribution plot for the privilege-set attribute of vulnerability.

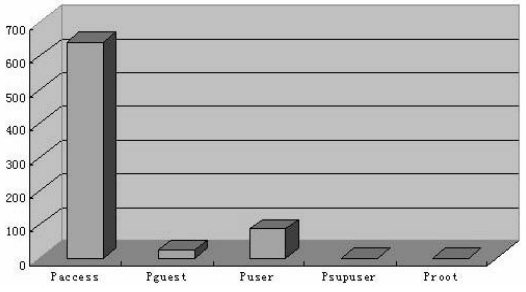



**Fig. 1.** Vulnerability Statistics by Ppre

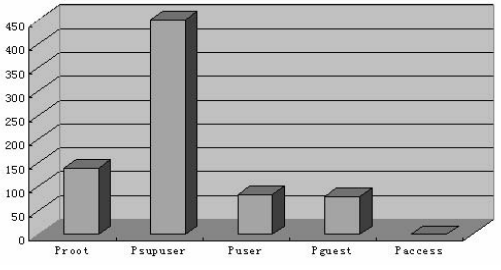



**Fig. 2.** Vulnerability Statistics byPcon

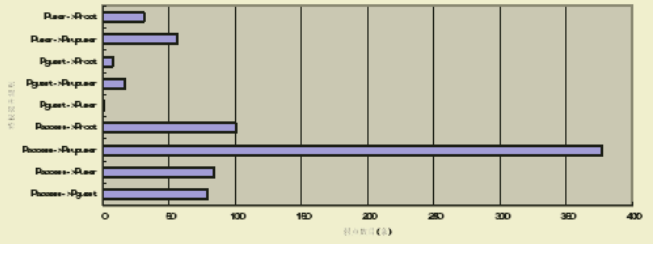



**Fig. 3.** Vulnerability Statistics by Ppre-Ppcon

*2) Attack complexity attribute of vulnerability*

The attack-complexity attribute of vulnerability is used to scale the difficulty of attack-exploiting and getting corresponding Privilege of consequence. It is influenced by many factors such as the availability of attack tool, time needed to perform the attack, compute power of attacker and sometimes also the action of system user etc, which this series of individual variable can be expressed as the attack-complexity of the vulnerability. And there are some ways to setting up the functions about these individual variables and the attack-complexity: analyzing of attack processes and events, definition and rating by security expert or evaluation by the ITSEC and CC evaluation manual.

Since the rapid upgrade of computer's hardware, nowadays personal computer system has gotten powerful compute ability. The popularization of network application provides a environment for the exploitation and promulgate of various attack tools. The multidimensional factors which influence the attack-complexity can mapped some variable functions. From the aspect of software dependability, Littlewood defines a measurement method of attack-complexity attribute of vulnerability, which is based on the random procedure. The quantitative value of a variable is given for vulnerability, which indicates the average effort of vulnerability utilization by attacker. He considers that the attacker can attack some vulnerability successfully as long as adequate effort is paid. The distribution of the effort accords with exponential distribution [10].

It is impossible to exactly describe the attack-complexity attribute of vulnerability. But it can approximately express the difference in attack-complexity attribute of different vulnerabilities by some relational factors which reflect attack-complexity attribute of vulnerability. This paper use the variable to describe attack-complexity attribute of vulnerability, the is the mean probability of success assigned to attack this vulnerability successfully.

### 4.2 Building Privilege Escalation Graph

The purpose of evaluation to computer security is used to describe, predict or compare the quality of system security of concern. Security requirement presents the system resource must be protected and security policy defines the rule these user must be accepted. For different security policy and security requirement, evaluation needs different evaluation policy. So we define security evaluation policy based on privilege escalation in advance as follows:

1) Compare the security quality between different computer operating system at the same times.
2) Describe the change of security quality in the same computer operating system in different times.

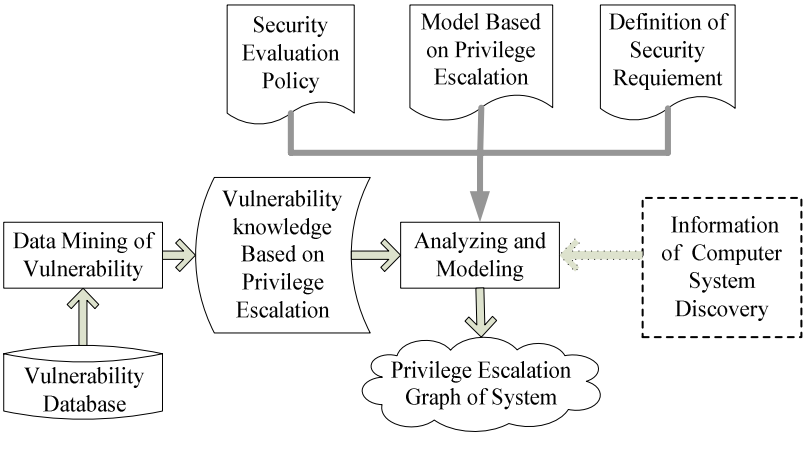



**Fig. 4.** Process of Making System Privilege Escalation Graph

3) Compare the quality of computer security in the same operation system but not same edition.
4) Analyze the real-time security state of computer system.

The first three policies can be fulfilled by the process of real line in Fig4. After realization to mine and coordinate the vulnerability information, at first, estimator sort them by class of operating system. The class of Operating system ($os_1$, $os_2$ … $os_m$) is defined in the vulnerability database. Since vulnerability in the component of operating system owns sole property, and the vulnerability in service program or application program always refers to different operating system type, we can analyze the security of operating system $os_j$ by traversing database and picking out relevant vulnerability set ($v_1$, $v_2$ … $v_n$). By this sort, it can reduce the quantity of data to deal with, and ensure self-contained quality of data set.

In the second, it is sorted by date. All different vulnerability is detected in all kinds of software since release time. So comparing security of the software have the same function, or security of the same system in different times, it must take the date that vulnerability is published as an evaluation factor. At the same time, the attack complexity of vulnerability is strong related with published time of vulnerability.

About the fourth policy, based on anterior knowledge, it adds the process of information detecting, and by tools as active detecting, sniffer, Real-timely and exactly gets the information of evaluation object, such as system type, service, open port, and so on, and attempts to attack some security vulnerability using vulnerability detecting system based on plug-ins. As the broken line part in Fig4, it combines the function of the solid line to realize Real-time analysis of the security of running system.

### 4.3 The Expression and Description of Privilege Escalation Graph Using FAT

According to security evaluation policy that for different security requirement, the privilege escalation process can be expressed by many types such as attack graph or Petri net and so on. Here we express privilege escalation graph with fault tree which is a directness formalization method. There are two basal structure in the attack attempt process with vulnerability, one is that a vulnerability can be used directly in the condition that privilege grade is fulfilled, so many vulnerability with same privilege set property constitute “OR” structure. Another is that under the same privilege grade, there must be two or more vulnerability to escalate the privilege, and these vulnerability constitute “AND” structure. In fault tree, they are expressed with OR element and AND element respectively.

Taking getting administrator privilege of system as an example, a simple privilege escalation graph is showed in Fig5 and Fig6 shows its fault tree. By statistic distribution of Ppre and Pcon, describe it simply as follows:

1) $P_0$, $P_1$, $P_2$, $P_3$, $P_4$ show a corresponding relationship with five privilege grade Paccess, Pguest, Puser, Psupuser, Proot in Table1 by grade from low to high, and show a relevant user’s instance of privilege set in Fig6.
2) To simply depict, mark vulnerability example in graph with $v_1$, $v_2$, $v_3$ … $v_9$, so with Ppre and Pcon vulnerability make up routeways distributed through different privilege grade.

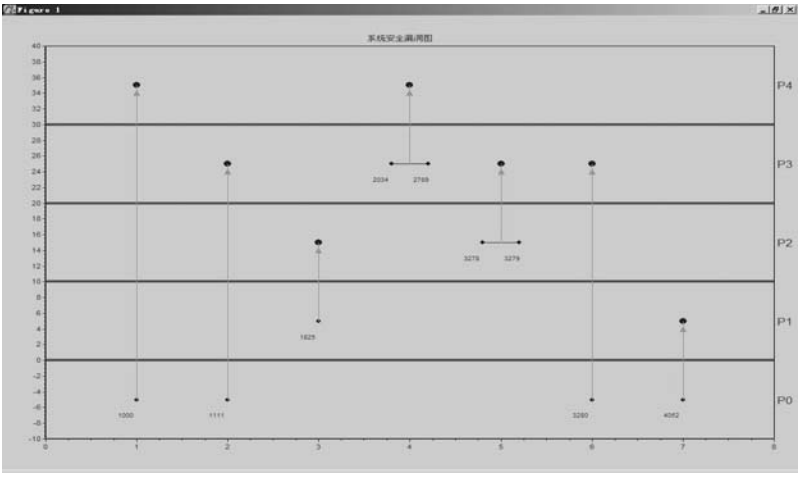



**Fig. 5.** Distributions of Vulnerability Association

3) Every vulnerability has a attack complexity, that is the probability that vulnerability can be used successfully at same condition is sure, expressed with $_1$, $_2$, $_3$ … $_9$, and the attack complexity between every vulnerability is unrelated.
4) Vulnerability $v_4$, $v_5$ must help each other to escalate privilege from $P_3$ to $P_4$, so they make up an And-Gate. ($v_6$, $v_7$) is similar to escalate privilege from $P_2$ to $P_3$. There are four And-Gates in Fig6.
5) There are two inputs to privilege $P_4$: $v_1$, ($v_4$, $v_5$, $P_3$) , and these two inputs, make up a OR gate. Similarly There are three OR gates in Fig6.

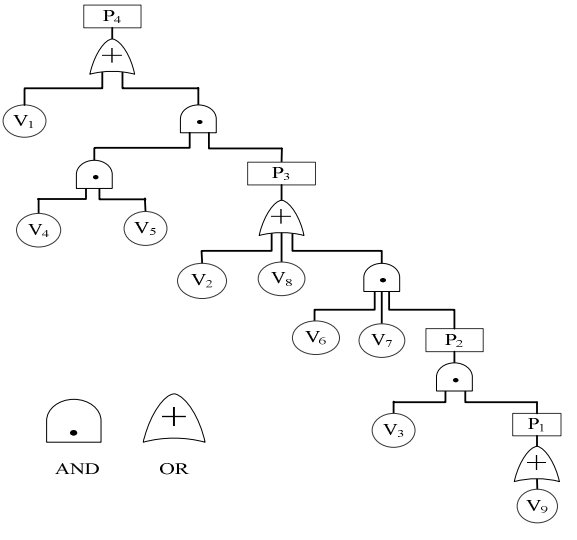



**Fig. 6.** Fault Tree of System Vulnerability Privilege Escalation

### 4.4 Analysis on Security

To analyze the system security expressed in Fig5 and Fig6, based on the analysis of network security incidents and attacker's actions, make assumptions as follows:

Assumption 1: Attacker has lowest privilege P0 at beginning, and his object is to get the highest privilege of system, expressed as P4.

Assumption 2: Attacker has the most attack ability, namely attacker well knows the vulnerability existing in system, and has the ability to attack these vulnerabilities.

Assumption 3: Attack complexity (AC) of vulnerability is known, and use single variable to express the probability that vulnerability can be used successfully in some phase.

1) Qualitative analysis:

For Fig 6 the cut sets are (v1), (v2, v4, v5), (v8, v4, v5) and (v3, v4, v5, v6, v7, v9), so the attacker could choose path as follows:

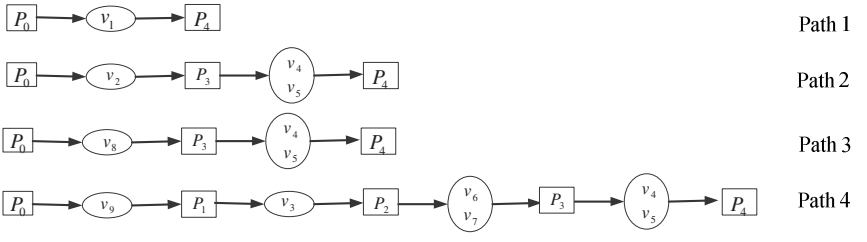



**Fig. 7.** Attack Path

2) Quantificational analysis:

Attacker can begin from state $P_0$, $P(P_0) = 1$. To every path, the probability that attacker can get the highest privilege is:

$$P(P_4) = {}_{1} \quad \text{(1) For Path1}$$

$$P(P_4) = {}_{2\,4\,5} \quad \text{(2) For Path2}$$

$$P(P_4) = {}_{8\,4\,5} \quad \text{(3) For Path3}$$

$$P(P_4) = {}_{9\,3\,6\,7\,4\,5} \quad \text{(4) For Path4}$$

Based on the results of the previous section, Fig5 describes the distribution of vulnerability of system by Ppre-Pcon, these present main qualitative factors correlative with system security. Using fault tree in Fig6, we could quantificationally analyze possibility of attacker gain administrator privilege, and presents rough experimenting result approached this method to WindowsXP, Windows2000 and AIX.

### 4.5 Vulnerability Impact to Security

Computer is a dynamic system it is incessant updated and its setting is often changed. The discovering of new vulnerability, setup of software patch should impact the construction of system privilege escalation and the relative fault tree. For example Fig6, the attacker have the lowest privilege $P_0$ at start, he have many path to choose for getting the highest privilege $P_4$. Now we assume that the vulnerability $v_9$ is removed due to some cause, in this case, though $v_6$, $v_7$ exist there, but they couldn't be exploited because the lack of appropriate privilege, so the fault tree should be constructed as Fig8. If $v_1$ disappears, the attacker only choose these complex path, for the attack complexity of these vulnerability is invariable, the system security is enhanced, the Fig 9 show it.

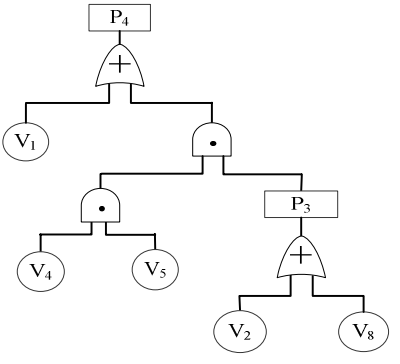



**Fig. 8.** Fault Tree of Removed $v_9$

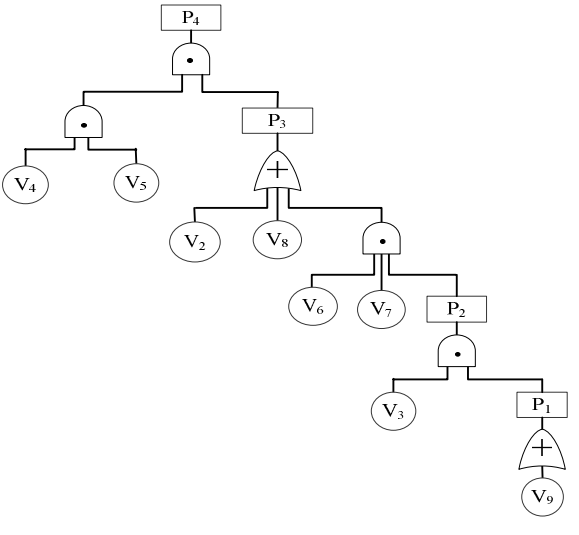



**Fig. 9.** Fault Tree of Removed $v_1$

The rapid growth of the Internet brings corresponding attack utility be developed, spread rapidly after a new vulnerability being discovered, so the attack complexity is a dynamic variable. For example the Red Code hurt the many computers in 2002 is discovered in 2000. It shows that if the automatic attack utility for some vulnerability is developed successfully, corresponding attack path in fault tree should stand out, becomes feeblest.

## 5 Conclusion and Future Work

This paper presented a method on security evaluation. From the privilege escalation graph of system and the expression of fault tree, security state of system is mainly impacted by the number of system vulnerability, privilege distribution of vulnerability and attack complexity of vulnerability, so an exact and self-contained privilege escalation graph can more exactly show security state of system. By traversing graph and calculating weight of every node, we can know the main risk the system maybe face, and offer guidance to enhance the security of system.

Expressing security state of system quantitatively should be a hot topic of this field. Information detecting based on data fusion, determining attack complexity of single weakness, data mining of vulnerabilities are all emphases and difficulty to research in security evaluation model. Along with network technology is used more widely and deeply, there must be new security questions and challenges, so the further work on security evaluation of computer system has very far-reaching significance.

## Acknowledgement

This work was supported by the pre-research project for national defense under the grant No.41315.7.1. Some implementations are done with the help of some members in our research center.

## References

1. CERT/CC.: CERT/CC Statistics 1988-2003. Available online at http://www.cert.org/stats/cert_stats.html#vulnerabilities
2. Marco de Vivo, Eddy Carrasco.: A review of port scanning techniques. ACM SIGCOMM Computer Communication Review, Volume 29, Issue 2, (April 1999) 41-48
3. NMAP.: http://www.insecure.org/nmap/index.html, 2004
4. Ofir Arkin.: ICMP Usage in Scanning, Version 3.0, June 2003
5. ISS.: http://www.iss.com/, 2004
6. Rodolphe Ortalo, Yves Deswarte, Affiliate.: Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security. IEEE Transactions on Software Engineering, Volume 25, Issue 5 (Sept 1999) 633-650
7. C. Ramakrishnan, R. Sekar.: Model-based analysis of configuration vulnerabilities. Journal of Computer Security. Volume 10, Issue1-2 (Oct 2002) 189～209

8. Wang Yuan, Jiang Fan, Chen Guo-liang.: A Network Security Analysis Method Research and Application Based on Graph Theory. Mini-Micro Systems, Volume 24 (10). (Oct 2003) 1865-1869
9. Tom Longstaff. Update: Cert/cc vulnerability knowledgebase. Technical presentation at a DARPA workshop in Savannah. Georgia, February 1997
10. B. Littlewood, S. Brocklehurst, N. Fenton, P. Mellor, S. Page.: Towards Operational Measures of Computer Security. Journal of Computer Security, vol. 2, nos 2/3, (1993) 211-229
11. Xing Xu-jia, Lin Chuang, Jiang Yi-xin.: A Survey of Computer Vulnerability Assessment. Chinese Journal of Computers, Volume 27, (Jan 2004) 1-11
12. Wang Li-dong.: Quantitative Security Risk Assessment Method for Computer System and Network. Ph.D .thesis, Harbin Institute of Technology. 2002.3
13. Zhang Yong-zheng, Yun Xiao-chun.: A New Vulnerbality Taxomomy Based on Privilege Escalation. 2004 Sixth Internatioan Conference on Enterprise Information Systems Proceedings. Porto: INSTICC 2004.

# An Identity-Based Grid Security Infrastructure Model*

Xiaoqin Huang, Lin Chen, Linpeng Huang, and Minglu Li

Department of Computer Science and Engineering,
Shanghai Jiao Tong University, 200030 Shanghai, China
{huangxq, chenlin}@sjtu.edu.cn

**Abstract.** Grid security is a wide topic, touching many of the core issues in information security. It is an area that has been overlooked by the established grid community. In this paper, We explore some roles of identity-based cryptography (IBC) in grid circumstance, and propose a grid security infrastructure model based on identity cryptography. We mainly discuss the grid security authentication and authorization architecture, public key infrastructure based on identity cryptography and security group communication scheme by using weil pairing. The security property of our scheme is discussed. Finally, we compare our ID-based security infrastructure with the public key infrastructure in grid circumstance.

**Keywords:** Identity Cryptography, Public Key Infrastructure, Security Group Communication.

## 1 Introduction

Grid computing aims to provide an infrastructure allowing access to a wealth of sharable resources such as processing power, storage, databases, applications and any other devices (hardware) or components (software). These resource may span a broad range of heterogeneous and geographically dispersed application. Grid security is an area that has been overlooked by the established grid community. A sound and effective security architecture in a grid system is of the utmost importance. Security requirements are fundamental to the grid design. Especially when it is put into the real commercial circumstances [1, 2]. For example, the rules which govern the sharing of resources must be clearly defined: both in terms of who can access what resource, and under what conditions.

The results of this research have been incorporated into a widely used software system called the Globus Toolkit [3] that uses public key technologies to address

* This paper is supported by SEC E-Institute: Shanghai High Institutions Grid project, the 863 high Technology Program of China(No. 2004AA104340 and No. 2004AA104280), Natural Science Foundation of China(No. 60433040 and No. 60473092)and ChinaGrid Program of MOE of China.

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 314–325, 2005.

issues of single sign-on, delegation [4], and identity mapping, while supporting standardized APIs such as GSS-API [5]. The basic grid security model consists of: User, Proxy, Mutual Authentication Process, Service, Access Control List, Backup Logs. The mutual authentication process is the core of the security model. Instead of using an LDAP repository to hold the public key (PKI), two parties who want to communicate with one another use their public key stored in their digital certificate to authenticate with one another. In Grid Security Infrastructure [20, 11], the authentication is composed of X.509 certificates and digital signature schemes. The certificate consists of user name, issuer, and a public key. The digital signature algorithms are usually RSA. But in recent years, cryptography based on identity is developing very quickly.

The concept of identity-based cryptography is due to Shamir [6]. Boneh and Franklin presented an identity-based encryption scheme based on properties of the Weil and Tate pairings on elliptic curves [7]. The properties of identity-based cryptography (IBC) may well match the qualities of grid computing. In the traditional public key cryptography, we must get users' public keys from Certificate Authority (CA). When there are thousands of users in grid circumstance, the CA will become the bottleneck. But in the IBC system, the public key may be the user's identity, for example, e-mail address, a user just get a secret key from Private Key Generator (PKG) once. The advantages of ID-based system are obvious.

Our purpose in writing is to identify the security issues that are distinctive within a grid environment (as distinctive from the classic client-server model commonly used over the Internet). Some of these issues have been clearly highlighted in the grid literature [20], but many of the subtleties remain unclear. It is our observation, in particular within the ChinaGrid project, that security poses substantial problems in the development and deployment of grid technology. Therefore we consider it to be a good time to explore any roles for IBC in grid circumstance. In this paper, we propose an ID-based security infrastructure for the authentication and authorization process, the secure group communication scheme and a public key infrastructure based on identity-based cryptography.

## 2 Related Work

### 2.1 ID-Based Short Signatures Without Random Oracles

Dan Boneh and Xavier Boyen present ID-based short signatures in paper [8]. The paper uses the same notation as in [7]. The bilinear maps and bilinear map groups satisfy the following properties:

1. $\mathbb{G}_1$ and $\mathbb{G}_2$ are two (multiplicative) cyclic groups of prime order $p$ ;
2. $g_1$ is a generator of $\mathbb{G}_1$ and $g_2$ is a generator of $\mathbb{G}_2$ ;
3. $\psi$ is an isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, with $\psi(g_2) = g_1$; and
4. $e$ is a bilinear map: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.

Let thus $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups as above, with an additional group $\mathbb{G}_T$ such that $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T|$. A bilinear map is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

1. Bilinear: for all $u \in \mathbb{G}_1, v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degenerate: $e(g_1, g_2) \neq 1$.

Let $(\mathbb{G}_1, \mathbb{G}_2)$ be bilinear groups where $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ for some prime $p$. As usual, $g_1$ is a generator of $\mathbb{G}_1$ and $g_2$ a generator of $\mathbb{G}_2$. For the moment we assume that the messages $m$ to be signed are elements in $\mathbb{Z}_p^*$, the domain can be extended to all of $\{0,1\}^*$ using a collision resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$.

**Key Generation:** Pick random $x, y \leftarrow_R \mathbb{Z}_p^*$, and compute $u \leftarrow g_2^x \in \mathbb{G}_2$ and $v \leftarrow g_2^y \in \mathbb{G}_2$. The public key is $(g_1, g_2, u, v)$. The secret key is $(x, y)$.

**Signing:** Given a secret key $x, y \in \mathbb{Z}_p^*$ and a message $m \in \mathbb{Z}_p^*$, pick a random $r \in \mathbb{Z}_p^*$ and compute $\sigma \leftarrow g_1^{1/(x+m+yr)} \in \mathbb{G}_1$. Here $1/(x + m + yr)$ is computed modulo $p$. In the unlikely event that $x+m+yr = 0$, we try again with a different random $r$. The signature is $(\sigma, r)$.

**Verification:** Given a public key $(g_1, g_2, u, v)$, a message $m \in \mathbb{Z}_p^*$, and a signature $(\sigma, r)$, verify that

$$e(\sigma, u \cdot g_2^m \cdot v^r) = e(g_1, g_2) \tag{1}$$

If the equality holds the result is valid; otherwise the result is invalid.

### 2.2 One Round Tripartite Diffie-Hellman Protocol

Antoine Joux [9, 10] presented a new efficient one-round tripartite key agreement protocols. A pairing is a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ with group $\mathbb{G}_1$ and $\mathbb{G}_2$ of a large prime order $q$, which satisfies the above Bilinear, Non-Degenerate and Computable properties. If Alice, Bob and Charlie want to get a secret share key, the protocol requires each party to transmit only a single broadcast message to establish an agreed session key among three parties.

$$\begin{aligned} A &\to B, C : aP \\ B &\to A, C : bP \\ C &\to A, B : cP \end{aligned} \tag{2}$$

After the session, A computes $K_A = (bP, cP)^a$. $B$ computes $K_B = (aP, cP)^b$ and $C$ computes $K_C = (aP, bP)^c$. The established session key is $K = K_A = K_B = K_C = (P, P)^{abc}$.

### 2.3 ID-Based Non-interaction Secret Sharing Protocol

Ryuichi Sakai [12] proposed non-interaction secret sharing protocol. $ID_A$ and $ID_B$ are respectively the identity information of Alice and Bob. $S_{IDA} = sH(ID_A)$ and $S_{IDB} = sH(ID_B)$ is respectively their secret key, $s$ is the system secret key. The protocol is as follows:

1. Alice computes the $K_{AB}, K_{AB} = \hat{e}(S_{IDA}, P_{IDB})$.
2. Bob computes the $K_{BA}, K_{BA} = \hat{e}(S_{IDB}, P_{IDA})$.
3. Using map $\hat{e}$ properties, we can verify $K_{AB} = K_{BA}$.

$K_{AB} = \hat{e}(S_{IDA}, P_{IDB}) = \hat{e}(sP_{IDA}, P_{IDB}) = \hat{e}(P_{IDA}, P_{IDB})^s = \hat{e}(P_{IDA}, sP_{IDB})$
$= \hat{e}(P_{IDA}, S_{IDB}) = K_{BA}$

The protocol doesn't need interaction.

## 3 Our ID-Based Grid Secure Authentication and Authorization Schemes

First of all, we introduce some major components in grid security architecture [15].

**Virtual Organization (VO):** A dynamic collection of users and resources that potentially span multiple administrative domains and governed by a set of defined sharing rules.

**User:** A subscriber to a grid. The user can belong to a VO or multiple VOs and he may share part or all of his local resources to other users.

**Resource:** This comprises from any sharable resource including hardware and software. A user can be a resource to other users if he could offer part or all of his local resource.

PKI is an authentication enabling technology. Using a combination of secret key and pubic key cryptography, it enables a number of other security services including data confidentiality, data integrity, and key management. Despite the importance of PKI, There is an increased research focus on IBC. The main stimulus for this trend is the problem of managing certificates and their associated keys using PKI. Imagine that there are a few thousands Grid users of the same VO spread across different continents of the world within the Grid infrastructure. Each user needs a copy of the certificates of other users or resources with whom he wishes to communicate. Note that this has not included interaction between the users and other users and resources of other VOs. Furthermore, some of the certificates may be invalidated quickly as users leave that VO [15]. So in the grid security architecture, we introduce the IBC to alleviate these problems and perhaps provide some advantages.

### 3.1 Grid User Authentication Schemes

In usual, there are two kinds of certificates: Identity Certificate and Authorization Certificate. The Identity Certificate is used to indicate the identity information. Usually we use the X.509 certificate form. This include: (Subject, Issuer, Public Key, Validity, Signature). Issuer is the Certificate Authority and Subject is the certificate's owner. Public Key is the owner's public key. Validity is the validity period of the certificate. Signature is the CA's signature. Authorization Certificate is the access rights to access some resources. The details will be

discussed in the section 3.2. When Alice wishes to communicate securely with Bob, she could simply encrypt the message with public key string: 'Bob's DN ‖ timestamp'. When we use the Identity-based cryptography, we neither need user's public key certificate nor verify his identity as the authentication task has been indirectly transferred to the PKG. Bob needs to authenticate himself to the PKG before he receives the appropriate corresponding private key [15].

### 3.2 Grid User Authorization Schemes

In this section, we describe a authorization scheme for a user to access a grid resource. The steps are as follows:

1. If a user(Alice) wants to access the grid resource, she first send her identity, for example her ID= (e-mail address ‖ date) to the Private Key Generator (PKG). Suppose the PKG has the public/ private key pair $(R_{PKG}, s)$, where $R_{PKG} = sP$, $P \in G_1$. By using the scheme of paper [7], the user (Alice)'s public key for signature generation is $P_{Alice} = H_1(ID)$. $H_1 : \{0,1\}^* \rightarrow G_1$. The PKG generates the user's secret key for signature $S_{Alice} = sP_{Alice}$. The PKG sends the $S_{Alice}$ to the user by a secure channel. Also the parameters (a finite message space and a finite ciphertext space) of PKG are publicly known.
2. The authorization certificate is represented by the 5-tuple: Issuer, Subject, Delegate, Authorization, Validation. Delegate is a Yes/No flag, Authorization is the description of what is being authorized and Validity is the validity period [16]. PKG forms the expression given by

$$\sigma = P_{Alice} \| Delegate \| Authorization \| Validity \tag{3}$$

   and then forms the public/private key pair given by $S_\sigma = sQ_\sigma$ where $Q_\sigma = H_1(\sigma)$. PKG then gives the private key $S_\sigma$ to Alice. The authorization public key $Q_{Alice}$ for Alice ($Q_{Alice} = Q_\sigma$) is also public known.
3. For Alice to use this resource now, she needs to demonstrate
   - She knows the private key corresponding to $P_{Alice}$, i.e. she is Alice.
   - She knows the private key corresponding to $Q_{Alice}$, i.e. PKG has given her the authorization.

   To demonstrate the two facts she can produce a signature to a random number using $S_{Alice}$ and a signature to $\sigma$ using the $S_\sigma$. The resource can verify the two signatures by the two corresponding public keys. If both pass the verification, then resource permits Alice to access the resource.

### 3.3 Grid Security Communication

In the current GSI, to allow secure communication within the grid, the OpenSSL package is installed as part of the Globus Toolkit [11]. OpenSSL is a software package that is used to create an encrypted tunnel using SSL/TSL between grid clients and servers. The authentication process is as follows [11]:

1. A grid user contacts a remote grid host to start a secure session by using a digital X.509 ID certificate.
2. The grid user and the host send the SSL version number, cipher settings, the digital certificate and so on each other.
3. The host verifies the user's certificate and the user verifies the host's certificate, if both are passed, the user encrypts the session key with the host's public key so that only the host can read the session key.
4. The server decrypts the session key using its own private key. The user sends a message to the host informing it that future messages from the user will be encrypted with the session key.
5. An SSL-secured session is now established. SSL then uses symmetric encryption to encrypt and decrypt messages within the SSL-secured "pipeline".

In the above process, the authentication process is complicated and needs some computing. But in our scheme, we can use the ID-based non-interaction secret sharing protocol to get the session key. There is no interaction. Suppose Alice and Bob want to communicate securely, each uses the (e-mail address $\|$ date) as the identity. So $ID_A$=(Alice e-mail address $\|$ date) and $ID_B$=(Bob e-mail address $\|$ date). $S_{IDA} = sH(ID_A)$ and $S_{IDB} = sH(ID_B)$ is respectively their secret key. $s$ is the system secret key. Alice computes the $K_{AB}, K_{AB} = \hat{e}(S_{IDA}, P_{IDB})$. Bob computes the $K_{BA}, K_{BA} = \hat{e}(S_{IDB}, P_{IDA})$. They get their session key $K_{AB} = K_{BA}$. Then they can use the session key to communicate securely.

### 3.4 Proxy Certificate

In grid circumstance, if a user uses a certificate, then he has to open his private key many times. We adopt the similar method with globus [11].

**Proxy Creation:**

1. A trusted communication is created between host A and host B.
2. Host A request host B to create a proxy certificate that delegates host A's authority.
3. Host B creates the request for host A's proxy certificate, and send it to PKG.
4. PKG computes $\sigma' = P_A \| 1 \| Delegate \| Authorization \| Validity$ and forms the public/private key pair given by $S_{\sigma'} = sQ_{\sigma'}$ where $Q_{\sigma'} = H_1(\sigma')$. PKG then gives the private key $S_{\sigma'}$ to host B.
5. Host B signs the $\sigma'$ using $S_{\sigma'}$.

**Proxy Action:**

1. A proxy sends host A's signature and host B's signature to host C.
2. Host C verifies the signatures using host A's public key $Q_\sigma$ and the proxy's public key $Q_{\sigma'}$.
3. If both pass the verification, then the host C compares the two Subjects. If one is $P_A$ and the other is $P_A \| 1$, the host C permits to access the resource; otherwise rejects.

The signing algorithm is the D. Boneh's short signature algorithm as in section 2.1.

## 4 Public Key Infrastructure Based on Identity Cryptography

As in paper [15, 16], there is a lot of virtual organizations. In each virtual organization, if there is only one PKG, this is a risky situation as the PKG is the single point of failure and becomes the workload bottleneck of the system. In a VO, there are several Trusted Communities (TCs), a TC has one PKG who is trusted by all entities (users and resources). If users are in the same TC, they can use IBC approach to communicate securely. If users belong to different PKGs, for example, Alice in a PKG1 want to communicate with Bob or Resource in another PKG2, Alice needs to obtain PKG2's public key certificate from the Grid CA through the normal method in PKI [15]. The architecture is in the Fig.1. In the

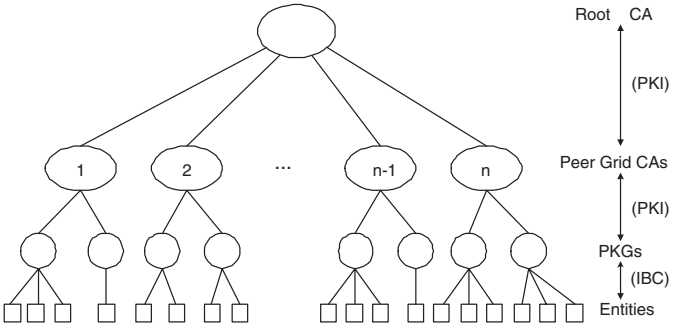



**Fig. 1.** A trust model for grid environment

Fig.1, each VO has a corresponding peer grid CA. When there are a lot of VOs, there are a lot of peer grid CAs. Grid users who trust one CA can not verify the certificates issued by the others. So we have to establish the trust relationships between CAs using cross-certification. The X.509 specification [19] defines a cross-certification in this way: " Two CAs exchange information used in establishing a cross-certificate. A cross-certificate is a certificate issued by one CA to another CA which contains a CA signature key used for issuing certificates." We adopt a distributed CA model using cross-certification as in paper [17]. But in this model, certification path processing is not efficient. For example, when a verifier who trust $CA_1$ verifies the certificate issued by $CA_n$, the certification path length is $n$. In order to reduce the length, we can use G. Ateniese's group signature scheme [18]. All peer grid CAs compose a group, where the root CA is the group manager. Each CA has a private key, but they have a same group public key. So each CA can sign his certificate, a verifier can verify it using the group public key. The certification path length is 1. Especially when grid scale is large, it is a very efficient method.

# 5 Our Secure Group Communication Scheme in Grid Computing

## 5.1 Security Group Communication Requirements

Grid systems may require any standard security functions, including authentication, access control, integrity, privacy, and nonrepudiation [20]. In order to develop security grid architecture, we should satisfy the following characteristics:

- Single sign-on: A user should be able to authenticate once.
- Protection of credentials: User credentials must be protected.
- Interoperability with local security solutions: Security solutions may provide interdomain access mechanisms.
- Exportability: We require that the code be (a) exportable and (b) executable in multinational testbeds.
- Uniform credentials/certification infrastructure: Interdomain access requires a common way of expressing the identity of a security principle such as an actual user or a resource.
- Support for secure group communication.

A computation can comprise a number of processes that will need to coordinate their activities as a group. The composition of a process group can and will change during the lifetime of a computation. Therefore, secure communication for dynamic groups is needed. So far, no current security solution supports this feature; even GSS-API has no provisions for group security contexts [20].

In our computational grid security architecture, there are a lot of processes. These processes need to communicate securely. They have to share a common secret key.

## 5.2 Our Secure Group Communication Scheme

In Grid computing circumstance, a group member often joins/leaves the group dynamically. When a user joins the group, the group key should change to a new value. Also when a user leaves a group, the original key should become invalid and the remaining member should have a new group key. So our group communication scheme should support the dynamic circumstance. Our algorithm consists of three steps as follows:

**Setup:** Suppose our system has two members initially. By using Sect.2.3 ID-based Non-Interaction secret sharing protocol, the user $u_1$ computes: $K_{AB} = \hat{e}(S_{IDA}, P_{IDB})$ and $u_2$ computes: $K_{BA} = \hat{e}(S_{IDB}, P_{IDA})$. $u_1$ and $u_2$ hold the sharing secret key $K_{AB} = K_{BA}$. The structure is as Fig.2.

**Join:** As in Fig.2, $u_3$ wants to join the group. We use the A. Joux's one round tripartite Diffie-Hellman protocol. A bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ with group $G_1$ and $G_2$, $P \in G_1$.

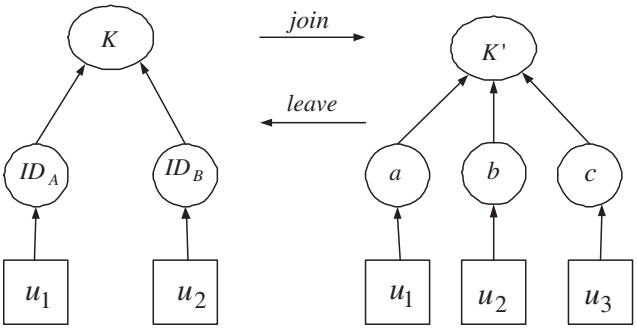



**Fig. 2.** Sharing secret key graphs before and after a user joins/leaves

To achieve the sharing secret key, the process is as follows:

- $u_1$ randomly chooses $a$ and computes $P_A = aP$.
- $u_2$ and $u_3$ randomly chooses $b, c$ and respectively compute the values:$P_B = bP, P_c = cP$.
- $\hat{e}(P_B, P_C)^a = \hat{e}(P_A, P_C)^b = \hat{e}(P_A, P_B)^c = \hat{e}(P, P)^{abc}$.

So $u_1, u_2, u_3$ holds the sharing secret key $\hat{e}(P, P)^{abc}$.

**Remove:** Suppose $u_3$ want to be deleted in Fig.3. The sharing secret key $\hat{e}(P, P)^{abc}$ is revoked. $u_1$ and $u_2$ get their sharing secret key using Sakai's Non-Interaction Secret Sharing Protocol. It is very simple.

When there exists sub-processes, for example, a parent process generates three sub-processes or there are a lot of users which want to share their secret key, we can use the following method. If a user generates three separate processes in order to complete a task, he generates three processes and each process generates three sub-processes as in Fig.3. The three sub-processes and the parent process can get their shared secret key as above. The $u_1, u_2, u_3$ and their parent process

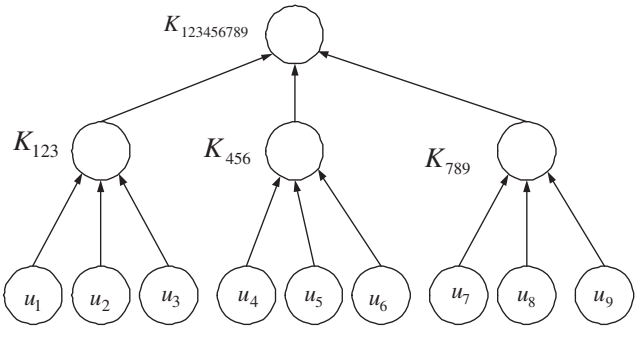



**Fig. 3.** Sharing secret key method when processes have sub-processes

can get the sharing key $K_{123}$. The $u_4, u_5, u_6$ and their parent process can get the sharing key $K_{456}$. The $u_7, u_8, u_9$ and their parent process can get the sharing key $K_{789}$. All users can get their sharing key $K_{1-9}$.

## 6 Security Analysis and Comparisons

IBC is a relatively new technology in comparison with PKI. In this field, a lot of new results have been achieved in recent years. We consider its applications in a grid system. So grid security is based on the Identity-based cryptography. When the grid system is developed to a large scale, the excellence is obvious. We just use other people's e-mail address $\|$ date as the public key, rather than get other people's public key from CA each time. When there are tens of thousands people using the CA, the CA will become the bottleneck. Also the short signature algorithm just need half the size of DSA signature with the same level of security. The short signature scheme which is existentially unforgeable under a chosen message attack without using random oracles. The security of the short signature scheme depends on a new complexity assumption called the Strong Diffie-Hellman assumption. This assumption has similar properties to the Strong RSA assumption. The short signature scheme is much shorter and simpler than signatures from schemes based on Strong RSA [8]. So our grid security authentication and authorization architecture is more practical than the traditional grid architecture based on the PKI. In our secure group communication scheme especially in the dynamic case, we use the One Round Tripartite Diffie-Hellman and ID-based non-interaction secret sharing protocol, it is more efficient than the traditional authenticated group Diffie-Hellman key-exchange protocol.

However, there are two drawbacks with identity-based cryptosystems [15]. The first drawback is the need for a user to maintain an independent secure channel for the distribution of the private key. This need not be such a problem within a grid infrastructure and within a controlled and closed VO, though it does remain an important area for future research. The second drawback is that the PKG knows the private keys of all its users. This can be solved by distributed PKGs and threshold cryptography. The analysis is in section 6.3.

### 6.1 Non-interactive Authentication

In the current version of GSI, if the user wants to access the resource, he first has to pass the authentication by the resource proxy. In our identity-based grid secure model, the computational load is about twenty to thirty percent of it has to perform in the current GSI [21]. In GSI, user proxy is in a computationally heavily loaded point both in computation and in communication. So the current GSI technique has a poor scalability. But in the identity-based grid security

infrastructure, the authentication process is non-interactive. The computing load is reduced and the scalability is good.

### 6.2 Forward Security Properties

In our grid security infrastructure, the user's public key can be $P_{user} = H_1(ID)$. $H_1 : \{0,1\}^* \rightarrow G_1$ and the user's secret key is $S_{user} = sP_{user}$. Where ID= (e-mail address $\|$ date). When the date is changed every day or every month, the user's secret key is changed every day or every month. If the secret key is exposed now, the user's secret key before is still valid. So the user's signature before is still valid. The user's signature is with the forward security properties. But in the current GSI, if the user's secret key is exposed, all the user's signature becomes invalid.

### 6.3 Security of Private Key Generator

When there is one PKG, the users' secret key is easily to be exposed, if the PKG is dishonest. This problem can be resolved by using a plural number of PKGs who collectively share the system master key $s$ [21]. Let $PKG_i$ denote an individual PKG whose master secret key is $s_i$. Then the user can obtain its private key from a list of PKGs. The user's private key is as follows:

$$S_{user} = \sum_i s_i P_{user} = \sum_i s_i H_1(ID) \tag{4}$$

Suppose these PKGs do not collude, then no one knows the user's private key. So the problem that the PKG knows the private keys of its users can be resolved very well.

## 7 Conclusions

Grid computing is one of today's most important technologies. The security problem is a complicated and important problem in grid computing. Current implementations rely on the traditional PKI technology. In this paper, we introduce a new cryptography technology (Identity-based Cryptography) into the grid security. We design the grid secure authentication and authorization architecture based on ID-based cryptography. We have discussed an IBC/PKI hybrid security architecture. To reduce the cross-certification path length between CAs, we adopt the group signature scheme. We also have discussed some security requirements in grid circumstance. Because secure group communication is seldom considered before, we propose a secret key exchange protocol based on Joux's One Round Tripartite Diffie-Hellman Protocol. The situation that processes have sub-processes is also being considered. The inherent qualities of IBC appear to closely match the demands of a dynamic environment in the grid circumstance where the availability of resources can change over the time frequently.

## References


1. Foster, I. and Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999, 2-48.
2. Welch, V., Siebenlist, F. and Foster, I.: Security for grid services. Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03), 2003.
3. Butler, R., Foster, D., Kesselman, I.: A National-Scale Authentication Infrastructure. IEEE Computer, 33(12). 60-66. 2000.
4. Gasser, M. and McDermott, E.: An Architecture for Practical Delegation in a Distributed System. Proc. 1990 IEEE Symposium on Research in Security and Privacy, 1990, IEEE Press, 20-30.
5. Linn, J.: Generic Security Service Application Program Interface, Version 2. INTERNET RFC 2078, 1997.
6. Shamir, A.: Identity-based cryptosystems and signature schemes in Proc. Crypto'84, LNCS Vol.196, Springer, pp.47-53, 1985.
7. Boneh, D. and Franklin, M.: Identity-Based Encryption from the Weil pairing," in Proc. Crypto 2001, LNCS Vol.2193, Springer, pp. 213-229, 2001.
8. Boneh, D. and Boyen, X.: Short Signatures Without Random Oracles. EUROCRYPT 2004, LNCS 3027, pp.56-73.
9. Joux, A.: A one-round protocol for tripartite Diffie-Hellman. Algorithm Number Theory Symposium -ANTS-IV, Lecture Notes in Computer Science 1838, Springer-Verlag (2000), pp. 385-394.
10. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. Journal of Cryptology. Vol17, pp.263-276, 2004.
11. Introduction to grid computing with globus. http://www.ibm.com/redbooks.
12. Sakai, R., Ohgishi, K. and Kasahara, M.: Cryptosystems based on pairing. In SCIS, Okinawa, Japan, 2000.
13. Lock, R. and Sommerville, I.: Grid Security and its use of X.509 Certificates. http://www.comp.lancs.ac.uk/computing/.
14. Groep, D., NIKHEF.: User Authentication Principles and Methods. http://www.nikhef.nl/ davidg/.
15. Lim, H.W. and Robshaw, M.J.B.: On Identity-Based Cryptography and Grid Computing. ICCS 2004, LNCS 3036, pp. 474-477, 2004.
16. Chen, L., Harrison, K., Moss, A., Soldera, D. and Smart N.P.: Certification of Public Keys Within an Identity Based System. ICS 2002, LNCS 2433, pp. 322-333, 2002.
17. KOGA S. and SAKURAI K.: Decentralization Methods of Certification Authority Using the Digital Signature Schemes. Gaithersburg, Maryland, USA April 28-29, 2003.
18. Ateniese, G., Camenisch, J., Joye, M. et al: A Practical and Provably Secure Coalition-resistant Group Signature Scheme. In CRYPTO 2000, Santa Barbara, CA, USA Aug 20-24, 2000.
19. Internet X.509 Public Key Infrastructure Certificate and CRL profile, RFC3280, 2002.
20. Foster, I., Kesselman, C., Tsudik., G.: A Security Architecture for Computational Grids. ACM Conference on Computers and Security, 83-9, 1998.
21. Mao, W.: An Identity-based Non-interactive Authentication Framework for Computational Grids. http://www.hpl.hp.com/techreports/2004/HPL-2004-96.pdf.

# Towards Multilateral-Secure DRM Platforms

Ahmad-Reza Sadeghi and Christian Stüble

Horst Görtz Institute for IT-Security,
Ruhr-University Bochum, Germany
sadeghi@crypto.rub.de
stueble@acm.org

**Abstract.** Digital Rights Management (DRM) systems aim at providing the appropriate environment for trading digital content while protecting the rights of authors and copyright holders. Existing DRM systems still suffer from a variety of problems that hamper their deployment: they (i) cannot guarantee policy enforcement on open platforms such as today's PCs, (ii) offer only unilateral security, i.e., focus mainly on requirements of the content owners/providers and not on those of consumers such as privacy, and (iii) restrict users regarding many legally authorized uses (fair use), e.g., disallow consumers to make backups.

In this paper we present a security architecture for computing platforms that, in the sense of multilateral security, is capable of enforcing policies defined by end-users and content providers. Our model provides methods and principles to practitioners to model and construct such systems based on a small set of assumptions. Further, we show how such a platform can be implemented based on a microkernel, existing operating system technology, and trusted computing hardware available today. Moreover, the platform's functionality can be extended with a mechanism called property-based attestation to prevent discrimination of open-source software and to protect the consumers' privacy.

## 1 Introduction

Trading digital goods over open networks, such as the Internet, is becoming increasingly important, and there is still a high potential to be utilized for a variety of business models in the area of distributed content-delivery. Although technical advancements improve the creation of different types of content and their secure distribution, they can also be used against the interests of copyright holders and open up new ways of misuse [18]. As a consequence, digital properties are similar to "'public goods"' and hardly any consumer is willing to pay for them [5]. In this context *Digital Rights Management (DRM) systems* should serve the setup of flexible electronic distribution chains for digital works which continuously enforce the adherence to copyrights [23]. In order to achieve this goal, DRM systems deploy various building blocks, such as content distri-

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 326–337, 2005.

bution and rights distribution infrastructures, payment-systems, access control and copy-protection methods, to name just the most prominent ones.[1]

DRM systems are concerned with a different trust model than typical security models in Internet: the trust resides at consumer's side who is seen as a potential thread!

The core technical part of a DRM system is a component called *trusted viewer*. Its task is to enforce the desired policies attached to the content according to terms and conditions of content providers, e.g., preventing consumers from unauthorized copying and distribution of the content. Existing trusted viewers, however, have not been successful on today's computing platforms such as PCs, simply because PCs are general purpose devices and thus are freely programmable and extensible. This offers potential attackers a wide range of possibilities for tampering. Users have full control over the operating system and its configuration. Since operating systems control applications and thus have full access to application data, users can bypass security policies of content providers by appropriately reconfiguring the underlying operating system [7, 10]. Therefore, content providers tend to *closed* DRM systems, preferring platforms with fixed configurations that are trusted not to allow consumers to circumvent their security policies. The consequence is that existing DRM systems aim at providing *unilateral* security focusing only on the needs of the content providers and leaving out those of consumers: using platforms that are under control of external parties (e.g., content providers) may have potential negative consequences regarding consumer's security such as privacy violation or censorship [3]. Moreover, DRM systems may restrict users' rights strongly regarding legally authorized uses in the context of *fair use* by, e.g., by preventing using from making private copies or backup [6, 9, 17].

These problems are highly important among those that have led to a general mistrust against DRM systems, especially in the open-source community. Hence, we require a security architecture that is capable of providing *multilateral* security on open platforms, and that can be implemented effectively by the technologies that are available today (see also [24, 25]). The main goal of this paper is to present methods and principles to practitioners enabling them to model and construct such systems and to implement them based on a small set of assumptions.

**Our Contribution.** We define a model of a computing platform that, in the sense of multilateral security, enforces policies defined by consumers and content providers. Moreover, we design the platform such that it is capable of offering fair solutions in case of conflicts. We derive a reasonable set of security properties to be provided by such a platform, and show that the resulting platform fulfills them. Further, we show how such a platform can be implemented based

[1] Despite the effectiveness of these technical protection mechanisms the protection of copyrights in practice can be satisfactorily solved only through an appropriate and coordinated combination of technical with legal and economical measures (see, e.g., [28]).

on microkernel, existing operating systems and trusted computing technology available today. On this platform, users can still use their existing computing environment, e.g., the operating system they are used to (here, we consider Linux). Last but not least we discuss further extensions to our platform such as *property-based attestation* mechanism [26, 22] enforced on the platform level as well as on the application level to prevent discrimination of open-source software.

**DRM Systems: Roles and Trust Relations.** A general DRM system includes various parties, components, and functionalities [12, 8]. The main parties involved are *content providers* $P$ and *consumers* $U$. Content providers distribute digital contents which are licensed by the consumers. To protect the content, $P$ enforces its access control policy that we call *DRM policy* $SP_P$. The consumer owns the *DRM platform*, the main component of the *DRM system* consisting of all the software and hardware components that are under its control. The consumer consumes content distributed by the content provider on the DRM platform using a *trusted viewer* that renders the content. We also consider a further role in a DRM system, namely the *platform vendor* who produces software components of the DRM platform. The platform vendor can be a company that sells the DRM platform, or even the open-source community that freely distributes its DRM-enabled operating system.

Certainly, the trust relationships among parties involved in an DRM system may vary strongly and conflicts may arise.[2] Ideally, a DRM system has to guarantee that the security requirements of all involved parties can be fulfilled, i.e., it should provide a multilateral secure environment.

## 2 Towards an Open Multilateral-Secure DRM Platform

In this section, we define a model of a multilateral-secure DRM platform where we first start with an ideal platform and then gradually approach the real world model and give arguments for its security. Due to lack of space we can only give a rough overview over the required steps. A more detailed discussion including more formal definitions and security arguments is available in the full version of this paper [27].

In our models, consumers, content providers, communication channels, and components of the DRM system are modeled as I/O automata called *machines* (see also [16]). The semantics of our graphical model are as follows: machines are drawn as boxes, and arrows between machines indicate communication channels. We use dotted arrows for insecure channels, dashed arrows for channels providing confidentiality and integrity, solid arrows indicate secure (confidential, integer, and authentic) channels, and fat solid arrows for reliable (secure and available) channels. Moreover, we label arrows only with the action name (e.g., $pay$ instead of $pay[Pay]$).

[2] A consumer may require the DRM platform to create backups of owned contents for availability reasons. Content providers instead may forbid the creation of copies at all to prevent that consumers can redistribute the content.

The general procedure towards a real world model is as follows: we first define an interface of a generic DRM system. Then we define what it means for a DRM system to be multilateral secure and derive the corresponding security requirements. Next we model an ideal system ("trusted host") providing the DRM system interface and that is secure by construction. This model already contains tolerable imperfections, which allow us to show that more realistic models are at least secure as the ideal model (see [20, 21]). We then model today's open platforms and argue why they cannot be used to realize the ideal model. Finally, we show how, in general, open platforms can be "tamed" to fulfill the desired security requirements.

**General DRM System Interface.** Figure 1) shows the general interface of a DRM system. One of its core functions is to render the view of a content. The *viewer function* $VF : (View, V_{i+1}) \leftarrow render(C, Par, V_i)$ computes a view $View$ and a resulting viewer state $V_{i+1}$ based on the content $C$, consumer input $Par$, and the current viewer state $V_i$. The exact view to be played (e.g., a chapter of a movie) is specified by $Par$. The state $V_i$ is used to store the actual licenses state, e.g., the number of remaining views.

We define a *license* attached to content $C$ as $L_C := (VF, R, V_0, Price)$. It consists of the following public[3] parameters certified by the provider: the viewer function $VF$ that exactly describes the license and thus the provider's DRM policy, the set of resources and services $R$ (e.g., required memory, CPU consumption, and an online connection to the provider) required by the viewer function to render the view, the initial state $V_0$, and a cost statement $Price$. In the model, the relation between $C$ and $L_C$ is ensured by the secure channel *input*.

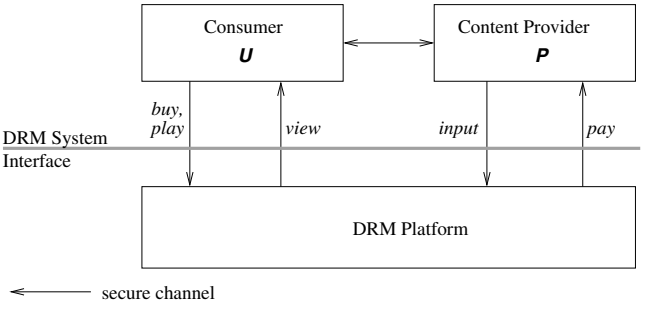



**Fig. 1.** The general interface provided by a DRM system

The general DRM system interface offers the following channels to the consumer $U$ and the content provider $P$:

[3] Consumers should be able to see what they get and whether they are willing to allow the viewer function to use the required resources.

*input*$[C, L_C]$: Using this action, the content provider $P$ sends the content $C$ and the appropriate license $L_C$ to the DRM system.
*buy*$[Pay', SP_U]$: Request to buy the content under the conditions of the appropriate license.
*pay*$[Pay]$: This action returns the payment *Pay* to the content provider.
*play*$[Par]$: Used by the consumer to view the content.
*view*$[View]$: This action returns the rendered view *View* to the consumer.

**Security Requirements.** We define a *multilateral-secure platform* to be a machine that does not violate the security requirements of the involved participants. In the context of DRM systems, the involved participants are the content provider $P$ and the consumer $U$. Thus, the security policies to be enforced are $SP_P$ and $SP_U$, and the main security requirements to be fulfilled by a multilateral-secure DRM platform are:

*Correctness:* Consumers that have correctly paid for a content can view it. In our model this can be expressed by requiring that the invocation of the *play* action has always to return the demanded view $View := render(C, Par)$, *iff* a payment with the value *Price* was send to the provider via the action *pay*.
*Security:* Consumers cannot view the content until they have send a the payment defined in $L_C$ to the provider via *pay*.
*Fair Use:* The DRM system never violates the security policy of the consumer and the provider. In our model this means that the platform shall pay for the license *only* if the consumer policy $SP_U$ allows the viewer function to use all required resources $R$.

**Adversary Model.** We only consider probabilistic polynomial-time adversaries. It has usually access to the network and can therefore eavesdrop, delete, insert, replace, and replay messages. Therefore, secure channels are modeled such that the adversary can only delete messages and/or observe the length of them (see, e.g., Figure 2).

**The Ideal Multilateral-Secure DRM System.** Figure 2 shows a DRM system specification *TH* that is multilateral-secure by construction. The channels *play*, *view*, and *pay* have to be reliable to prevent that the adversary can delete messages and thus violate the security policies. As we will later see in Section 3, a secure user interface can provide a reliable channel to the consumer. In the full paper [27], we discuss solutions to simulate reliable *pay* channels. Moreover, we consider an adversary $A$ that can use the following tolerable imperfections:

$*$*_len*[*len*]: Allows the adversary to observe the length of messages send over the channels *input*, *buy*, and *pay*. This imperfection is required since it is not practicable in real scenarios to hide the length of messages, even if they are encrypted.
$*$*_cont*[]: Since the channels *buy* and *input* are secure (and not reliable), the adversary can delete messages. This fact is modeled by this channels that allow the adversary to explicitly relay the messages sent over this channels.

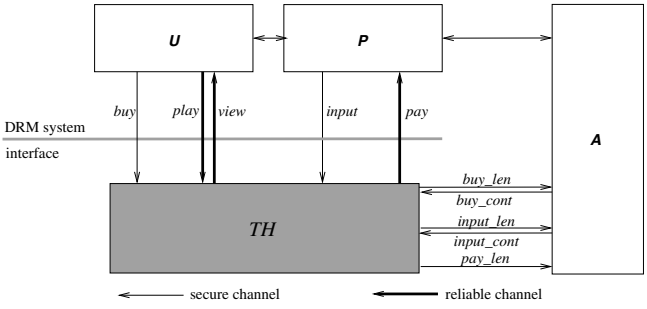



**Fig. 2.** The model of the ideal DRM system including tolerable imperfections

The additional communication channels between $U$ and $P$ and $A$ model possible information flow outside the DRM system. A state-transition model of an example implementation of the trusted host $TH$ is provided in the full version of this paper.

**Open Platforms.**Today's general-purpose computing platforms like PC's or PDA's allow their platform owners to boot arbitrary operating systems and applications. The openness of these platforms makes them inappropriate as the basis of a multilateral secure DRM platform: firstly, consumers have full control over the operating system and thus over the applications. Therefore, consumers can always bypass security policies of content providers by appropriately reconfiguring the underlying platform. Secondly, consumers have direct access to the state (the storage) of the platform allowing them to access application data and thus any content.[4] Thirdly, content providers and consumers cannot verify whether the platform is trustworthy or not, since the used communication channels do not provide mechanisms to verify the platform's current configuration (e.g., the current operating system). Fourthly, common user interfaces of general-purpose computing platforms do not provide reliable channels to the users.

Figure 3 shows a (insecure) DRM system based on currently available open platforms. Since it is not persistent, the DRM system consists of a basic open platform and a separate (persistent) machine *storage* (e.g., a harddisk). The additional interface provided by today's open basic platforms is as follows:

$boot[T_P]$ : This action allows the adversary to initialize the basic platform with any configuration $T_P$. The fact that this function is invoked over an insecure channel indicates that today consumers cannot expect their platform to be initialized in a secure or at least known configuration (see, e.g., [4, 13, 1]).

$load[S]$ and $store[S]$ : Consumers can usually access (e.g., replace, read, overwrite) the persistent storage. Thus, the adversary in our model can access the storage using these actions.

[4] Note that cryptographic mechanisms such as encryption do not help here, since the platform cannot securely store the cryptographic keys such that consumers cannot access them.

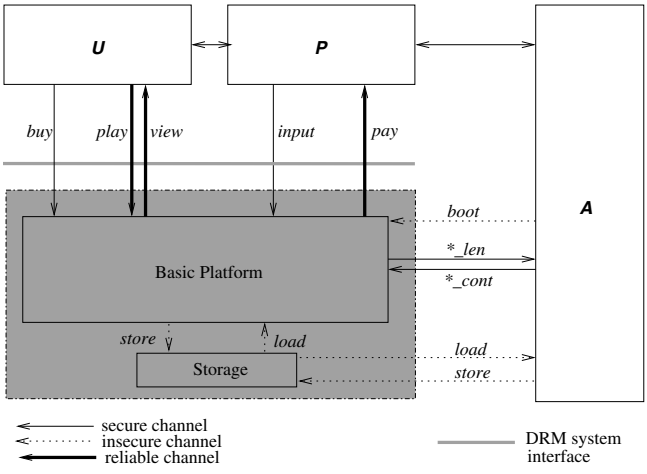



**Fig. 3.** The general model of an DRM system based on an open platform

**Building Trust on Open Platforms.** As discussed above, open platforms are not capable of enforcing security policies defined externally (e.g., by the provider). Therefore, we extend the open platform model by three trusted components providing a small set of security functions that make the overall DRM platform behave like the ideal DRM platform (see Figure 4). The new components are specified as follows:

**Trusted Boot.** Initialization of the open DRM platform is only possible via the component *trusted boot* that itself initializes the other components (basic platform, communication service, and secure storage) with $T_P$ via secure channels. Note that the adversary can still boot any platform configuration $T_P$, because the implementation of trusted boot only guarantees that the secure communication and the secure storage are initialized with the same $T_P$ that is used to initialize the basic platform.

**Secure Storage.** The secure storage provides confidentiality, integrity, authenticity, and freshness[5] of states stored by the basic platform resp. a trusted viewer.

**Secure Communication.** The secure communication component provides confidential, integer, and *type authentic* channels between the involved parties and the basic platform. Type authentic means in this context that senders can require an expected behavior (policy enforcement) of the destination machine (see full paper for a discussion of this topic).

[5] Note that it is a common attack against today's DRM systems to reset the internal state (e.g., to reset a license).

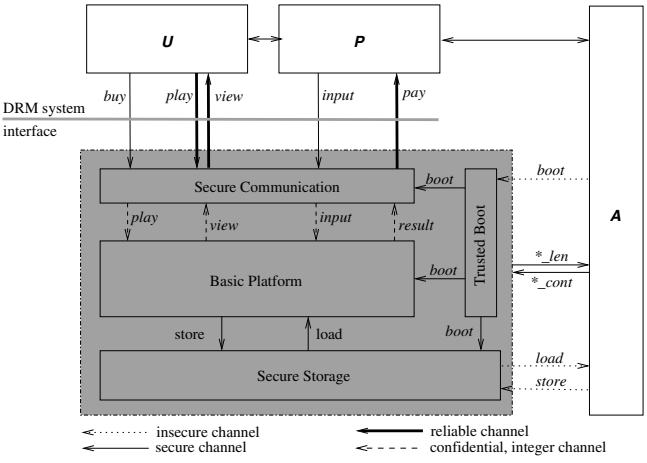



**Fig. 4.** The general model of an open DRM platform

## 3 Realization

In this section, we show how an open multilateral-secure DRM platform as modeled in the last section can be realized based on commonly available trusted computing hardware like a *Trusted Platform Module* (TPM) [29, 11].[6] For a detailed discussion about the provided functionality of TCG hardware based on an abstract model, see, e.g., [26].

In the real model, machines are implemented by *processes* executed on top of an operating system layer. This layer provides elementary resource sharing and process management functions to be able to execute several processes on top of one hardware platform and enforces on the process level the consumer's security policy. In our implementation, this secure operating system layer is realized by a security architecture [19, 24] based on a microkernel [14, 15], but the use of other platforms providing process isolation and secure inter-process communication (IPC) is also possible. Figure 5 gives an overview over the important components of the real DRM platform.[7]

In the following, we briefly describe how the functions and assumptions of the real model can be mapped to the concrete implementation of the services provided by the trusted computing base (TCB):

**Trusted Boot.** This component is realized by the Core Root of Trust Measurement (CRTM) (see [29]), a bootloader, and a TPM. The CRTM initializes a TPM-enabled platform and measures the type of the BIOS and the bootsec-

[6] Examples of TPM vendors are Atmel, Infineon, and National Semiconductor. Hardware components shipped with TPM's are, e.g., available from IBM, HP, and Intel.

[7] In fact our implementation contains of more user-level services, like resource management, memory management, device drivers, etc.

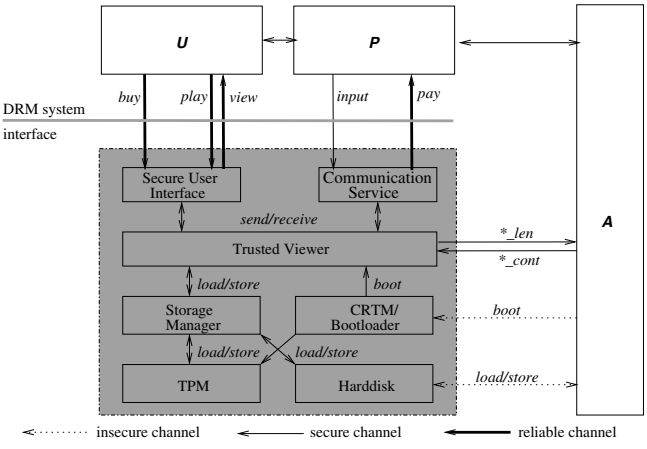



**Fig. 5.** The real model of an open DRM platform containing of services and applications managed by a microkernel

tor. Then the GRUB bootloader, we extended with TPM support [2], measures the integrity of the TCB's software components and securely stores them into the TPM using a secure channel. This allows external parties to verify whether the platform was initialized by trustworthy code using the attestation function offered by the TPM.

**Secure Storage.** This component is realized by the *storage manager* service and a TPM. The storage manager uses an encryption key created by the TPM that is bound to the platform configuration and a global counter secured by the TPM. On system shutdown, the storage manager increases the TPM counter and encrypts its current value and stores the result to the persistent storage. If the storage manager is restarted, it first loads the counter array and checks its freshness.

**Secure Communication.** This component is realized by a *secure user interface* and the *communication service*. Secure user interfaces play a security critical role, since they have direct access to security-critical information. Misuse of this information can lead to loss of privacy. While other components, like the harddisk or the network driver, can be used by tunneling (cryptographically protecting) critical information, the user interface has access to unencrypted data to be able to interact with the consumer. Therefore, it is the task of the secure user interface to provide a reliable channel between applications and consumer.

The current implementation uses the underlying OS layer to gain exclusive access to the input and output devices (e.g., video memory, appropriate interrupts, and PCI devices) to provide a secure channel to the hardware. It acts as a window manager by rendering the window labels and borders and copying the contents of the buffer to the video memory. Thus, applications can never directly access the video memory or window contents of other applications.

The task of the secure communication component is to provide a secure, type authentic channel between the DRM platform and the content provider.

The current implementation uses the sealing functionality [29] offered by TPM to ensure that only DRM platforms of a desired configuration can decrypt the content.

## 4 Fair Use

The proposed DRM platform is multilateral-secure in the sense that it enforces security policies of content providers on the application layer and those of consumers on the operating system layer. Conflicts between these policies are prevented in a fair way, since the DRM platform checks for conflicts before the content is shown and the appropriate payment is transmitted. However, multilateral security has to be provided on different abstraction levels of DRM platforms. In the following, we shortly discuss three important aspects (see full version of this paper [27].

**Secure Backup.** To ensure availability of the content, consumers should be able to create backups of their licenses without being able to perform replay attacks. Therefore, one cannot allow consumers to create backups of the secure storage containing the states $V_i$ of the viewer functions, since by importing older backups licenses can be reset or expires licenses be re-enabled. Availability of the secure storage has to be provided by alternative measures, e.g., redundancy. Although a backup of the TPM's storage root key (SRK) can be made using the maintenance function, a backup of the secure TPM counter is, to our knowledge, currently not possible. Extending the maintenance function by counter backups seems to be a first step, but further research on this topic is required.

**Private Copies.** Another important aspect is to allow consumers to make a fixed number of first-level copies of licenses. A multilateral secure DRM platform could, e.g., implement and enforce a "copy protection" flag: Whenever a new license is owned by the consumer, the DRM platform creates $n$ copies of the license that can be used as the original one, but cannot be copied. Private copies can also solve the secure backup problem.

**Property-Based Attestation.** Remote attestation allows to remotely verify the platform configuration of a TPM enabled computing system. Especially the open-source community has concerns regarding this functionality, since it allows content providers to discriminate specific software or a whole operating system. To face this problem, property-based attestation was introduced in [26, 22]. This mechanism enables us to attest properties instead of binary code. Our DRM platform could support this new paradigm by extending the communication service component.

## 5 Conclusion and Outlook

In this paper, we define the model of a multilateral-secure DRM platform where we first start with an ideal platform and then gradually approach the real world

model and give arguments for its security. The platform can be realized based on existing open platform technologies, microkernels and trusted computing hardware like a TPM. We show that a DRM platform can be carefully designed such that it fulfills security requirements of all involved parties and that policy conflicts can be detected *before* security policies are violated. Moreover, we discuss in the context of fair use further properties to be considered by the design of a multilateral secure DRM platform, namely secure backup, the creation of private copies, and property attestation.

We have implemented a prototype of the proposed components of the DRM system, e.g., a secure user interface including trusted viewer function, the secure communication service, a TPM driver, and a TPM-enabled bootloader based on GRUB. Unfortunately, TPMs (of the version 1.2) providing a secure counter are currently not available, thus our realization of the storage manager can currently not detect replay attacks.

The next steps to be done are to continue formalizing the models to be able to provide formal security proofs. Moreover, the formal models have to be extended by new functionalities such as secure backup, private copies, and property-based attestation.

## References

1. A. Alkassar, A.-R. Sadeghi, and C. Stüble. Secure object identification - or: Solving the chess grandmaster problem. In *Proceedings of the New Security Paradigm Workshow (NSPW)*, pages 77–86, 2003.
2. O. Altmeyer, A.-R. Sadeghi, M. Selhorst, and C. Stüble. Enhancing security of computing platforms with TC-technology. In S. Paulus, N. Pohlmann, and H. Reimer, editors, *Information Security Solutions Europe (ISSE 2004)*, pages 346–361. Vieweg Verlag, 2004.
3. R. J. Anderson. Security in open versus closed systems — the dance of Boltzmann, Coase and Moore. Technical report, Cambridge University, England, 2002.
4. W. A. Arbaugh, D. J. Farber, and J. M. Smith. A reliable bootstrap architecture. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 65–71, Oakland, CA, May 1997. IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Society Press.
5. W. Buhse. Implication of digital rights management for online music – a business perspective. In *ACM DRM Workshop*, pages 201–212, 2001.
6. D. L. Burk and J. E. Cohen. Fair use infrastructure for rights management systems. *Harward Journal of Law and Technology*, 15(1), 2001.
7. ElcomSoft. ebook security: theory and practice. http://www.download.ru/defcon.ppt, July 2001.
8. J. S. Erickson. Fair use, DRM, and trusted computing. *Communications of ACM*, 46(4), 2003.
9. B. L. Fox and B. LaMacchia. Encouraging recognition of fair uses in DRM systems. *Communications of ACM*, 46(4), 2003.
10. N. M. Gleb Nauvomich. Preventing piracy, reverse engineering, and tampering. *Computer*, 37(7):64 – 71, 2003.
11. T. C. Group. TPM main specification. `http://www.trustedcomputinggroup.org`, Nov. 2003. Version 1.2.

12. S. Guth. A sample DRM system. In *Digital Rights Management, Technological, Economics, Legal and Political Aspects*, pages 150 – 161, 2003.
13. N. Itoi, W. A. Arbaugh, S. J. Pollack, and D. M. Reeves. Personal secure booting. In V. Varadharajan and Y. Mu, editors, *Information Security and Privacy — 6th Australasian Conference, ACISP 2001*, volume 2119 of *Lecture Notes in Computer Science*, pages 130–144, Sydney, Australia, July 2001. Springer-Verlag, Berlin Germany.
14. J. Liedke. On u-kernel construction. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP'95)*, Copper Mountain Resort, Colorado, Dec. 1995. Appeared as ACM Operating Systems Review 29.5.
15. J. Liedke. Towards real micro-kernels. *Communications of the ACM*, 39(9), 1996.
16. N. A. Lynch and M. R. Tuttle. An introduction to Input/Output automata. *CWI-Quarterly*, 2(3):219–246, Sept. 1989.
17. D. K. Mulligan. Digital rights management and fair use by design. *Communications of the ACM*, 46(4):31–33, 2003.
18. National Research Council. *The Digital Dilemma, Intellectual Property in the Information Age*. National Academy Press, 2000.
19. B. Pfitzmann, J. Riordan, C. Stüble, M. Waidner, and A. Weber. The PERSEUS system architecture. Technical Report RZ 3335 (#93381), IBM Research Division, Zurich Laboratory, Apr. 2001.
20. B. Pfitzmann, M. Schunter, and M. Waidner. Cryptographic security of reactive systems. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 32, 2000. Workshop on Secure Architectures and Information Flow, Royal Holloway, University of London, December 1 - 3, 1999.
21. B. Pfitzmann, M. Schunter, and M. Waidner. Provably secure certified mail. Research Report RZ 3207 (#93253), IBM Research, Aug. 2000.
22. J. Poritz, M. Schunter, E. V. Herreweghen, and M. Waidner. Property attestation—scalable and privacy-friendly security assessment of peer computers. Technical Report RZ 3548, IBM Research, May 2004.
23. W. Rosenblatt, W. Trippe, and S. Mooney. *Digital Rights Management: Business and Technology*. Jon Wiley & Sons, 2001.
24. A.-R. Sadeghi and C. Stüble. Bridging the gap between TCPA/Palladium and personal security. Technical report, Saarland University, Germany, 2003.
25. A.-R. Sadeghi and C. Stüble. Taming "trusted computing" by operating system design. In *Information Security Applications*, volume 2908 of *Lecture Notes in Computer Science*, pages 286–302. Springer-Verlag, Berlin Germany, 2003.
26. A.-R. Sadeghi and C. Stüble. Property-based attestation for computing platforms: Caring about properties, not mechanisms. In *The 2004 New Security Paradigms Workshop*, Virginia Beach, VA, USA, Sept. 2004. ACM SIGSAC, ACM Press.
27. A.-R. Sadeghi and C. Stüble. Towards multilateral-secure drm platforms. Technical report, Horst Görtz Institute, Ruhr-University Bochum, January 2005.
28. P. Samuelson. DRM, AND, OR, VS, The Law. *Communications of ACM*, 46(4):41 – 45, 2003.
29. Trusted Computing Platform Alliance (TCPA). Main specification, Feb. 2002. Version 1.1b.

# Hiding Data in Binary Images

Chin-Chen Chang[1], Chun-Sen Tseng[1], and Chia-Chen Lin[2]

[1] Department of Computer Science and Information Engineering,
National Chung Cheng University, Chiayi 621,
Taiwan, R.O.C.
{ccc, tcs92}@cs.ccu.edu.tw
[2] Department of Computer Science and Information Management
Providence University,
Taichung 433, Taiwan, R.O.C.
mhlin3@pu.edu.tw

**Abstract.** This paper presents a novel scheme for embedding secret data into a binary image. In Tseng et al.'s scheme, a random binary matrix and a weight matrix are used as the secret keys to protect the secret information. In our scheme, we use a serial number matrix instead of a random binary matrix to reduce computation cost and to provide higher security protection on hidden secret data than Tseng et al. do. Given a cover image divided into blocks of m×n pixels each, our new scheme can hide $\lfloor \log_2(mn+1) \rfloor$ bits of hidden data with one modified bit at most in each block in the cover image. In addition, the hiding capacity of our new scheme offers is as large as that of Tseng et al.'s scheme, but we support higher stego-image quality than Tseng et al.'s scheme does.

## 1 Introduction

The rapid advancement of the Internet technology has made the Internet the most popular channel for digital data exchanges. Generally speaking, digital data transmitted on the Internet can be in the form of text messages, images, or audio and video files. Despite the convenience the Internet offers for data exchange, one major problem occurs. That is, the data on the Internet is easily tampered with and stolen by attackers during the transmission. In order to deal with this problem, two strategies have been proposed: cryptography and steganography [1-16]. The former strategy usually transfers data into a set of seemingly meaningless codes. Only the authorized user can transform it back to its original form by using some secret information. Many famous encryption schemes, such as RSA, DES, and so on, are already widely accepted commercially. However, the meaningless appearance may be a clue to an unauthorized user and shows that there might be something interesting hidden inside.
The other strategy, called steganography or data hiding, hides a secret message in a cover media to avoid arousing attackers' attention. For example, the outline of a computer motherboard can be embedded into Vincent van Gogh's famous painting "The Starry Night" to fool attackers so that it can be transmitted on the Internet unnoticed. The concept of steganography is similar to the concept of camouflage, which is

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 338 – 349, 2005.

used by animals to protect them from being attacked. Generally, the objective of steganography is to hide a secret message well enough so that unauthorized users will not even be aware of its existence. Several steganographic schemes have been developed to solve the privacy problem [1-16]. In general, the steganography approach can be classified into three categories. In the first category, the schemes hide a secret message in the spatial domain of the cover image [3, 10, 13, 14, 16]. In Lee and Chen's scheme [16], the least significant bit (LSB) of each pixel in the cover image is modified to embed the secret message. In Wang et al.'s scheme [14], the optimal substitution of LSB is exploited. In addition, Chung et al. offered a singular value decomposition (SVD)-based hiding scheme [10], and Tsai et al. exploited the bit plane of each block truncation coding (BTC) block to embed the secret message [13]. In the second category, the schemes embed a secret message into a transformed cover image [1, 2, 4, 8]. Several transformation functions, such as the discrete cosine transformation (DCT) and the discrete wavelet transformation (DWT), are widely used. For example, in Chang et al.'s scheme [1], the middle frequency coefficients of the DCT transformed cover image are employed to embed the secret message. In addition, the quantization table of JPEG is modified to protect the embedded secret message. In Kobayashi et al.'s scheme [2], a secret message is hidden in the JPEG encoded bit streams. In the third category, several schemes that work on index-based cover images [5, 7, 11]. In fact, index-based images such as vector quantization (VQ)-based images, color quantization (CQ)-based (palette-based) images, are commonly used.

However, most cover images of the above schemes are gray-level images or color images. The binary image is not often used to be a cover media [6, 9, 12, 15, 17, 18]. The major reason is that the modification is easily detected when a single pixel is modified in a binary image. In [6], Chen et al. decomposed an image into many blocks first. Then, they divided each block into several non-overlapping subgroups, called partitions. Finally, they come up with the characteristic value of each partition to decide where to hide the secret data. However, they can only conceal just one bit in a 4×4 block. To improve the hiding capacity, Tseng et al.'s scheme divides a cover image into many non-overlapping blocks and hides data in each block [15]. They also generated two matrices, a binary matrix and a weight matrix, to decide which bits need to be modified so that the secret information can be hidden and the good image quality of stego-image can be achieved. Given an m×n block from the cover image, Tseng et al.'s scheme can conceal bits of data in a block by changing two pixels at most.

Since changing any pixel in a binary image can cause a detectable change in the cover binary image, it is important to reduce the number of modified pixels. Otherwise, the steganography is easily detected by the human visual system. With our new scheme, we can hide as many bits as Tseng et al.'s scheme can, and there is only one pixel at most that is modified in each block. The image quality of the cover image is thus improved, and the hidden information is well protected.

The remaining text of this paper is organized as follows. In Section 2, we shall introduce our proposed serial matrix first and then present our proposed scheme. In Section 3, we shall analyze the security of the proposed scheme. Section 4 will discuss our experimental results and compare our performance with Tseng et al.'s. Finally, Section 5 presents the conclusions.

## 2 Proposed Data Hiding Scheme for Binary Images

To maintain good image quality of stego-image and to reduce the number of modified pixels when hiding secret message in a block, we propose a serial number matrix here to decide which pixel needs to be modified. The proposed serial number matrix is our key technique. Therefore, we will introduce the proposed serial number matrix in Subsection 2.1 and then proves its function in Subsection 2.2. At last, we will present our proposed scheme in Subsection 2.3

### 2.1 The Proposed Serial Number Matrix

In our proposed scheme, we try to embed $\lfloor \log_2(mn+1) \rfloor$ bits into an $m \times n$ block in a cover image by changing one pixel at most in each block. In other words, using our proposed data hiding scheme, the better case is that none of the pixels needs to be modified, and the worst case of the two is that one pixel in a block needs to be changed. Whichever the case, we always need a good selection mechanism to pick out the pixels to be modified so that the image distortion can be reduced while keeping high stego-image quality and high hiding capacity. This is quite a challenging task. In this paper, we offer a serial number matrix $O$ as our selection mechanism, which can help us to change only one pixel at most in a block. In our proposed serial number matrix $O$ sized $m \times n$, at most $(m \times n-1)$ non-duplicate integers appear. Assume $b$ is a number that shows up in the serial number matrix $O$, and the values in different positions of the serial number matrix $O$ can be duplicated. Let $H_j$ be the value of the jth secret bit, $j$=1, 2,…, $\lfloor \log_2(mn+1) \rfloor$ . According to the proposed serial number matrix $O$, the general hiding equation is as follows.

$$H_j = \sum_{i \in N_j} p_i(x) \text{ ,} \tag{1}$$

$O =$

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 1 |

(a)

$O_b =$

| | | | |
|---|---|---|---|
| 0001 | 0010 | 0011 | 0100 |
| 0101 | 0110 | 0111 | 1000 |
| 1001 | 1010 | 1011 | 1100 |
| 1101 | 1110 | 1111 | 0001 |

(b)

$A =$

| | | | |
|---|---|---|---|
| 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |

(c)

$A' =$

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |

(d)

**Fig. 1.** (a) An example of the proposed serial number matrix $O$ sized 4×4, (b) the binary representation of $O$ named $O_b$, (c) an example binary cover image $A$ sized *4×4*, and (d) a stego-image $A$' that is the embedding result of the cover image $A$

where $p_i(x)$ is the pixel value of binary cover image $A$, which corresponds to serial number $x$ in position $i$ in the serial number matrix $O$ when the $j$th bit value of the binary representation of $x$ is 1, where $_{i \in N_j}$, $1 \le x \le \lfloor \log 2(mn+1) \rfloor$. $N_j$ is the serial number set which is defined by proposed rule with corresponding $H_j$, we describe the rule for grouping serial number into each $N_j$ in the following example.

In order to visualize how the above matrix works, let us take a 4×4 image block for example. The proposed serial number matrix O sized 4×4 contains 15 non-duplicate integers shown in Fig. 1(a). The numbers that show up in the serial number matrix are from 1 to 15, and the value in the position 1 in the serial number matrix O is 1, which is the same as the value in position 16.

Assume that we want to hide 4 bits in the binary cover image $A$ shown in Fig. 1(c). According to the proposed serial number matrix $O$ shown in Fig. 1(a) and the general hiding equation shown in Equation (1), four hiding equations are constructed as follows.

$$H_1=p_1(1)+p_3(3)+p_5(5)+p_7(7)+p_9(9)+p_{11}(11)+p_{13}(13)+p_{15}(15)+p_{16}(1). \quad (2)$$
$$H_2=p_2(2)+p_3(3)+p_6(6)+p_7(7)+p_{10}(10)+p_{11}(11)+p_{14}(14)+p_{15}(15). \quad (3)$$
$$H_3=p_4(4)+p_5(5)+p_6(6)+p_7(7)+p_{12}(12)+p_{13}(13)+p_{14}(14)+p_{15}(15). \quad (4)$$
$$H_4=p_8(8)+p_9(9)+p_{10}(10)+p_{11}(11)+p_{12}(12)+p_{13}(13)+p_{14}(14)+p_{15}(15). \quad (5)$$

Based on the general hiding equation shown in Equation (1), each value of $H_j$ (for $j$ = 1, 2, 3, 4) is either odd or even. Besides, we can make sure there will be any two of hiding equations share at least one $p_i(x)$, any three of hiding equations share at least one $p_i(x)$, and any four of hiding equations share at least one $p_i(x)$. Still, each of these four equations also has at least one $p_i(x)$ which will not appear in any other equations. For example, in Equation (2), the serial numbers in positions 1 and 16 in the serial number matrix $O$ are all 1, where the *1*st bit value of their binary representations are 1, and the remaining bit values of their binary representations are 0. Therefore, the corresponding values $p_1(1)$ and $p_{16}(1)$ only appear in Equation (2). The serial number of position 3 in serial number matrix $O$ is 3, where both the first and second bit values of its binary representation are 1. Therefore, its $p_3(3)$ appear in Equations (2) and (3). To sum up, we can modify only one component in each equation to adjust each $H_j$ from odd to even or from even to odd using the above four equations. Based on the above arrangement, first, we can get the corresponding $h_j$ for each $H_j$ by using Equation (6). Then, we can compare $h_j$ with the $j$th bit value of the secret data $s_j$ to generate $R_j$ using the principle listed in Equation (7). Next, we put $R_j$, $j$=4, 3, 2 and 1 together to generate a stream $R$ and then convert the stream $R$ into its decimal representation. Finally, we can obtain the modification position in the block.

$$h_j = H_j \bmod 2, \text{ for } j = 1, 2, 3, \text{ and } 4, \quad (6)$$
$$R_j=0, \text{ if } h_j=s_j$$
$$R_j=1, \text{ otherwise.} \quad (7)$$

With our proposed serial number matrix $O$, the largest hiding capacity of a block sized $m \times n$ is the same as the number of hiding equations generated by the serial number matrix $O$. In other words, when a 4×4 serial number matrix $O$ contains 15 non-duplicate integers, we can construct four hiding equations according to the general hiding equation shown in Equation (1), and that means the largest hiding capacity of each block is 4 bits.

To describe the function of our proposed serial number matrix $O$, a simple example is presented as follows. Give a block $A$ as shown in Fig. 1(c). Assume that the secret data is "1001". Fig. 1(a) is an example of our serial number matrix $O$, and its binary representation is shown in Fig. 1(b). Table 1 shows which pixel needs to be modified, and Fig. 1(d) is the embedding result of block $A$. Let us take the first bit of the secret data for example. First, we can generate equation $H_1$ by using Equation (1) and then get the result $H_1$ ($H_1$=$p_1(1)$+$p_3(3)$+$p_5(5)$+$p_7(7)$+$p_9(9)$+$p_{11}(11)$+$p_{13}(13)$+$p_{15}(15)$+$p_{16}(1)$ =1+0+0+0+1+1+0+0+0+0=3). We can obtain $h_1$ =1 using Equation (6), and $R_1$ =0 according to Equation (7). The same operations are conducted to obtain $R_2$, $R_3$ and $R_4$, respectively, shown in Table 1. Finally, we put $R_j$, for $j$=4, 3, 2 and 1 together to generate an $R$ stream as "1100". After converting the $R$ stream into its decimal representation, we can obtain $12_{(10)}$. It means that we only need to modify position 12 from 0 to 1 in block $A$ to hide the secret data "1001". The embedding result is $A'$ shown in Fig. 1(d).

**Table 1.** The secret data, modified $H_j$'s and their corresponding $h_j$'s and $R_j$'s, and the modification position given by the serial number matrix $O$

| Block | $s_4,s_3,s_2,s_1$ | $H_4,H_3,H_2,H_1$ | $h_4,h_3,h_2,h_1$ | $R_4,R_3,R_2,R_1$ | Modification Position *(MP)* |
|---|---|---|---|---|---|
| $A$ | 1, 0, 0, 1 | 4, 3, 4, 3 | 0, 1, 0, 1 | 1, 1, 0, 0, | $12=2^3*1+2^2*1+2^1*0+2^0*0$ |

Since the modification position is 12, $p_{12}(12)$ appears in $H_3$ and $H_4$ simultaneously. After modification, the value of the modified $H_j$'s are 3, 4, 4, 5, respectively, where $j$=4, 3, 2 and 1. The new $h_j$'s and $R_j$'s are generated according to Equations (6) and (7). Please note that each new $h_j$ is the same as its corresponding secret data $s_j$, and no pixel needs to be changed. The modified $H_j$'s and new $h_j$'s and $R_j$'s, where $j$=4, 3, 2 and 1, are listed in Table 2.

**Table 2.** The secret data, modified $H_j$'s, their corresponding $h_j$'s and $R_j$'s, and the modification position given by the serial number matrix $O$

| Block | $s_4,s_3,s_2,s_1$ | $H_4,H_3,H_2,H_1$ | $h_4,h_3,h_2,h_1$ | $R_4,R_3,R_2,R_1$ | Modification Position *(MP)* |
|---|---|---|---|---|---|
| $A$ | 1, 0, 0, 1 | **5**, **4**, 4, 3 | **1**, **0**, 0, 1 | **0**, **0**, 0, 0, | $0=2^3*0+2^2*0+2^1*0+2^0*0$ |

## 2.2 Verification of Our Proposed Serial Number Matrix

In the previous subsection, we have illustrated how our serial number matrix $O$ functions. In addition, we claim that the proposed serial number matrix $O$ can guarantee that only one pixel at most is needed in a block to be modified to embed $r$ bits, and $r$ is the number of hiding equations generated by the proposed general hiding equation and serial number matrix $O$. For example, given a serial number matrix $O$ sized 4×4 with 15 non-duplicate integers, the serial number matrix $O$ can generate four hiding equations at most, which means the largest hiding capacity of each block in the cover image is 4 bits. Before we give more details as to our proposed hiding scheme based on the serial number matrix $O$, we will try to prove our claim in this subsection.

We claim that if there are $2^r$-1 elements in the serial number matrix $O$ that are non-duplicate, then as many as $r$ hiding equations can be generated according to our proposed general hiding equation mentioned in the previous subsection. Besides, each of hiding equations has at least one element of a block which will not appear in any other equations, any two of hiding equations share at least one element of a block, any three of hiding equations share at least one element of a block, and so on. Please refer to Equations (2)-(5).

In this case, do we only need to explore $2^r$-1 pixels of a block in a cover image if we want to hide $r$ bits of secret data into a block? Our proof is quite straightforward. We list all the possible modification solutions to check whether the maximum number of modification pixels is $2^r$-1.

$$\sum_{c=1}^{r} C_c^r = C_1^r + C_2^r + ... + C_{r-1}^r + C_r^r \cdot \tag{8}$$

Here, $c$ stands for the number of pixels to be modified to hide the secret data. In other words, $c$ is also the number of the modified hiding equations. Since $c$ equals 0, which means that there is no hiding equation, $C_0^r$ is not included in Equation (8). $C_1^r$ means only one $h_j$, where $1 \le j \le r$, is different from its relative secret bit $s_j$, $1 \le j \le r$, and only one hiding equation needs to be modified. Since each of hiding equations has at least one element of a block, which will not appear in any other equations, in this case, we only need to modify one pixel of a block in the cover image to achieve our goal. $C_2^r$ means two $h_j$'s, where $1 \le j \le r$, are different from their relative secret bits $s_j$'s, where $1 \le j \le r$, and two hiding equations need to be modified. According to the designing principle of our hiding equations, any two of the hiding equations share one common pixel. Therefore, we can still change one pixel to modify the values of *h's* and then achieve our goal. In all the other cases, the same logic applies. Finally, we come to this conclusion: If we want to hide $r$ bits in a block, we only need to explore $2^r - 1$ ($\sum_{c=1}^{r} C_c^r = 2^r - 1$) pixels at most in our hiding equations and change one pixel at most in any case. Since only $2^r - 1$ pixels need to be explored in each block, the maximum amount of embedded bits is $\lfloor \log_2(mn+1) \rfloor$ in a block whose size is $m \times n$ by modifying one pixel at most.

### 2.3 The Proposed Data Hiding Scheme for Binary Images

In this subsection, we shall first illustrate how to apply the proposed serial number matrix O to hide secret data in a binary image. Then, the extraction procedure will be presented.

#### A. The Embedding Procedure

Our proposed scheme not only uses a binary matrix $K$ as the secret key but also uses a serial number matrix $O$ to increase the number of the candidate modification positions. Therefore, the security of the embedded data is enhanced. The inputs to our scheme are as follows.

1) $I$ is a host binary image (i.e., bitmap) to be modified to embed secret data. Here, $I$ is partitioned into non-overlapped blocks $B_i$ sized $m \times n$. For simplicity, we assume that the size of $I$ is a multiple of $m \times n$.
2) $K$ is a secret key shared between the sender and the receiver. It is a randomly selected bitmap sized $m \times n$ .
3) $r$ is the number of bits to be embedded in each $m \times n$ block of $I$, which is predetermined by the sender and receiver. The value of $r$ satisfies $2^r - 1 \le mn$ .
4) $O$ is a serial number matrix shared between the sender and the receiver. It contains $2^r - 1$ non-duplicate integers at most. $O_b$ is the binary representation of $O$.
5) $S$ is critical information consisting of $kr$ bits to be embedded in $I$, where $k$ is the number of $m \times n$ blocks in $I$. $S$ is divided into $k$ groups, where each group consists of $r$ secret bits. $s_{ij}$ is the $j$th secret data embedded in the image block $B_i$. The order for each secret data in a group is from right to left. For example, assume the secret data is 0101 for block $B_1$, then their order is $s_{11}$=1, $s_{12}$=0, $s_{13}$=1 and $s_{14}$=0.

Assume that the size of $K$ and $O$ is 4×4. Let's consider a 4×4 image block $B_i$, which is a part of the host image $I$. The purpose is to show how to embed 4 ($r$=4) bits of data in $B_i$. Suppose we have the following inputs, shown in Fig. 2:

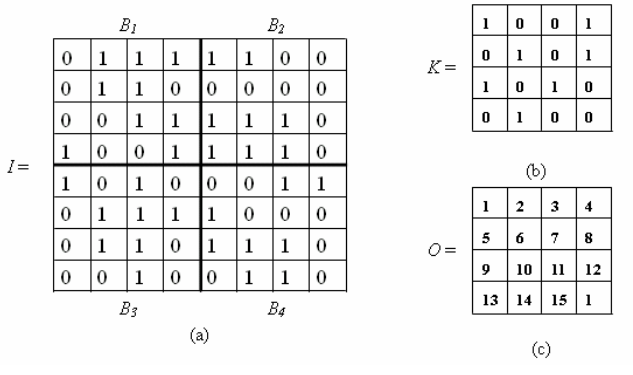



**Fig. 2.** (a) An example of binary cover image $I$ sized 16×16, (b) A secret key $K$ sized 4×4, and (c) A decimal serial number matrix $O$ sized 4×4

The proposed embedding procedure takes the following six steps to process each block of cover image $I$.

**Step 1.** Compute $C_i = B_i \oplus K, 1 \le i \le k$ .

**Step 2.** Generate $r$ hiding equations according to Equation (1), and then obtain $H_{ij}$ of $C_i$, where $1 \le j \le r$, and $1 \le i \le k$ .

**Step 3.** Compute $H_{ij} \bmod 2$ of $C_i$ to get relative $h_i$, where $1\ 1 \le j \le r$, and $1 \le i \le k$ .

**Step 4.** Compare $h_{ij}$ and $s_{ij}$ of $C_i$. If they are identical, $R_{ij}$ is set to be 0; otherwise, $R_{ij}$ is set to be 1, where $1 \le j \le r$, and $1 \le i \le k$ .

**Step 5.** Concatenate $R_{ij}$, where $1 \le j \le r$, to generate a $R_i$ stream, and convert an $R_i$ stream into the decimal representation as the modification position $MP_i$ for $B_i$.

**Step 6.** Modify the pixel value of position $MP_i$ of block $B_i$ from 0 to 1 or 1 to 0.

Steps 1 to 6 are repeated until all blocks of the cover image $I$ have been processed. At last, a stego-image $I'$ is generated and is sent to the receiver. Then, to extract the hidden data, the secret key $K$ and serial number matrix $O$ are sent to the receiver in advance through a secure channel. A simple example is presented as follows using the cover image $I$, secret key $K$, and serial number matrix $O$ shown in Fig.2. Assume that the secret data is 0010 0000 1011 0110. We can obtain $I'$ shown in Fig. 3 after performing $B_i \oplus K$, $1 \le i \le 4$.

$I'$ =

| $C_1=B_1\oplus K$ | | | | $C_2=B_2\oplus K$ | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| $C_3=B_3\oplus K$ | | | | $C_4=B_4\oplus K$ | | | |

**Fig. 3.** The result of computing $B_i \oplus K$

Then, we can generate four hiding equations of block $C_1$ according to Equation (1) as follows. The remaining blocks also need to generate their data hiding equations to obtain $H_{i1}$, $H_{i2}$, $H_{i3}$, $H_{i4}$, where $i$=2, 3, and 4.

$H_{11}=p_1(1)+p_3(3)+p_5(5)+p_7(7)+p_9(9)+p_{11}(11)+p_{13}(13)+p_{15}(15)+p_{16}(1)$.
$H_{12}=p_2(2)+p_3(3)+p_6(6)+p_7(7)+p_{10}(10)+p_{11}(11)+p_{14}(14)+p_{15}(15)$.
$H_{13}=p_4(4)+p_5(5)+p_6(6)+p_7(7)+p_{12}(12)+p_{13}(13)+p_{14}(14)+p_{15}(15)$.
$H_{14}=p_8(8)+p_9(9)+p_{10}(10)+p_{11}(11)+p_{12}(12)+p_{13}(13)+p_{14}(14)+p_{15}(15)$.

$\bar{I}$ =

| $\bar{B}_1$ | | | | $\bar{B}_2$ | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | **1** | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| **1** | 1 | 1 | 0 | 1 | **0** | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| $\bar{B}_3$ | | | | $\bar{B}_4$ | | | |

**Fig. 4.** A stego-image $\bar{I}$ generated by our proposed embedding procedure

After conducting Steps 3 to 6, we can generate the relative parameters of block $C_i$, shown in Table 3. From Table 3, we observe that $MP_1$ is 10 for block $B_1$, which means we only need to modify the pixel value of position 10 in block $B_1$ from 0 to 1 to hide secret data 0010 in block $B_1$. $MP_2$ is 0 for $B_2$, which means no pixel needs to be modified to embed secret data 0000. After modifying the pixel values in cover image $I$ according to the modification positions ($MP$s) listed in Table 3, a stego-image $\bar{I}$ is generated as shown in Fig. 4.

## B. The Extracting Procedure

In Fig. 4, $\bar{I}$ is the stego-image with the secret data already hidden in by the sender, and the receiver can only extract the secret data from it by using our proposed extracting procedure. Basically, the receiver has the same secret key $K$. S/he can generate $\bar{I}'$ (shown in Fig. 5) by computing $\bar{I} \oplus K$ and then extract the secret data by performing Steps 2 and 3 as described in our embedding procedure.

$\bar{I}'$ =

| $\bar{C}_3=\bar{B}_3\oplus K$ | | | | $\bar{C}_3=\bar{B}_3\oplus K$ | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| $\bar{C}_3=\bar{B}_3\oplus K$ | | | | $\bar{C}_3=\bar{B}_3\oplus K$ | | | |

**Fig. 5.** An image $\bar{I}'$ which is result of performing $\bar{I} \oplus$K

To illustrate our extracting procedure, we use the above example to explain how to extract secret data from block $\bar{B}_1$ of stego-image $\bar{I}$ shown in Fig. 4. After receiving the stego-image $\bar{I}$, the receiver uses the shared secret key $K$ to perform XOR computation on block $\bar{B}_1$ and generates the result $\bar{I}'$ shown in Fig. 5. Next, s/he constructs four data hiding equations according to the predetermined parameter $r$ and the shared serial number matrix $O$, and learns that the values of $H_{14}'$, $H_{13}'$, $H_{12}'$, *and* $H_{11}'$ are 6, 4, 5, and 6, respectively. S/he can also obtain the values of $h_{14}'$, $h_{13}'$, $h_{12}'$, and $h_{11}'$, in this case 0, 0, 1, and 0, respectively, by performing $H_{i1}$ mod 2, where $i$ equals 1. At last, the receiver links the above four $h_{i4}'$, where $i$ equals 1, to get the extracted secret data related to $\bar{B}_1$ of the stego-image. The remaining secret data are extracted following the same procedure. The intermediate results and the extracted data for each block of the stego-image are listed in Table 4.

**Table 4.** The results of extracting data from stego-image $\bar{I}$

| Block | $H_{i4}', H_{i3}', H_{i2}', H_{i1}'$ | $h_{i4}', h_{i3}', h_{i2}', h_{i1}'$ | Extracted Secret Data ($s_{i4}, s_{i3}, s_{i2}, s_{i1}$) |
|---|---|---|---|
| $\bar{C}_1$ | 6,4,5,6 | 0,0,1,0 | 0,0,1,0 |
| $\bar{C}_2$ | 4,4,4,2 | 0,0,0,0 | 0,0,0,0 |
| $\bar{C}_3$ | 3,4,5,3 | 1,0,1,1 | 1,0,1,1 |
| $\bar{C}_4$ | 2,3,4,4 | 0,1,1,0 | 0,1,1,0 |

## 3 Security Analysis of the Proposed Scheme

In Fig. 1 (c), we select the first 15 positions in the serial number matrix $O$ and assign increasing integers to them. For positions 1 and 16 of $O$, they are assigned the number "1", and their corresponding pixels in $I'$ are the modification candidates when users need to modify $H_{i1}$ from 0 to 1 or 1 to 0 to hide secret data. To enhance the security of the serial number matrix $O$ and make the data hiding equations more complex and unpredictable, we can assign a different value, such as 10, to the 16th position of matrix $O$ rather than 1. Assume that we give the serial number 10 to the 16th position, both $H_{i2}$ and $H_{i4}$ have to contain the value of the 16th position of matrix $O$, and $H_{i1}$ will only contain eight rather than nine elements, according to the principle presented in Equation (1). Although the serial number matrix $O$ has been modified, it still contains 15 non-duplicate integers, and therefore, we can conceal at least $r$ secret data in a block and maintain very good stego-image quality. In addition, we can also design various versions of the serial number matrix $O$ with different numbers of non-duplicate integers.

Our proposed serial number matrix $O$ can have numerous variants. This characteristic offers the sender multiple modification positions, and thereby enhances the security of the hidden secret data. In general, if we want to embed $r$ bits into an $m \times n$ block, the number of candidate modification positions is $C_{2^r-1}^{mn} \times (2^r-1)! \times (2^r)^{mn-(2^r-1)}$, where $C_{2^r-1}^{mn}(2^r-1)!$ means that we randomly select $2^r-1$ elements out of the block and assign non-duplicate increasing integers to them. The remaining $m \times n - (2^r-1)$ positions of the block are faced with two possible cases, one is to be assigned with arbitrary values, which are identical to the values of other positions, and the other is not used for hiding secret data. Therefore, the value of each position has $2^r$ candidates. In [15], Tseng et al.'s scheme uses a weight matrix to represent the embedded data. The function of their weight matrix is like our serial number matrix $O$. Based on a weight matrix, the number of candidate modification positions is $C_{2^r-1}^{mn} \times (2^r-1) \times (2^r-1)^{mn-(2^r-1)}$.

Please note that our proposed scheme allows users to appoint some positions of a block in the cover image as unchangeable positions those are not used to hide secret data. In contrast, in Tseng et al.'s scheme, each position of a block in the cover image is considered changeable and is used to hide secret data. Therefore, our hiding strategy is more complex, and provides better protection for the hidden data than Tseng et al.'s scheme does.

## 4 Experimental Results

In Tseng et al.'s scheme, they make two slight enhancements when they conduct their experiments to improve the image quality of the stego-image [15]. One is that pixels around black-and-white margins are modified at a higher priority, and the other is that no secret bits are concealed in an entirely black or white block. Their reason for taking those steps is that a block Bi,j may not be entirely black or white, but it could become completely black or white after some secret data are hidden in it. Their experimental results have confirmed that their enhancements are effective. Therefore, in our experiments, we also followed the same strategies to obtain good stego-image quality. Besides, we used three different types of images, all sized 512×512, as our host images, including an English text image, a Chinese text image, and the "Baboon" image, to hide the same amount of secret data as Tseng et al. did in their experiments (i.e. bits in an m×n block).

Table 6 presents the *PSNR* values of all the stego-images by our scheme and Tseng et al.'s scheme, respectively. As the results show, the *PSNR* values of our scheme are always higher than those of Tseng et al.'s scheme when the hiding capacities are the same. For both our new scheme and Tseng et al.'s scheme, the probability of occurrence of the case where *r* bits are embedded into a block without altering any pixel is $\frac{1}{2^r}$. Except for that case, our scheme always changes one bit at most in a block. By contrast, Tseng et al. may need to modify two bits in a block.

**Table 6.** Stego-image *PSNR*s generated by Tseng et al.'s scheme and our scheme

| Block size / Host images | 16×16 | | 32×32 | |
|---|---|---|---|---|
| | Our scheme | Tseng et al's scheme | Our scheme | Tseng et al's scheme |
| English text image | 55.49 dB | 53.51 dB | 69.31 dB | 67.08 dB |
| Chinese text image | 55.60 dB | 53.89 dB | 69.31 dB | 67.11 dB |
| Baboon | 56.78 dB | 54.69 dB | 69.59 dB | 67.40 dB |

## 5 Conclusions

In this paper, we have offered a novel data hiding scheme to hide secret data in binary images. To provide a more complex hiding strategy than Tseng et al.'s, in our scheme, a shared key matrix and various versions of our proposed serial number matrix are used to decide in which candidate modification positions the $\lfloor \log_2(mn+1) \rfloor$ bits of secret data are to be hidden. We alter only one pixel value at most when hiding r secret data in an *m*×*n* image block, and the experimental results have confirmed that our stego-image quality is indeed better than that of Tseng et al.'s scheme. In the three cases in our experiments, the average PSNR value our scheme gave is greater than that given by Tseng et al.'s scheme. To sum up, our scheme enhances the security of the hidden secret data with simple operations; meanwhile, it also effectively improves the stego-image quality.

## References


1. C. C. Chang, T. S. Chen and L. Z. Chung, "A Steganographic Method Based upon JPEG and Quantization Table Modification," *Information Sciences*, Vol. 141, pp.123-138 (2002).
2. H. Kobayashi, Y. Noguchi and H. Kiya, "A Method of Embedding Binary Data into JPEG Bitstreams," *IEICE Transactions*, Vol. J83-D2, No. 6, pp.1469-1476 (2000).
3. J. Fridrich, M. Goljan, and R. Du, "Detecting LSB Steganography in Color and Gray-scale Images," *IEEE Multimedia*, Vol. 8, Issue 4, Oct.-Dec. pp. 22-28 (2001).
4. M. Iwata, K. Miyake, and A. Shiozaki, "Digital Steganography Utilizing Features of JPEG Images," *IEICE Transactions on Fundamentals*, Vol. E87-A, No. 4, April, pp. 929-936 (2004).
5. C. H. Tzeng, Z. F. Yang, and W. H. Tsai. "Adaptive Data Hiding in Palette Image by Color Ordering and Mapping with Security Protection," *IEEE Transactions on Communications*, Vol. 52, No. 5, May, pp. 791- 800 (2004).
6. J. Chen, T. S. Chen, M. W. Cheng, "A New Data Hiding Scheme in Binary Image," in *Proc. Fifth Int. Symp. on Multimedia Software Engineering*. Proceedings, pp. 88-93 (2003).
7. J. Fridrich, "A New Steganographic Method for Palette-Based Images," in *Proc. of the IS&T PICS Conference*, Savannah, Georgia, April pp.285-289 (1998).
8. J. Spaulding, H. Noda, M. N. Shirazi and E. Kawaguchi, "BPCS Steganography Using EZW Lossy Compressed Images," *Pattern Recognition Letters*, Vol. 23, No. 13, pp.1579-1587 (2002).
9. J. Zhao and E. Koch, "Embedding Robust Labels into Images for Copyright Protection," in *Proc. Int. Conf. Intellectual Property Rights for Information Knowledge*, New Techniques, Munich, Germany, pp. 242-251 (1995).
10. K. L. Chung, C. H. Shen and L. C. Chang, "A Novel SVD- and VQ-based Image Hiding Scheme," *Pattern Recognition Letters*, Vol. 22, No. 9, pp.1051-1058 (2001).
11. M. Jo and H. D. Kim, "A Digital Image Watermarking Scheme Based on Vector Quantization," *IEICE Transactions on Information and Systems*, Vol. E85-D, No. 6, pp. 1054-1056 (2002).
12. M. Y. Wu and J. H. Lee, "A Novel Data Embedding Method for Two-color Facsimile Images," in *Proc. Int. Symp. on Multimedia Information Processing*, Chung-Li, Taiwan, R.O.C., Dec. (1998).
13. P. Tsai, Y. C. Hu and C. C. Chang, "An Image Hiding Technique Using Block Truncation Coding," in *Proc. of Pacific Rim Workshop on Digital Steganography*, Kitakyushu, Japan, July, pp. 54-64 (2002).
14. R. Z. Wang, C. F. Lin and J. C. Lin, "Image Hiding by Optimal LSB Substitution and Genetic Algorithm," *Pattern Recognition*, Vol. 34, No. 3, pp.671-683 (2001).
15. Y. C. Tseng, Y. Y. Chen, and H. K. Pan, "A Secure Data Hiding Scheme for Binary Images," *IEEE Transactions on Communications*, Vol. 50, No. 8, August, pp.1227-1231 (2002).
16. Y. K. Lee and L. H. Chen, "High Capacity Image Steganographic Model," in *Proc. of IEE International Conference on Vision, Image and Signal Processing*, Vol. 147, No. 3, pp. 288-294 (2000).
17. Min Wu and Bede Liu, "Data hiding in binary image for authentication and annotation," in *IEEE Transactions on Multimedia*, Vol. 6, Issue 4 , Aug., pp. 528 – 538, (2004).
18. Haiping Lu, Kot, A.C., and Jun Cheng, "Secure data hiding in binary document images for authentication," in *Proc. of the International Symposium on Circuits and Systems*, 2003(ISCAS '03.).Vol. 3 , 25-28 May, pp. III-806 - III-809 (2003).

# Performance Analysis of CDMA-Based Watermarking with Quantization Scheme*

Yanmei Fang[1,2], Limin Gu[1,3], and Jiwu Huang[1,2]

[1] School of Information Science and Technology, Sun Yat-Sen University, Guangzhou 510275, P. R. China
[2] The Guangdong Key Laboratory of Information Security Technology
{issfym, isshjw}@zsu.edu.cn
[3] Beijing Research and Development Center, China Construction Bank, Beijing 100055

**Abstract.** In most existing spread spectrum watermarking algorithms, the embedding parameters, such as the embedding strength and spreading code length, are frequently determined via experiments. In this paper, the theoretical formulas that associate the embedding strength with the user number, or with the spreading code length, are estimated and tested, by analyzing the CDMA (Code Division Multiple Access) spreading strategies in quantization-based data hiding scenario. Moreover, a performance analytical schema in terms of BER (bit error rate) and SNR (signal-to-noise ratio) is proposed and tested both theoretically and experimentally. The interesting conclusions show that the performance of the CDMA-based data-hiding systems, focusing on quantization scheme, is independent of the user number under the constraints of imperceptibility, and an increase of the spreading code length will lead to a decrease of the robust performance. The simulation results are presented to support the conclusions. Although the work presented in this paper focuses on image watermarking, it may be extended to audio/video watermarking.

## 1 Introduction

Theory and method in digital communication have been applied to the embedding and detection of digital watermarking [1]. There are a number of advantages in a CDMA communication system: multiple access, large capacity, high security, resistant to interference and noise etc. Furthermore, CDMA systems are secure by pirate attacks because the third party is unable to reconstruct the base-band signals. It is an efficient method that embedding the digital watermarking information into the digital media applying the principle of CDMA communication [2, 3-8].

Joseph et al [3] firstly proposed in 1998, that the watermarking information could be spread out to the *m* sequences in the form of string sequences using CDMA techniques. The CDMA-encoded watermark information was embedded into the 128 ×128 DCT coefficients. Silvestre et al [4] proposed that information was embedded in the

* Support by NSF of China (60325208, 60133020), NSF of Guangdong (04205407), funding of China National Education Ministry.

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 350 – 361, 2005.

frequency domain by modulating the selected DFT values of the image and using sets of orthogonal codes in a fashion similar to CDMA. The DFT values were grouped into different bands defining independent channels for carrying data. Kohda et al [5] proposed a method for embedding the digital watermark into the color image using spreading spectrum channels with CDMA. The RGB image was transformed into YIQ signal. He selected the first 15 DCT coefficients of signal Y, 6 coefficients of signal I, 3 coefficients of signal Q to form the separate CDMA channel with spreading sequences of variable-period to transmit YIQ signals. Vassaux et al [6] proposed that the separate CDMA channel was formed directly in the space domain by dividing the original image into multiple layers, and then select the 1,2,4,8 layers to embed the watermark message. Bijan [7,8] claimed that the principle of spreading sequences CDMA communication had a natural application in the uncompressed digital video watermarking.

In most of existing spread spectrum watermarking algorithms, the embedding parameters are determined through experiments. For a CDMA watermarking channel, how about the constraint relationship among the robustness, denoted as BER in this paper, the number of orthogonal sequences, i.e. the user number, the length of orthogonal spreading codes, the capacity of watermark and the imperceptibility of watermark in host work. How do the above parameters affect the BER?

In this paper, as an extension of the previous work [2, 9], we address the above issues. The Type-II data hiding methods had been characterized by the use of quantizer structures in the embedding and detection, and frequently employed with oblivious data hiding systems [10]. For a CDMA-based Type-II data hiding scenario, this paper will present an analysis on watermarking performance through the following aspect.

- To investigate the watermarking model in terms of CDMA encoding, quantization-based watermark embedding and detecting approaches.
- To discover the trade-offs between the user number and watermark embedding strength, between the length of orthogonal spreading codes and the watermark embedding strength, so as to provide the estimation formulas.
- To analyze theoretically the BER performance of watermark in the above scenario, and offer a BER performance schema of CDMA watermarking channel with quantization based embedding. Under the constraints of imperceptibility, the BERs of detected watermark is independent of the user number, and also, an increase of the spreading code length will leads to a decrease of the robust performance, in some extent.

## 2 Channel Model: CDMA and Quantization-Based Watermarking

By the analysis in our previous work [2], the improved Gold sequences are completely orthogonal code with zero cross-correlation. In our next theoretical analysis, Gold sequences are regarded as the main instance of orthogonal spreading codes to be discovered. But our conclusion can be extended to general orthogonal spreading codes, such as Walsh codes, *m* sequences, or Kasami code, etc.

Let the watermark message be represented in binary form and transformed into vector $b = (b_1, b_2, ..., b_B)$ by mapping, where $b_i \in \{1, -1\}$. $B$ denotes the number of bits or the capacity of the watermark information to be encoded. The mapping $1 \to -1$ and

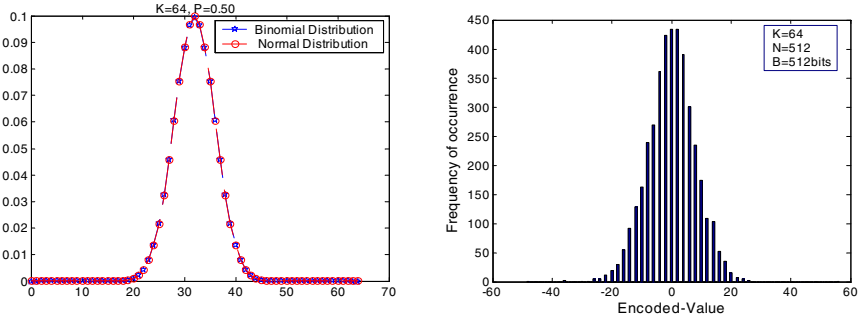



**Fig. 1.** Analysis of *C(n)*. From left to right: (a) Theoretical analysis; (b) Experimental data

$0 \to 1$ is an extremely important step because it essentially enables us to replace the *XOR* operator used in finite field algebra with multiplication. The Gold sequences are mapped in the same way. The generated Gold orthogonal sequences set is presented as $g_1(1), g_2(2), ..., g_k(n)$ , each sequence corresponding to a bit $b_k$, where $g_k(n) \in \{+1, -1\}$ , $1 \le k \le K$ ( $K \le B$ ), $1 \le n \le N$ , $N$ is the length of the spreading sequence and $K$ is the number of spreading sequences selected from a Gold codes set. For B bits of watermark, $B/K$ sets of Gold sequences are used to encode the bits. So, we can obtain each of the $B/K$ encoded vectors as follows:

$$\mathbf{C}(n) = \left( \sum_k b_k g_k(n) \right), \quad k = 1, 2, \ldots, K \quad n = 1, 2, \ldots, N , \tag{1}$$

The sequence *C(n)* is the watermark information that obtained from CDMA encoding. The analysis of encoded information is shown as Fig.1. We can draw that the information derived from CDMA spread-spectrum encoding has a random distribution. That is, for $K$ sufficiently large, the statistical distribution of this message should approach a Gaussian distribution. This follows the Central Limit Theorem.

In watermark embedding, quantization-based model is frequently used. According to the statistic property of CDMA encoding information, we adopt the following formula (2) as the embedding model [11], concerned with unitary transform domain, e.g. DCT, DFT, or DWT domain.

$$\begin{aligned} y_k &= f_k - f_k \bmod(S) + \frac{S}{2} + \alpha_k C(n) , \\ \alpha_k &= \alpha_0 \cdot \frac{S}{2} \cdot \frac{1}{\max(C(n))}, \quad 0 < \alpha_0 < 1, \ 1 \le n \le N . \end{aligned} \tag{2}$$

Where, $f_k$ and $y_k$ are the transform coefficients of the host image and their modified versions. The operator *mod* calculates the modulus, and the formula $f_k - f_k \bmod(S)$ corresponds to the fact that the residue of the coefficient $f_k$ dividing by $S$ is set to zero. $S$ is the maximum value on the premise of invisibility, called embedding strength or quantization step size. *C(n)* denotes the encoding information derived from equation

(1), and $\alpha_k$ presents the gain with embedding. In order to compromise the invisibility and the robustness of watermark, the particular value of $\alpha_k$ will be adjusted automatically in inverse proportion to the max value of *C(n)*. Obviously, $\max(|C(n)|) \le K$ .

According to the embedding algorithm, we extract the respective least significant bit from the corresponding coefficients $\hat{y}_k$ in the transform domain of the watermarked image. The embedded data is extracted as $\hat{C}(n) = \hat{y}_k \bmod(S) - S/2$ . According to the orthogonal feature, the binary watermark image has the following decision formula [1]:

$$b_k' = \begin{cases} +1 & \text{if } \Re_k \ge 0 \\ -1 & \text{if } \Re_k < 0 \end{cases}. \tag{3}$$

Where, $\Re_k = \sum_n \hat{C}(n) g_k(n), k = 1,2,\ldots,K,\ n = 1,2,\ldots,N$ , so, we obtained a series of binary watermark information sequences $b_1', b_2', \ldots, b_k', \ldots, b_B'$, where *B* denotes the bit number of watermark information.

## 3 Analysis of CDMA-Based Watermarking Channel

As mentioned in Section 2.1, the CDMA encoded watermarking information is drawn from Gaussian distribution $N(\mu, \sigma^2)$ . According to the so-called '$3\sigma$ Rule' for normal random variables, it is almost certain that the value of encoded information is within the scope of $[\mu - 3\sigma, \mu + 3\sigma]$. In other words, the distribution probability of encoding information *C(n)* satisfies

$$P\{|X - \mu| < 3\sigma\} \ge 0.9974 .$$

Suppose there are *K* users. The probability that the value of *C(n)* falling into the scope $[-3\sqrt{K}, 3\sqrt{K}]$ is near closely to 1, and the part outside of this scope is the error probability. Hence, the value of the embedding strength (or the quantization step size) is selected as:

$$S = 6\alpha\sqrt{K}, \quad 0<\alpha<1 . \tag{4}$$

Under the constraint of imperceptivity for watermark channel (preserving an accepted PSNR), the embedding strength, *S*, is directly correlated with payload, *K*. Let $K_1$ and $K_2$, $S_1$ and $S_2$ denote the payloads, i.e. the user number, and the embedding strength in two different coding schemes, respectively. If keeping the host coefficients and embedding model unchanged, we may have the following equation derived from formula (4) [9],

$$\alpha_2 = \alpha_1 \cdot \sqrt{\frac{K_1}{K_2}} \tag{5}$$

### 3.1 User Number Versus Embedding Strength

Suppose that the capacity $B$ of watermark and the spreading code length $N$ are unchanged. This section investigates how to choose $K$, so as to keep an accepted PSNR of the host signal.

Let $t$ be the number of coefficients modified in the transform domain while embedding watermark information, where $t = N \cdot (B/K)$. Obviously, the smaller the $K$ is, the more are the coefficients to be modified with a certain capacity of watermark bits. Assuming that each spreading sequence is modulated to a bipolar signal as $\{-\alpha, +\alpha\}$, and the variable $C(n)$ which derived from formula (1) is drawn from Gaussian distribution, with parameters as $E\{C(n)\} = 0$, $V\{C(n)\} = K$.

Let $C_k$ denote the data that will be embedded into the $k^{\text{th}}$ coefficient in transform domain. According to the embedding model in formula (2), the change for each coefficient can be calculated as follows

$$|\Delta f_k| = \left| f_k \bmod(S) - \frac{S}{2} - \alpha_k C_k \right| = \left| \frac{S}{2} + \alpha_k C_k - f_k \bmod(S) \right| \tag{6}$$

Assume that the pixel size of host image is $M \times M$, where $b_m$ denotes the maximum grayscale level of pixels in a host image, and $T_{psnr}$ is the minimum threshold can be reached by PSNR, that is, $PSNR \geq T_{psnr}$. Furthermore, it can be rewritten as [12]

$$T_{PSNR} \leq PSNR = 20\lg_{10} \frac{Mb_m}{\sqrt{\sum_k \Delta f_k^2}}. \tag{7}$$

Let $X = S/2 + \alpha_k C_k$, $Y = f_k \bmod(S)$, where X~$N$(S/2, $K$). Without loss of generality, assuming that variable $Y$ is drawn from uniform distribution over the quantization interval, say, $[0, S]$, we have the following features.

$$E\{X\} = \frac{S}{2};\ E\{X^2\} = K + \frac{S^2}{4},\ \text{and}\ E\{Y\} = \frac{S}{2};\ E\{Y^2\} = \frac{S^2}{3}$$

Considering that $C_1, C_2, \ldots, C_k$ are i.i.d Gaussian random variables, according to the Monto Carlo methodology and the Central Limit Theorem for i.i.d variables, we have the following equation.

$$\begin{aligned} T_{PSNR} &\leq 20\lg_{10} \frac{Mb_m}{\sqrt{\sum_k (X+Y)^2}} \\ &= 20\lg_{10} \frac{Mb_m}{\sqrt{t\left[E\{X^2\} + E\{Y^2\} - 2E\{X\}E\{Y\}\right]}} = 20\lg_{10} \frac{Mb_m}{\sqrt{BN + \frac{BNS^2}{3K}}} \end{aligned} \tag{8}$$

Therefore, given an accepted PSNR for the watermarked image, we can exploit the constraint relationship between the user number and the embedding strength. Let $C_{const}$ be a constant in the context of a certain PSNR. Thus

$$\sqrt{BN + \frac{B}{K} \cdot N \cdot \frac{S^2}{3}} = C_{const} = Mb_m 10^{-\frac{T_{PSNR}}{20}} . \tag{9}$$

Let $K_1$ and $K_2$, $S_1$ and $S_2$ denote the number of users and embedding strengths in two different coding schemes, respectively. Without changing $B$ and $N$, we can obtain the following equation, $K_2 S_1^2 = K_1 S_2^2$ .This formula can be rewritten as

$$S_2 = S_1 \cdot \sqrt{\frac{K_2}{K_1}} . \tag{10}$$

Formula (10) states the relationship between $K$ and $S$, under the constraint of imperceptivity for watermark channel. The embedding strength, $S$, is directly correlated with payload, $K$. Given a certain PSNR for the watermarked image, an increase of $K$ leads to an increase of $S$. Fig.2 illustrates this constraint relationship both by theoretical analysis and experiments on Lena image in DWT domain. It can be shown that formula (10) describes precisely the actual constraints.

### 3.2 Code Length Versus Embedding Strength

By the analysis in Section 3.1, while keeping the embedding model in formula (2) unchanged, that is, keeping PSNR and the watermark payload unchanged, and assuming that $f_k \bmod(S)$ is drawn from uniform distribution, we had derived the formula (9). Furthermore, we have,

$$N_1(3K_1 + S_1^2) = N_2(3K_2 + S_2^2)$$

where $N_1$ and $N_2$, $S_1$ and $S_2$ denote the spreading code lengths and embedding strengths in two different coding scheme, respectively. While keeping the user number unchanged, we can obtain

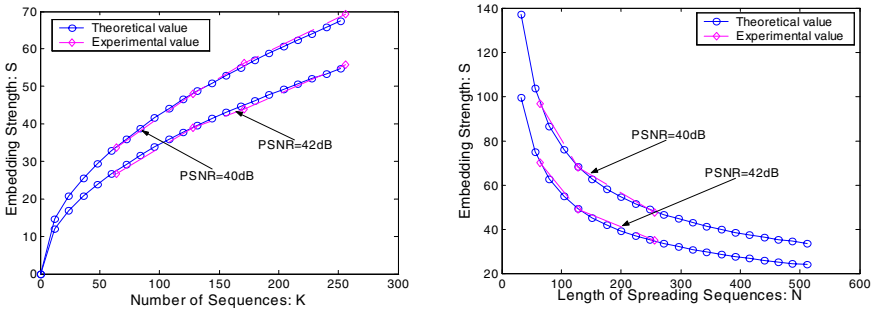



**Fig. 2.** The constraint relationship lies in the embedding parameters (in the case of "Lena" image, DWT domain, 512 bits watermark). (a) user number $K$ vs. embedding strength $S$. (b) code length $N$ vs. embedding strength $S$

$$S_2 = \sqrt{3K(\frac{N_1}{N_2} - 1) + S_1^2 \cdot \frac{N_1}{N_2}} \tag{11}$$

Formula (11) states the relationship between $N$ and $S$, under the constraint of imperceptivity for watermark channel. The embedding strength, $S$, is directly correlated with spreading sequence length (i.e. code length), $N$. Given a certain PSNR for the watermarked image, an increase of $N$ leads to a decrease of $S$. Fig.3 illustrates this constraint relationship both by theoretical analysis and experiments on Lena image in DWT domain. It can be shown that formula (11) describes precisely the actual constraints.

### 3.3 Analysis of BER Performance

Assume that the channel noise is an additive Gaussian noise with zero mean and variance $\sigma^2$. According to the communication theory [13], the channel bit error rate can be expressed as

$$p_e^{\ u} = Q(\frac{S}{4\sigma}) = Q(x).$$

We can observe that the channel BER is related to watermark embedding strength, while keeping the noise unchanged. However, according to the analysis in Section 3.1 and 3.2, the embedding strength $S$ is directly related to $K$ and $N$, respectively. Thus, given a PSNR of the watermarked image, the BER of watermark detection is also affected by the sequence number and spreading sequence length, besides the power of noise.

In the detecting procedure, the received information is an i.i.d random variable, which can be written as $\bar{C}(n) = C(n) + n_0$. Let $\mathbf{g}$ denotes the $N$-dimension orthogonal sequence vector at the receiver, which is a continuous Gauss random variable. Thus, BER in watermark detection can be expressed as

$$\begin{aligned} P_{be} &= P\left\{\sum_{i=1}^{N} \bar{C}(i) g_k(i) > 0 \middle| b_k = -1\right\} \cdot P\left\{b_k = -1\right\} \\ &\quad + P\left\{\sum_{i=1}^{N} \bar{C}(i) g_k(i) < 0 \middle| b_k = +1\right\} \cdot P\left\{b_k = +1\right\} \\ &= \frac{1}{2} P\left\{\sum_{i=1}^{N} \bar{C}(i) g_k(i) > 0 \middle| b_k = -1\right\} + \frac{1}{2} P\left\{\sum_{i=1}^{N} \bar{C}(i) g_k(i) < 0 \middle| b_k = +1\right\} \end{aligned} \tag{12}$$

For a data bit $b_i$, $g_k(i)$ is determined. Let $z = \sum_{i=1}^{N} \bar{C}(i) g_k(i)$, which follows the Gauss distribution with the parameters as (in the case of $b_k = -1$)

$$E\{z\} = -\sum_{i=1}^{N} g_k(i) g_k(i) = -\mathbf{g}_k \bullet \mathbf{g}_k = -N,\ V\{z\} = -\sum_{i=1}^{N} \left(g_k(i)\right)^2 V\left\{\bar{C}(i)\right\} = N\sigma^2.$$

So $z \sim N(-N,\ N\sigma^2)$. Then the watermark detection error probability is expressed as

$$P_e = \frac{1}{2}P\left\{z>0\middle|b_k=-1\right\}+\frac{1}{2}P\left\{z<0\middle|b_k=+1\right\}$$
$$=\frac{1}{2}\int_0^{+\infty}\frac{1}{\sqrt{2\pi}\cdot\sqrt{N}\sigma}\exp\left\{-\frac{(x+N)^2}{2\sqrt{N}\sigma}\right\}dx+\frac{1}{2}\int_{-\infty}^{0}\frac{1}{\sqrt{2\pi}\cdot\sqrt{N}\sigma}\exp\left\{-\frac{(x-N)^2}{2\sqrt{N}\sigma}\right\}dx$$
$$\underline{\underline{Let\ \ t=\frac{x+N}{\sqrt{N}\sigma},\ u=\frac{x-N}{\sqrt{N}\sigma}}}\ \ \frac{1}{2}\int_{\frac{\sqrt{N}}{\sigma}}^{+\infty}\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{t^2}{2}\right\}dx+\frac{1}{2}\int_{-\infty}^{-\frac{\sqrt{N}}{\sigma}}\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{u^2}{2}\right\}dx$$
$$=\frac{1}{2}Q(\frac{\sqrt{N}}{\sigma})+\frac{1}{2}Q(\frac{\sqrt{N}}{\sigma})=Q(\frac{\sqrt{N}}{\sigma}) \tag{13}$$

In fact, when $b_k=\pm 1$, and $\alpha$ presents the watermark-embedding weighting factor, we obtain

$$E\left\{\bar{C}(i)\right\}=\pm\alpha C(i)\ ,\ V\left\{\bar{C}(i)\right\}=\sigma^2,\ E\left\{z\right\}=\pm\sum_{i=1}^{N}C(i)\alpha C(i)=\pm\alpha N,\ V\left\{z\right\}=N\sigma^2.$$

So, the watermark detection error probability is expressed as

$$P_e = Q(\frac{\alpha\sqrt{N}}{\sigma}). \tag{14}$$

According to the equation (4), $\alpha$ can be expressed as

$$\alpha=\frac{S}{6\sqrt{K}} \tag{15}$$

Thus, equation (9) can be rewritten as

$$\sqrt{BN+12\alpha^2BN}=C_{const}. \tag{16}$$

By solving for $\alpha$, we obtain

$$\alpha=\sqrt{\frac{C_{const}^2-BN}{12BN}} \tag{17}$$

Substituting it into formula (14), we obtain

$$P_e = Q(\frac{1}{\sigma}\sqrt{\frac{C_{const}^2-BN}{12B}})=Q(\frac{1}{2\sqrt{3}\sigma}\sqrt{\frac{C_{const}^2}{B}-N}). \tag{18}$$

In formula (18), the constant $C_{const}=Mb_m10^{-\frac{T_{PSNR}}{20}}$ is determined by the experimental values of *B*, *N*, *K*, and *S*. The greatly degradation of the host signal quality (PSNR) by embedding, results in a bigger value of $C_{const}$. Fig. 3 shows the effects using three different spreading code lengths with the same watermark payload and different PSNRs. In the case of a smaller $C_{const}$, e.g. 300, a bigger length, *N* can lead to a higher detect error probability. However, under a bigger $C_{const}$, e.g. 600, corresponding to a

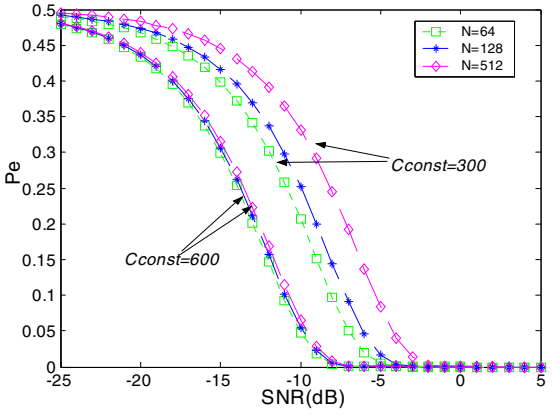



**Fig. 3.** Analysis of the BER performance vs. different spreading code length

greatly degraded host signal quality, the spreading code length has a little effect on BER.

## 4 Simulation Results

In the simulations, we choose DWT domain of a number of 512×512×8 bits test images, such as Lena, Baboon, Peppers, to illustrate our analytic results. The BER is measured in the case of JPEG compression and Gaussian noise, respectively. It is worthily noted that our models are still applicable for other transform domain.

In the case 1, we embed 512 bits into host image with different $K$. Fig.4 shows the measured curves of BER against JPEG compression. In the case 2, we embed 256 bits into host image. Fig.5 shows the measured curves of BER against JPEG compression. It can be shown that BER curves are very close under the different user number, $K$.

Hence, the analytic result in formula (18) is illustrated, that is, the watermark detection error rate is independent of the user number. Based on the consistency of the theoretical performance analysis and experiment results, we believe that our model and methodology to evaluate the watermark performance are reasonable.

In the case 3, we embed 512 bits into Lena image (PSNR=42dB) and Baboon image (PSNR=40dB) by employing Walsh codes under different code length. Figs.6~7 show the measured curves of BER against JPEG compression and Gaussian noise. In the case 4, we embed 128 bits into Lena image by employing Gold codes under the different code length. Fig.8 shows the measured curves of BER against JPEG compression and Gaussian noise. It can be shown that BER performance can be distinguished from different spreading code length.

As a result, under the constraint of imperceptivity for watermark channel, a bigger code length ($N$=256 or $N$=512) leads to a higher BER, and a smaller code length ($N$=64 or 32) leads to a lower BER. With a lower payload (128 bits), the spreading sequence length has a larger effect on the detection error probability. Hence, the analytic results in formula (18) are illustrated. Our performance framework can be extended to general

orthogonal spreading codes, including Walsh codes, *m* sequences, or Kasami codes, etc. Figs. 6~7 are the results employing Walsh codes. Diagrams in Fig.8 show the results applying our improved Gold codes in our previous work [2].

By the performance analysis of CDMA watermarking channel both theoretically and experimentally, we can draw some conclusions as follows.

- Detection error probability $p_e$ vs. the user number, $K$. Under the constraint of imperceptibility, there is a constraint relationship among the user number, the embedding strength, and the capacity of watermark. Theoretical analysis and experiment results illustrate that $p_e$ is independent of the user number $K$.
- Detection error probability $p_e$ vs. spreading code length, $N$. Under the constraint of imperceptibility, there is a constraint relationship among the spreading code length, the embedding strength, and the capacity of watermark. Theoretical analysis and experiment results illustrate that the performance reactivity introduced by $N$ is related to the context of $C_{const}$. With a lower payload added to the host signal, an increase of $N$ will lead to a decrease of $S$, and thus performance is degraded. On the other hand, with a larger payload, the different length of spreading code has a lighter effect on the detection error probability.

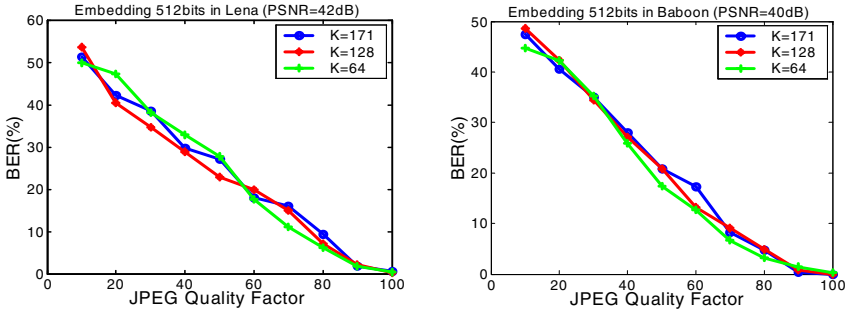



**Fig. 4.** BER vs. $K$, in the case of embedding 512 bits in host image. From left to right: (a) Lena PSNR=42dB; (b) Baboon PSNR=40dB

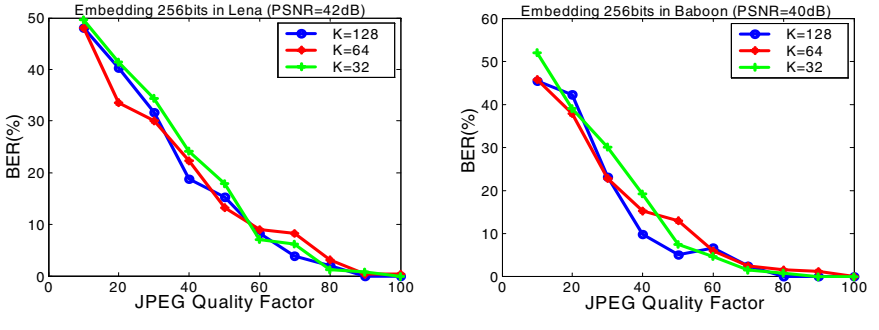



**Fig. 5.** BER vs. K, in the case of embedding 256 bits in host image. From left to right: (a) Lena PSNR=42dB; (b) Baboon PSNR=40dB

## 5 Conclusions

In this paper, we have investigated the constraint relationships in CDMA Type-II watermarking channel. We present a watermarking performance framework from both theoretical analysis and experimental works. The main contributions and conclusions are as follows.

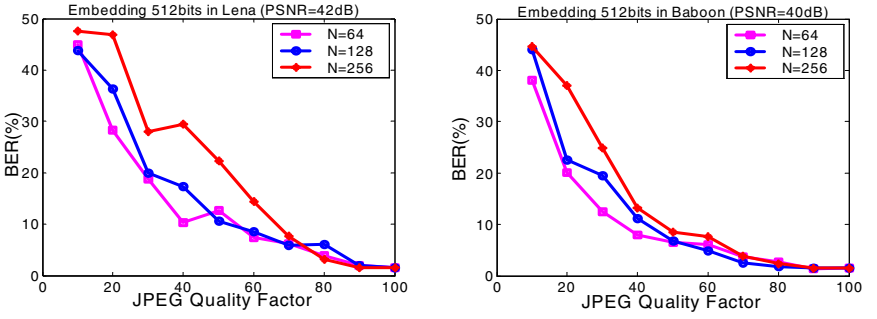



**Fig. 6.** BER vs. $N$, in the case of JPEG compression ($K$=64). From left to right: (a) Lena PSNR=42dB; (b) Baboon PSNR=40dB

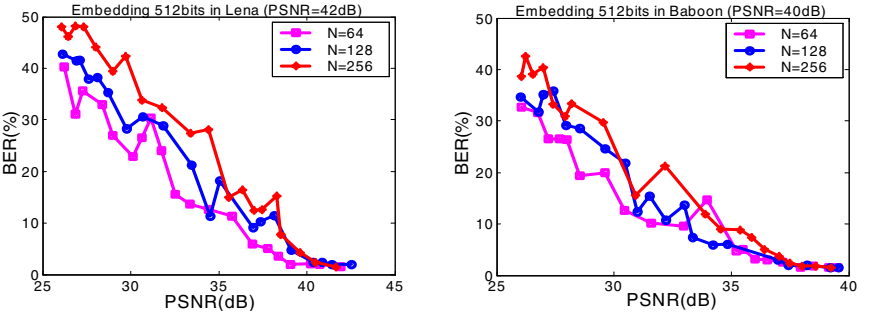



**Fig. 7.** BER vs. $N$, in the case of Gaussian noise ($K$=64). From left to right: (a) Lena PSNR=42dB; (b) Baboon PSNR=40dB

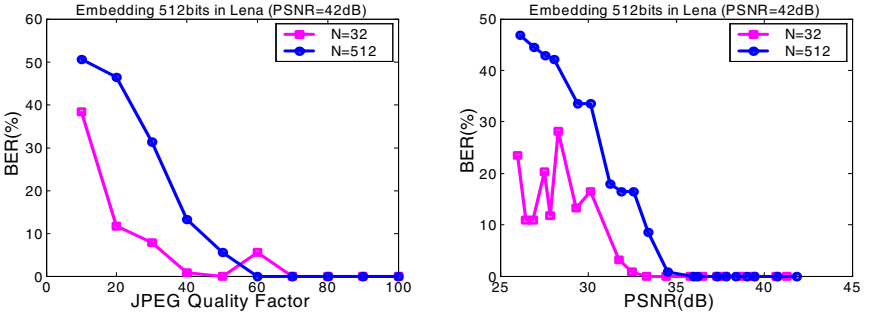



**Fig. 8.** BER vs. $N$, in the case of Gold codes ($K$=16). From left to right: (a) JPEG compression; (b) Gaussian noise

- Investigated the watermarking model in terms of CDMA encoding, quantization-based watermark embedding, and detecting approaches.
- Discovered the constraint relationships between the user number and watermark embedding strength, between the orthogonal spreading code length and the watermark embedding strength.
- Analyzed theoretically the BER performance of watermark under the constraints of imperceptibility, and offer a BER framework. As a result, the robust performance of detected watermark is independent of the user number, and an increase of the spreading code length will lead to an increase of BER in some extent.

Many experiments have been made to test the constraint relationships and verify the performance framework derived in this paper. Experimental results support our analysis. Most of our analysis for image watermarking can be extended to video and audio spread spectrum watermarking. However, the differences in quantization-based and additive embedding model must be taken into account. Further research will attempt to extend the analysis to general multimedia watermarking model.

## References


1. Cox I. J., Miller M.L. and Bloom J. A.: Digital Watermarking, Morgan Kaufmann, 2001
2. Fang Y., Huang J.: Image Watermarking Algorithm Applying CDMA, Proc. of IEEE Int. Sym. on Circuits and Systems, Bangkok, Thailand, Vol. 2(5) (2003) 948-951
3. Joseph J.K., Ruanaidh Ò., Thierry P.: Rotation, Scale and Translation Invariant Spread Spectrum Digital Image Watermarking, Signal Processing. Vol. 66(3) (1998) 303-317
4. Silvestre G.C.M., Dowling W.J.: Embedding Data in Digital Images Using CDMA Techniques. In: Proc. of IEEE Int. Conf. on Image Processing, Vancouver, Canada 1(2000) 589-592
5. Kohda T., Ookubo Y., Shinokura K.: Digital Watermarking Through CDMA Channels Using Spread Spectrum Techniques. In: IEEE 6th Int. Sym. on Spread Spectrum Techniques and Applications, Parsippany, NJ, USA, Vol. 2(2000) 671 –674
6. Vassaux B., Bas P., Chassery J.M.: A New CDMA Technique for Digital Image Watermarking Enhancing Capacity of Insertion and Robustness. In: Proc. of IEEE Int. Conf. on Image Processing, Thessalonica, Greece, Vol. 3(2001) 983 -986
7. Bijan G.M. Exploring CDMA for Watermarking of Digital Video. Villanova University, Villanova, PA, USA, 19085, http://www.ece.villanova.edu/~mobasser/mypage/3657-10.pdf.
8. Bijan G. M. Direct Sequence Watermarking of Digital Video Using m-frames. In: Proc. of IEEE Int. Conf. on Image Processing, Chicago, Illinois, USA, Vol. 2(1998) 399 -403
9. Gu L.M., Huang J.W., Shi Y.Q.: Analysis of the Role Played by Error Correcting Coding in Robust Watermarking, In: Proc. of IEEE Int. Sym. on Circuits and Systems, Vol. 3(5) (2003) 798-801
10. Sencar H.T., Ramkumar M. Akansu A.N.: A Robust Type-III Data Hiding Technique against Cropping & Resizing Attacks, Proc. of IEEE Int. Sym. on Circuits and Systems, Vol. 2(5) (2002) 444-446
11. Tsai M.J., Yu K.Y., Chen Y.Z.: Joint Wavelet and Spatial Transformation for Digital Watermarking, IEEE Trans. On Consumer Electronics, Vol. 46(1) (2000) 241-245
12. Liu R.Z., Tan T.N.: A General Watermark Framework for Optimal Energy Estimation, Chinese J. Computer, Vol. 24(3) (2001) 242-246
13. Cao Z.G., Qian Y.S.: The Principle of Modern Communication, Tsinghua University Press, Beijing, China, 1992.

# Protecting Mass Data Basing on Small Trusted Agent

Fangyong Hou, Zhiying Wang, Kui Dai, and Yun Liu

School of Computer, National University of Defense Technology,
Changsha, 410073, P.R.China
fyhou@etang.com

**Abstract.** Providing data confidentiality and integrity is essential to ensure secure or trusted computing. Designs for such purpose always face substaintial difficulties, as providing solid security will be contrary to achieving satisfied performance. Basing on a less rigor precondition that will be tenable in many cases, such designs can be implemented with smaller endeavors. The core idea is to let a trusted agent to trustworthily hold one unique timestamp for each untrusted data block; and encrypts each block, as well as the related integrity code, through the corresponding timestamp. In such way, any malicious disclosure and tamper can be prevented. At the same time, each block can be directly verified by the associated timestamp without requiring additional data to minimize the cost of integrity checking, and OTP encryption scheme can pre-computes keystream to remove most encryption latencies.

## 1 Introduction

Data confidentiality means that data must be restricted to who sees what data or contents are not readily accessible, that is, it prevent unauthorized disclosure of information. Data integrity means protection of data from corruption or unauthorized modification, that is, provides a tamper-proof environment.

Providing data confidentiality and integrity is necessary to ensure secure or trusted computing. But implementation of solid protection is often contrary to realizing high performance. For confidentiality, protection is fulfilled through cryptography. However, high encryption intensity requires high cost also. For integrity verification, it becomes a more difficult task. One difficulty comes from the requirement of online integrity checking, which can avoid committing error result but requires more cost to check frequently. The most intractable difficulty is to resist against the potential replay attack. Replay attack means that adversary stores a message and its signature, then uses them to spoof users later. To prevent from replay attack, system must regard the content of all files/blocks at some point as one continuous set of data, and maintains a single *(all of the data, authentication code)* pair. However, making signature on mass data is never a lightweight work.

In fact, a strict precondition makes implementation of such protections become a hard task. That is, only a very small trusted core can be depended on. In this trusted core, just very small security related information, like the root secret key and the root authentication code, can be hold permanently. All of others must be left in untrusted region. So, big cost will be paid to authenticate them as valid before reuse them again.

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 362 – 373, 2005.

Actually, many practical situations can have a relative big trusted region. For example, a PC in your bedroom shouldn't be brittle to hardware attacks, as adversary has no way to touch it physically. All of this PC can be treated as trusted when you read data from your company server (which may be controlled by adversary out of your sight). From this point, we propose an approach relying on a trusted agent to protect mass data, which adapts to the needs of many application cases. Such agent can securely hold security information. The security information is unique timestamps associated with each data block separately. Basing on a unique timestamp, each data block, as well as its integrity code, is encrypted by a unique secret keystream through an OTP (One-time-pad) cipher. In such way, data can be protected with advantages as:

- *Encryption/decryption has low run-time cost*. Because the keystream can be prepared in advance, most data encryption/decryption processes only require XOR operations when transforming plaintext/ciphertext data.
- *Online integrity verification is fulfilled*. Judgments can be immediately made according to the data block that is currently read. For example, checking integrity of a file data block needn't to wait all the data blocks of the file to be reached.
- *Replay attack is prevented without performance penalty*. Each block's integrity can be checked individually, but it still being able to resist against replay attack. As it needn't maintain signature on much related data, checking cost is minimized.

The rest of this paper is organized as follows. Section 2 gives an overview of related works and the protection model. Section3 elaborates the method of our protection. Section 4 describes some specific applications. Section 5 concludes this paper.

## 2 Protection Issues

This section gives an overview of related works and proposes the protection model.

### 2.1 Related Works

Protecting confidentiality of mass data is commonly fulfilled through secret key cryptography, such as block ciphers like AES (Advanced Encryption Standard) or stream ciphers. The process of encryption/decryption is relative straightforward. The main concern of it is the encryption intensity and performance.

Generally, integrity is verified by the integrity code that is also referred as MAC (Message Authentication Code, which is hash of a message with a fixed length cryptographic fingerprint of the message, and can be computed over the data in combination with its identification ID like the block number). But MAC is brittle to replay attack. Adversary can replace the *(message, MAC)* pair for one data with pairs stored for the same data on earlier stores without being detected.

To resist against replay attack, hash tree or Merkle tree [1], is an available way [2]. Hash tree maintains a single *(all of the data, hash)* pair in an iterative manner. Adversary cannot replace some of the files (or blocks) without being detected. But a naive hash tree will bring a significant performance overhead. Some optimizing measures are put forward [3] [4]. However, notable performance decline still exists.

There exist many systems that protect mass data. Among them, some only provide confidentiality, some give integrity relying on MAC, and some use the principle of hash tree to resist against replay attacks. CFS [5] encrypts file data to provide confidentiality. Tripwire [6] uses file's MAC to check integrity of files. SFSRO [7] uses hash of a block as the block identifier to guarantee the integrity of content data. SUNDR [8] uses a hash of a block as the block identifier and a hash tree to provide data integrity at the file system level. TDB [9] describes a database in which it integrates encryption and hashing with a low-level data model that protects data and metadata uniformly, and adopts the techniques used in the Log-Structured File System (LFS). PFS [10] protects data integrity at the block level. It keeps a list called the block map in order to map a file system block number to a hash of a block to protect data integrity. Arbre [11] builds hash tree into file system design tightly to protect data integrity on untrusted remote storage, with a notable point of protecting integrity of the entire file system. But these existing systems have some disadvantages. For example, LFS approach used by TDB needs to write a big map containing hash tree to disk at a time and is difficult to achieve it without performance degradation. PFS cannot give entire protection. Arbre inherits some limitations that tree-structured file systems have.

### 2.2 Considered Protection Model

Considering an agent can be relied upon, we propose an approach that can achieve solid protection and good performance at the same time. Focusing on method investigation, we first make some simplifications to facilitate our descriptions. After simplifications, data accessing has the model shown in fig.1.

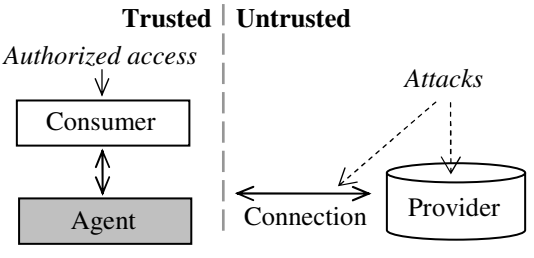



**Fig. 1.** Protection Model. Agent makes the data stored on untrusted Provider to be trustworthy to Consumer

In fig.1, Consumer and Agent lie in the inner of trusted boundary, while Provider is always untrusted. Consumer is the owner/user of data; it can act as a file system to interpret data. Provider supplies mass data storage; it can be seen as a block server. Agent lies between Consumer and Provider; its task is to provide protection for those data blocks flowing between Consumer and Provider. The definition of trusted boundary can be different for different cases. For example, entire of your PC can be trusted when you accessing another remote PC; and only its processor (and/or memory) can be trusted when you think that the hard disk attached to it may be tampered. The character of Agent is that it has a secure storage to hold security related informa-

tion. Such storage of Agent has a relatively small capacity when compared with the storage capacity of Provider. This is also the reason that we call it "Small Trusted Agent".

In this model, function of any data accessing is: (i) given a *(block-address, trusted data block)* pair by Consumer, it is converted by Agent, and then is saved by Provider; (ii) given a block-address by Consumer, corresponding untrusted data block is returned from Provider, and then is re-converted by Agent to reach Consumer as trusted. Adversary can do attacks in any imaginable ways. For example, a malicious administrator of remote server can subvert or forge file data placed in this server; also, any passive and active listeners can tap the data or modify it on its network path. The purpose of conversions executed by Agent is to prevent all of such attacks.

## 3 Protecting Approach

Our approach tries to achieve several purposes. The first is to make encryption bring little influence to run-time performance. The second is to detect any integrity tampering (including replay attacks), and solve the problem of performance penalty that hash tree scheme has suffered from. The third is to give online checking. Additionally, it is better to have simple management requirement, without complicated data structures and operations. The details of our protection are described as the followings.

### 3.1 Basic Security Process

For convenience, we use "D-Data/D-MAC/D-Block" to represent a data/MAC/block in readable form (i.e., it is not encrypted); use "E-Data/E-MAC/E-Block" to represent a data/MAC/block in ciphertext form; use "TS" for timestamp; and use "ID" to denote the addressing information to locate the storage place of a block (like block number in meta-data of file systems). Among them, one D-Block is a *(D-data, D-MAC)* pair, while one E-Block is a *(E-data, E-MAC)* pair. In order to protect any data blocks, we define the following functions of Consumer, Agent and Provider.

Consumer executes as:

- *WriteData*. Consumer sends D-Data and its related ID to Provider; update operations will be treated as a new *WriteData*, even it has the same ID with previous writes and updates.
- *ReadData*. Consumer sends ID to Provider.
- *ShowData*. Consumer interprets the returned D-Data.

Agent executes as:

- *TimeIncrement*. Agent increases TS to produce a new one.
- *KeyGenerate*. Relying on each unique TS, Agent generates a unique keystream.
- *SignBlock*. For each D-Data flowing from Consumer to Provider, Agent does: (i) hashes D-Data to get its D-MAC; (ii) concatenates D-Data with D-MAC to build a D-Block; (iii) produces a new TS through *TimeIncrement*, and uses *KeyGenerate* to generates a new keystream from this TS; (iv) encrypts D-Block by OTP cipher; (v) sends the resultant E-Block, or *(E-data, E-MAC)* pair, to Provider; and (vi) saves the corresponding TS and ID to the trusted storage place of Agent.

– *CheckBlock*. When an E-Block is flowing from Provider to Consumer, Agent does: (i) fetches the corresponding TS, according to the value of ID; (ii) feeds this TS to *KeyGenerate* to get the corresponding keystream; (iii) decrypts E-Block by OTP cipher to get D-Data and D-MAC; (iv) re-calculates the MAC of the decrypted D-Data and compares the result with the decrypted D-MAC; and (v) if matching, sends D-Data to Consumer, or else, indicates an violation of security.

Provider executes as:
– *ReadBlock*. Given a specific ID from Consumer, Provider returns the corresponding E-Block, which including one E-Data and its related E-MAC.
– *WriteBlock*. Given an E-Block and its corresponding ID from Consumer (through Agent), Provider can store them to the appropriate storage place; answers an acknowledgement after completing the store operations.

With these functions, any data block is processed as:
– *Write a block*. Does in turn: (i) *WriteData*, (ii) *SignBlock*, and (iii) *WriteBlock*.
– *Read a block*. Does in turn: (i) *ReadData*, (ii) *ReadBlock*, (iii) *CheckBlock*, and (iv) *ShowData*.

### 3.2 OTP Design and Confidentiality Protection

In order to use OTP, the core of Agent should have a build-in security key called "RootKey", which can never be accessed from the outer part of the trusted boundary. To produce encryption keystream, Agent generates a new and unique value of TS for each write operation. TS can be the physical time such as the value of system timer (but it must have sufficient fine granularity to distinguish two continuous write operations), or can be just an incremental counter. But the best way to generate TS is to use high quality random numbers, because such TS will enhance the random character or the quality of keystream. As TS never repeats, keystream will never be repeated also.

The lengths of RootKey and TS are important selections. Longer of them can produce better keystream with higher encryption intension, but will require most cost to deal with them. The minimum size of RootKey should be 128bit, and less length of it may compromise security. As we will use SHA-2 (512) [12] hash function to produce keystream, we select the size of RootKey to be 1024bit. Such a long size of RootKey is sufficient to resist against cryptanalysis, and it equals the size of the input block of SHA-2 (512). TS should have a length of 64bit to distinguish each write operation from others. A shorter length of TS like 32bit will make keystream to be repeated more probably, which will give adversary chance to perform replay attack. But TS shouldn't be too long also; or else, it will increase the cost of holding them securely.

With one given pair of RootKey and TS, encryption engine of OTP generates keystream as ("||" represents concatenation):

```
Loop for i =1 to k
  keyseg_i = SHA512(LeftTruncate1024(RootKey||TS||i))
  keystream_OTP = keystream_OTP + keyseg_i
  i = i + 1
End Loop
```

Here, SHA512 is hash function of SHA-2 (512), which has 512bit output result. The input (truncated to be 1024bit) of this hash function is the concatenation of RootKey, the current value of TS, and a variable of *i*. The output of it composes one encryption bit sequence sub-segment. Concatenating *k* such continuous sub-segment gets the needed keystream to encrypt one block. The value of *k* is determined by the length of block to be encrypted. For example, encrypting one 4KB block needs a *k* with the value of *(4KB/512b = 64)*.

When reading a block, Agent fetches the same TS associated with this block. So, the same keystream is produced to decrypt the encrypted block correctly.

Keystream can be pre-computed before data encryption occurs, as the generation of keystream needn't to wait the destination block ID to be determined. System can utilize its spare time (e.g., when CPU isn't busy) to produce keystream in advance. Then, only XOR operation is needed to encrypt a block. Decryption keystream can also be pre-computed, as the values of TS will be fetched more quickly than the returned blocks for most cases. In such way, the run-time performance of encryption/decryption will be satisfying.

The conventional OTP scheme is proved to be secure [13]. OTP cipher proposed here can be enhanced by some improvement measures [14], and there exist many sound instantiations of counter-mode encryption scheme. Users can select any of them to replace our OTP scheme, while the idea of our protection is still valid.

### 3.3 MAC Design and Integrity Verification

The original MAC (i.e., the D-MAC) is produced by hashing D-Data directly. As Wang [15] has found collisions for MD5 hash function, SHA-1 and SHA-2 should be better for very security requirement. In fact, MD5 can still preserve secure for many common applications. Additionally, as the data block and MAC are all encrypted, ability of finding collision for MD5 hash result has no direct use for attacking.

As adversary doesn't know the secret keystream for a given E-Block (including its E-Data and associated E-MAC), he cannot produce a *(D-Data, D-MAC)* pair that can still preserve matching after decryption. So, modifications to the bits of E-Block can be detected. Copying one E-Block to another is also impossible, because decryption keystream will be wrong and the resultant *(D-Data, D-MAC)* pair will be invalid.

By assigning different encryption keystream to different blocks, replay attack is impossible. Sending a stale E-Block to spoof will incur mismatch when verify the attached MAC, because the decryption keystream for a fresh E-Block is not the same one for old E-Blocks and wrong *(D-Data, D-MAC)* pair will be produced.

Because integrity of each data block is verified directly by the attached MAC, it gives online checking. Without requiring other blocks, resistance against replay attack is implemented with little performance decline, which is one distinct advantage.

### 3.4 Consistency

Maintaining data consistency doesn't suffer from security protection. As the original architecture of Consumer and Provider can be kept with no change (Agent can be just embedded between Consumer and Provider), only inconsistency between an E-Block and its corresponding TS value should be considered. If acknowledgement for each

write operation has returned from Provider and storing of corresponding TS value has succeeded, no consistency will exist. Consumer can buffer its recent write operations, and retries until receiving acknowledgements (Note: Retries are not treated as new write operations of Consumer; so, no changing of TS); or else, it should abandon those uncompleted operations, including related storing of TS.

In this sense, management of protection is simple. It is convenient for integrating such protection mechanism into existing systems.

## 4 Security Applications

This section applies our protection approach to build several specific applications.

### 4.1 Protect Data in Untrusted Remote Server

In many distributed computing systems, remote nodes may contribute to store data for a client. For example, an SCSI device is connected to IP network, and the file system implemented in client can use this remote storage device to store data blocks. Such case becomes more popular in today's storage architecture. Another example, in Grid computing environment, although a computer system in Grid usually has ability to store some data locally, it may want some contributors to provide it with extra storage space to hold more data. In these cases, our security protection may be useful, which can utilize some trusted information held locally to protect ustrusted data stored remotely.

Applying our protection, Consumer is the file system implemented in client, and meta-data is hold by the file system itself. Provider is the remote server that provides storage of data blocks. Agent is an additional component in client. Related deployment is shown as fig.2.

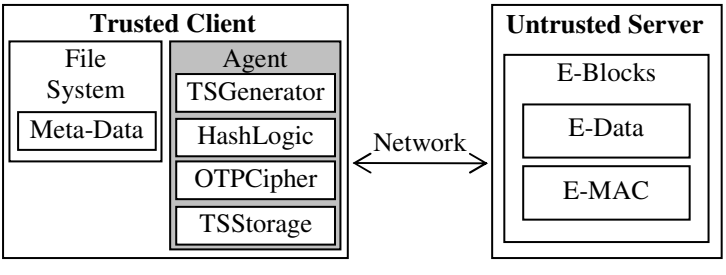



**Fig. 2.** Architecture of protecting data in remote untrusted server. Confidentiality and integrity of data stored in remote server are protected

In fig.2, trusted client uses its file system to interpret data. Through meta-data held by file system itself, file system is simply treated to be: given the root directory, it knows each number/addressing of blocks of any specific files. File blocks that actually contain data are placed in untrusted server and are encrypted together with their MACs. Agent implements security protections.

The processes of each data write are:
1. File system sends out a D-Data and its ID (contained in meta-data of file system);
2. Agent uses HashLogic to compute the D-MAC of D-Data;
3. Agent uses TSGenerator to produce a new value of TS;
4. Agent uses OTPCipher to generate new keystream through the new value of TS;
5. Agent concatenates D-Data and the corresponding D-MAC to compose a D-Block, and encrypts this D-Block through OTPCipher, which implements XOR operations on every bits of D-Block and keystream;
6. Agent sends the resultant E-Block forward, and saves the corresponding values of TS and ID if an acknowledgement returned from untrusted server.
   Note: Steps (3.) and (4.) can be finished at any time before step (5.).

The processes of each data read are:
1. File system sends out an ID, then related E-Block is returned from untrusted server;
2. Agent fetches the corresponding TS and uses OTPCipher to generate keystream (this step can be carried with step (1) simultaneously);
3. Agent uses OTPCipher to decrypt E-Block;
4. Agent uses HashLogic to calculate MAC for the decrypted D-Data;
5. Agent matches the decrypted D-MAC with the new calculated one, and returns valid data to file system if matching.

We implement a prototype. Two PCs are used to act as client and untrusted server separately. On client, file system of ext2 (Second Extended File System, a Linux file system) is simulated by software. Agent consists of several software programs to implement OTP encryption, SHA-1 hash function with 160bit output, and random number generation for TS. Agent also maintains a database to hold the values of TS. Untrusted server is simulated by database to act as a block server. One database record of it has three fields: an 8KB binary field as one encrypted file block (needn't to be really as long as 8KB for making experiment), a 160bit binary field stores one E-MAC, and a block number field of 24bit (used for addressing index).

Performance is deduced by comparing the delay introduced by security mechanism with the total time of completing file data exhibition. We don't see any noticeable performance decline in common cases, such as load a document from server. Hash function can have a throughput more than several hundreds Mbps with FPGA realization, and Gbps if considering to operate at the frequency of processor. Even purely implemented by software, hash throughput of tens MB/s is easily to be achieved. The encryption processes only need the cost of XOR operation when converting plaintext into ciphertext, because keystream can be pre-computed. Fetching a value of TS from its storage place can be done concurrently with the reading from server, and will be finished ahead of the returned data in most cases. So, decryption keystream can also be prepared in advance.

The main cost of such protection is that Agent will consume some disk space of client to store values of TS. For an 80GB protected remote storage space and 8KB bock size, it maintains *(80GB/8KB = 10M)* blocks. Each block has a TS with the length of 64bit and a ID of 24 bits, so storage of TS requires *((64b+24b)*10M = 110MB)*. Others like TS and keystream generation, hash logic, and XOR operation, can all be implemented by software (They can also be accelerated by hardware).

File share has a limitation in such security architecture. If two clients cannot communicate with each other, one client cannot read a file of another from server directly, because it cannot decrypt data and interpret data without knowing about related key-stream and meta-data. Another limitation is that it cannot rollback an invalid data block to a correct one when it detects a violation of security.

### 4.2 Guard Large Hard Disk with Small Removable Disk

Today, removable disk of flash memory is becoming an essential commodity for computer users. Usually, it plugs into USB ports and provides several tens megabytes to several gigabytes of removable and easily transportable storage. Such devices can give us security benefits.

Considering the case that you will leave office-room and you want nobody to pry or tamper your important files left in your computer when you are absent, you can apply our protection to achieve this purpose. That is, you can store TS values of each protected file blocks to this removable disk and take this device home (Supposing the RootKey cannot be accessed by anybody, including adversary; or else, you should remove RootKey to this removable disk also to give little chance to cracking the encryption process). After you returned back to office-room and plugging this removable disk into USB ports again, Agent can fetch related TS values from it, decrypts data blocks of your secret files, and checks whether there exist malicious modifications when you staying in your home. In such way, any files you want to be protected cannot be understood by others, and tampering to them can be found immediately when you open them again. This application case is shown in fig.3.

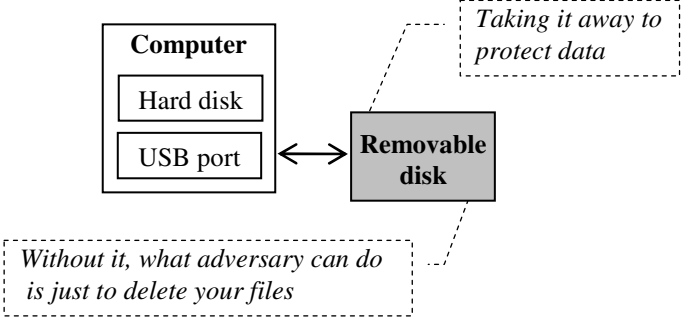



**Fig. 3.** Using a small flash disk to guard large hard disk. Adversary cannot uncover or modify your important file data

In fig.3, supposing your hard disk has a protected partition of 4GB, and each cluster in this partition has a size of 4KB (one commonly used file block size). Then, a total number of *(4GB/4KB = 1M)* data clusters need to be protected. Each TS value is 64bit, and 20bit is used for cluster number. So, removable disk of flash memory will take out *((64b+20b)*1M)*, about 11MB, to hold these TS values. This cost can be acceptable, as most of such removable disks have a capacity more than 64MB.

### 4.3 Trusted Multi-site Download

To improve usability and speed, web content download is often provided by multi-sites, where several web sites mirror the same content of the main site. This brings more possibilities that distributed content may be disturbed, because adversary may compromises one or more mirror sites. Ensuring a trusted multi-site download can be achieved as the followings.

Content provider (i.e., the main site) splits the content data into blocks. Then, it calculates MACs for each data block, and generates a unique value of TS for each *(D-Data, D-MAC)* pair. With one unique TS value, it produces a unique keystream to encrypt each *(D-Data, D-MAC)* to be *(E-Data, E-MAC)*, and sends the corresponding *(TS, E-Data, E-MAC)* pairs as E-Blocks to its mirror sites. After finishing content distributing to its mirror sites, the main site preserves these E-Blocks only; then, builds a secure communication with client (such as establishing a secure TCP/IP connection, or using an out-of-band channel), transfers its secret RootKey, TS sequence, as well as the authentication code of the TS sequence (i.e., a MAC calculated on the concatenation of all the values of TS, which are sorted by the time order of their generation), to client.

Client verifies the TS sequence, and uses TS as meta-data to retrieves E-Blocks from each available site (including ustrusted mirror sites). After one *(E-Data, E-MAC)* pair reached, client decrypts it through the corresponding TS value, and verifies the attached MAC. Valid data is accepted by client to assemble the content.

Such download protection has a concise architecture shown in fig.4.

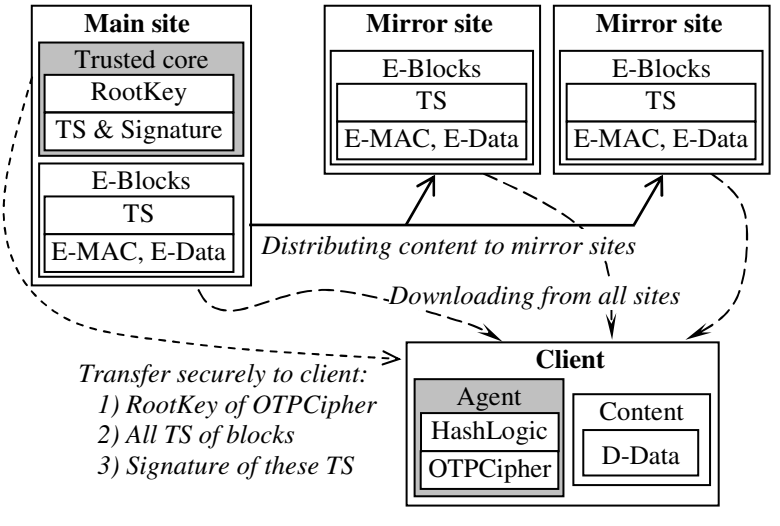



**Fig. 4.** Making multi-site download to be trusted. Compromised mirror sites won't break data security

In fig.4, as mirror sites are refused to be provided with RootKey, understanding the content on them is impossible; also, any tampering to content can incur a mismatch in client. Through such way, security of content distribution is protected.

In this application, values of TS are opened without protection (because TS values are also placed on untrusted mirror sites). As the content data is read only, maintaining the correctness of each TS values and their order is enough to detect any tampering (correctness is ensured by the authentication code of the TS sequence). Also, we can hide TS from mirror sites and use block number as meta-data.

## 5 Conclusions

Our approach can make difficult protecting task to become an easy one in many application situations. By holding a little amount of unique values of timestamp in trusted Agent, untrusted mass data can be protected. Our design integrates the principle of OTP scheme with the idea of verifying integrity individually to achieve solid protection and high performance. Additionally, the cost of securely holding timestamps can be acceptable in many cases. In this paper, we elaborate the related model, approach and specific application examples. As we have demonstrated and discussed, it is a practical and available way to protect mass data against disclosure and tampering.

## References

1. R. C. Merkle.: Protocols for public key cryptography. IEEE Symposium on Security and Privacy, Oakland, California, (1980) 122-134
2. M. Blum, W. S. Evans, P. Gemmell, S. Kannan, and M. Naor.: Checking the correctness of memories. IEEE Symposium on Foundations of Computer Science, San Juan, Puerto Rico (1991) 90-99
3. B. Gassend, G. E. Suh, D. Clarke, M. van Dijk, and S. Devadas.: Caches and merkle trees for efficient memory authentication. Ninth International Symposium on High Performance Computer Architecture, Anaheim, California (2003) 8-12
4. G. E. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas.: Hardware Mechanisms for Memory Integrity Checking. MIT Laboratory for Computer Science, MIT LCS TR-872 (2003)
5. M. Blaze.: A cryptographic file system for unix. 1st ACM Conference on Communications and Computing Security (1993) 9-16
6. Tripwire. http://www.tripwire.org
7. K. Fu, F. Kaashoek, and D. Mazieres.: Fast and secure distributed read-only file system. In Proceedings of OSDI 2000 (2000)
8. D. Mazieres and D. Shasha.: Don't trust your file server. In Proceedings of the 8th Workshop on Hot Topics in Operating Systems (2001)
9. U. Maheshwari, R. Vingralek, and W. Shapiro.: How to Build a Trusted Database System on Untrusted Storage. In Proceedings of OSDI 2000 (2000)
10. C. A. Stein, J. H. Howard, and M. I. Seltzer.: Unifying file system protection. In 2001 USENIX Annual Technical Conference (2001) 79-90
11. Fujita Tomonori and Ogawara Masanori.: Protecting the Integrity of an Entire File System. First IEEE International Workshop on Information Assurance (2003) 95-105

12. SHA-2 Standard. National Institute of Standards and Technology (NIST), Secure Hash Standard, FIPS PUB 180-2, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf
13. R. J. Anderson.: Security Engineering: A Guide to Building Dependable Distributed Systems. John Wiley and Sons (2001)
14. Fangyong Hou, Zhiying Wang, Yuhua Tang, Zhen Liu.: Protecting Integrity and Confidentiality for Data Communication. Ninth IEEE Symposium on Computers and Communications, Alexandria, Egypt (2004) 357-362
15. Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu.: Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. Crypto2004 (2004)

# On the Security of Some Nonrepudiable Threshold Proxy Signature Schemes*

Zuowen Tan[1,2], Zhuojun Liu[1,2], and Mingsheng Wang[3]

[1] Key Laboratory of Mathematics-Mechanization, AMSS, CAS,
Beijing 100080, P.R. China
ztan@163.com
[2] State Key Laboratory of Information Security,Graduate School of CAS,
Beijing 100039, P.R. China
zliu@mmrc.iss.ac.cn
[3] Institute of Software, Chinese Academy of Sciences,
Beijing 100080, P.R. China
mingsheng-wang@yahoo.com.cn

**Abstract.** A $(t, n)$ threshold proxy signature scheme enables an original signer or a group of original signers to delegate the signature authority to a proxy group of $n$ members such that not less than $t$ proxy signers can cooperatively sign messages on behalf of the original signer or the original signer group. In the paper, we show that Sun's and Yang et al.'s threshold proxy signature schemes are insecure against the original signer's forgery, and that Tzeng et al.'s threshold multi-proxy multi-signature scheme is vulnerable against the actual original signer group's forgery. We also show that Hsu et al.'s threshold proxy signature scheme suffers from the conspiracy of the original signer and the secret share dealer SA, and that Hwang et al.'s threshold proxy signature scheme is universally forgeable. In other words, none of the above-mentioned schemes holds the unforgeability and provides non-repudiation.

## 1 Introduction

In a proxy signature scheme, an original signer delegates a user, called a proxy signer, to sign message on its behalf. Since Mambo et al. introduced the concept of proxy signature [9], many proxy signature schemes have been proposed ([1],[8],[10],[11]). Mambo et al.'s schemes [9] satisfy the following property: no one except the original signer and the proxy signer can create a valid proxy signature on behalf of the original signer. Lee et al. [8] improved the security property of the proxy signature: only the proxy signer can create a valid proxy signature and anyone else, even the original signer, can not generate a valid proxy signature. Thus, for a valid proxy signature, the actual proxy signer cannot deny that he/she has signed the message and the original signer cannot deny

* Partially Supported by NSFC(10371127) and NKBRPC(2004CB318000).

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 374–385, 2005.

that he/she has delegated the signing authority to the actual proxy signer. That is, the proxy signature scheme holds the security property non-repudiation.

Based on secret sharing schemes ([12],[13],[15]) and threshold cryptosystems [2], Zhang and Kim et al. independently proposed threshold proxy signature schemes [7],[22], respectively. In a $(t, n)$ threshold proxy signature scheme, a proxy signature key is shared among the subset of the $n$ proxy signers such that only $t$ or more than $t$ proxy signers can cooperatively sign messages. To avoid dispute about who are the actual signers, Sun proposed a nonrepudiable threshold proxy signature scheme with known signers (Sun-scheme [16]). The Sun-scheme eliminates Kim et al.'s scheme's disadvantage that the verifier is unable to determine whether the proxy group key is generated by the legal proxy group. However, Hsu et al. [4] showed that the Sun-scheme is vulnerable against the conspiracy attack: any $t$ or more than $t$ proxy signers can obtain the secret keys of other proxy signers. Hsu et al. proposed a scheme (HWW scheme [4]). Yang et al. [21] made an improvement on the HWW scheme. Yang et al.'s scheme (YTH scheme [21]) is more efficient in terms of computational complexity and communication cost. Hwang et al. proposed a nonrepudiable threshold proxy signature scheme (HLL scheme [5]). Tzeng et al. [17] and Yang et al. [20] found that in the HLL scheme, a malicious original signer can forge the proxy signatures. Yang et al. [20] also showed that Tzeng et al.'s improvement on the HLL scheme still suffers from the original signer's forgery attack. Tzeng et al. [18] extend threshold proxy signature schemes with only an original signer to threshold multi-proxy multi-signature schemes with shared verification and a group of original signers (TYH scheme). In the TYH scheme, a subset of original signers can authenticate a designated proxy group to generate a proxy signature so that only subsets of the specified verifiers can verify the proxy signature. All these schemes above are based on the discrete logarithm cryptosystems. Recently, Hwang et al. proposed an RSA-based threshold proxy signature scheme [6]. Unfortunately, Wang et al. showed that it is insecure [19].

In this paper, we will show a different insider attack against the Sun-scheme from the conspiracy attack proposed by Hsu et al. [4], with which a malicious original signer can forge a proxy signature on any message. The HLL scheme suffers from the original signer's forgery attack as shown in [20] and [18], but we further find that the HLL scheme is universally forgeable. In addition, we show that the HWW scheme, which is the revised scheme of the Sun-scheme, is vulnerable to the conspiracy attack of the original signer and the secret share dealer SA. We also find that the YTH scheme, which is claimed to improves the HWW scheme, is still insecure. We present a security analysis of the newly proposed TYH scheme.

## 2 Preliminaries

### 2.1 Notations

We will use the notations throughout this paper.

-$p$, $q$: two large prime numbers, $q \mid p-1$.
-$g$: an element of $\mathbb{Z}_p^*$, its order is $q$.

-$O_i$: the original signer.
-$(x_{o_i}, y_{o_i})$: $O_i$'s secret/public key pair, $y_{o_i} = g^{x_{o_i}} \pmod{p}$.
-$(x_i, y_i)$: the proxy signer $P_i$'s secret/public key pair , $y_i = g^{x_i} \pmod{p}$.
-$APSID$: the group of the actual proxy signers, or their identities.
-$AOSID$: the group of the actual original signers, or their identities.
-$m_w$: a warrant which records the type of the information delegated, the signers' identities and the period of delegation, etc.

### 2.2 Pedersen's Threshold Distributed Key Generation Protocol

Assume that $\{P_1', P_2', \cdots, P_n'\}$ are $n$ players. $\mathcal{PTDK}$ protocol [14] comprises $n$ Feldman's $(t, n)$ verifiable secret sharing schemes [3] and the three stages.

**(1)** Each $P_i'$ randomly chooses a polynomial $f_i(z)$ over $\mathbb{Z}_q$ of degree $t-1$.

$$f_i(z) = a_{i0} + a_{i1}z + a_{i2}z^2 + \cdots + a_{i,t-1}z^{t-1}. \tag{1}$$

$P_i'$ computes and sends $f_i(j) \pmod{q}$ to $P_j'$ $(j = 1, 2, \cdots, n,\ j \neq i)$ in a secure manner. Then $P_i'$ broadcasts $g^{a_{i0}}, g^{a_{i1}}, \cdots, g^{a_{i,t-1}}$.

**(2)** Each $P_j'$ verifies the validity of the share $f_i(j) \pmod{q}$ by checking:

$$g^{f_i(j)} = g^{a_{i0}}(g^{a_{i1}})^j (g^{a_{i2}})^{j^2} \cdots (g^{a_{i,t-1}})^{j^{t-1}} \pmod{p}.$$

If all $f_i(j)$ are valid, $P_j'$ computes $v_j = \sum_{i=1}^{n} f_i(j) \pmod{q}$ as his share.

**(3)** Let $f(z) = v + a_1 z + a_2 z^2 + \cdots + a_{t-1}z^{t-1} = \sum_{i=1}^{n} f_i(z) \pmod{q}$. If any $t$ secret shares, say $v_1, v_2, \cdots, v_t$, are given, the shared secret key $v$ can be reconstructed by the Lagrange interpolating polynomial:

$$v = f(0) = \sum_{j=1}^{t} v_j \prod_{i=1, i\neq j}^{t} \frac{(0-i)}{(j-i)} \pmod{q}. \tag{2}$$

## 3 Cryptanalysis of Some Threshold Proxy Signature Schemes

### 3.1 On the security of the Sun-Scheme

**Review of the Sun-Scheme.** The Sun-scheme [16] comprises the following phases.

**[ Secret Share Generation Phase]**
The proxy group $\{P_1, P_2, \cdots, P_n\}$ needs to generate a group private/public key pair $(v, y_G) \in \mathbb{Z}_q^* \times \mathbb{Z}_p$. The proxy group run $\mathcal{PTDK}$ protocol. Here, each $P_i$ uses $f_i(z) = x_i + a_{i1}z + a_{i2}z^2 + \cdots + a_{i,t-1}z^{t-1}$. Therefore, the secret key shared is

$v = \sum_{i=1}^{n} x_i$, and the corresponding public key is $y_G = \prod_{i=1}^{n} y_i \pmod p$. $P_i$ obtains a secret key share $v_i = \sum_{l=1}^{n} f_l(i) \pmod q$. Let $A_j = g^{a_j} \pmod p, 1 \le j \le t-1$.

**[Proxy Share Generation Phase]**
First, $O$ randomly chooses $k \in \mathbb{Z}_q$, and computes $K = g^k \pmod p$, $\sigma = x_0 h(m_w \,||K) + k \pmod q$. Next, $O$, as a dealer, distributes the proxy key $\sigma$ among the proxy group by executing Feldman's VSS scheme [3]. In particular, $O$ randomly chooses a polynomial of degree $t-1$: $f'(z) = \sigma + b_1 z + b_2 z^2 + \cdots + b_{t-1} z^{t-1}$, computes and secretly sends $\sigma_i = f'(i) \pmod q$ to the proxy signer $P_i$ $(i = 1, 2, \cdots, n)$. $O$ publishes $(m_w, K)$ and $B_j = g^{b_j}$ $(j = 1, 2, \cdots, t-1)$.

$P_i$ accepts $(\sigma_i, m_w, K)$ if the equation $g^{\sigma_i} = y_0^{h(m_w||K)} K \prod_{j=1}^{t-1} B_j^{i^j} \mod p$ holds. Finally, $P_i$ computes his proxy share $\sigma_i' = \sigma_i + v_i \cdot h(m_w||K) \pmod q$.

**[Proxy Signature Generation Phase]**
Without loss of generality, we assume that $\{P_1, P_2, \cdots, P_t\}$attempt to sign a message $m$ as the actual proxy group.

First, the $t$ proxy signers execute $\mathcal{PTDK}$ protocol for sharing a random number $c_0 = \sum_{i=1}^{t} c_{i,0}$ by using $f_i''(z) = (c_{i,0} + x_i) + c_{i,1} z + c_{i,2} z^2 + \cdots + c_{i,t-1} z^{t-1} \pmod q$. Thus, each $P_i$ obtains the public value $y = g^{c_0} \pmod p, C_j = g^{c_j} \pmod p$ and the secret random number share $v_i' = f''(i) = \sum_{i=1}^{t} x_i + c_0 + c_1 i + c_2 i^2 + \cdots + c_{t-1} i^{t-1} \pmod q$, where $c_j = \sum_{i=1}^{t} c_{ij}$ for $1 \le j \le t-1$.

Next, $P_i$ computes his proxy signature share $s_i = v_i' y + \sigma_i' h(AOSID||m) \pmod q$ and sends $s_i$ to the proxy signers $P_j$ $(j = 1, 2, \cdots, t, j \ne i)$ in a secure manner. $P_j$ can verify the validity of $s_i$ by checking if the following equation holds:

$$g^{s_i} = \left[ y \left( \prod_{j=1}^{t-1} C_j^{i^j} \right) \left( \prod_{j=1}^{t} y_j \right) \right]^{y} \left[ \left( K y_0^{h(m_w||K)} \prod_{j=1}^{t-1} B_j^{i^j} \right) \cdot \left( y_G \prod_{j=1}^{t-1} A_j^{i^j} \right)^{h(m_w||K)} \right]^{h(APSID||m)} \mod p. \tag{3}$$

Each signer in the actual proxy group can generate $s = f''(0) y + [f(0) + f'(0)] h(APSID||m)$ by employing the Lagrange interpolation formula to $s_i$. The proxy signature on $m$ is $(m, m_w, K, APSID, y, s)$.

**[Proxy Signature Verification Phase]**
Any verifier can identify the original signer and the actual proxy signers from $m_w$ and $APSID$, and validate the proxy signature by checking if

$$g^s = \left[ K y_0^{h(m_w||K)} \prod_{i=1}^{n} y_i \right]^{h(APSID||m)} \left( y \prod_{i=1}^{t} y_i \right)^{y} \mod p. \tag{4}$$

**Cryptanalysis of the Sun-Scheme.** In the Sun-scheme, a malicious original signer can generate a proxy signature on any message $m$ and claim that any $t$ or more than $t$ proxy signers are the actual proxy signers of the proxy signature. Assume that $O$ chooses a proxy group $APSID = \{P_1, P_2, \cdots, P_t\}$. $O$ randomly chooses $\alpha \in_R \mathbb{Z}_q, \beta \in_R \mathbb{Z}_q$ and computes $K = (\prod_{i=1}^{n} y_i)^{-1} g^{\alpha} \pmod p$, $y = (\prod_{i=1}^{t} y_i)^{-1} g^{\beta} \pmod p$. Then $O$ computes

$$s = (\alpha + x_0 h(m_w||K))h(APSID||m) + \beta y \pmod q. \tag{5}$$

Then, $(m, m_w, K, APSID, y, s)$ is a valid proxy signature. This is because:

$$\begin{aligned} g^s &= (g^{\alpha} g^{x_0 h(m_w||K)})^{h(APSID||m)} (g^{\beta})^y \pmod p \\ &= [K y_0^{h(m_w||K)} \prod_{i=1}^{n} y_i]^{h(APSID||m)} (y \prod_{i=1}^{t} y_i)^y \pmod p. \end{aligned}$$

### 3.2 On the Security of the HWW Scheme

**Review of the HWW Scheme.** The HWW scheme [4] is as follows.

**[Secret Share Generation Phase]**
In order to reduce the computation and communication cost in [4], SA is responsible for performing the secret share generation. SA chooses a proxy group private/public key pair $(x_{SA}, y_{SA}) \in \mathbb{Z}_q^* \times \mathbb{Z}_p$, where $y_{SA} = g^{x_{SA}} \pmod p$, and randomly generates a $(t-1)$-degree polynomial in $\mathbb{Z}_q[z]$:

$$f(z) = x_{SA} + a_1 z + a_2 z^2 + \cdots + a_{t-1} z^{t-1} \pmod q. \tag{6}$$

SA computes and sends $v_i = f(i) \pmod q$ as $P_i$'s secret share in a secure manner, and then publishes the corresponding value $g^{v_i} \pmod p$.

**[Proxy Share Generation Phase]**
$O$ generates the proxy share in the same way as in the Sun-scheme. $O$ computes and sends $\sigma_i = f'(i) \pmod q$ in a secure manner to $P_i$. Finally, $P_i$ computes $\sigma_i' = \sigma_i + v_i \cdot h(m_w||K) \pmod q$ as his proxy share.

**[Proxy Signature Generation Phase]**
Assume that the actual proxy group $\{P_1, P_2, \cdots, P_t\}$ cooperatively sign $m$.

First, each $P_i$ randomly chooses $k_i \in \mathbb{Z}_q^*$, broadcasts $r_i = g^{k_i} \pmod p$, and computes

$$R = \prod_{j=1}^{t} r_j \pmod p, s_i = k_i R + (L_i \sigma_i' + x_i) h(R||APSID||m) \pmod q,$$

where $L_i$ is a Lagrange coefficient. $P_i$ sends his individual proxy signature $s_i$ to the designated clerk. If the following equation holds for $1 \le i \le t$,

$$g^{s_i} = r_i^R \left( \left( (y_0 g^{v_i})^{h(m_w||K)} \left( \prod_{j=1}^{t-1} B_j^{i^j} \right) K \right)^{L_i} y_i \right)^{h(R||APSID||m)} \pmod p,$$

the designated clerk computes $s = \sum_{i=1}^{t} s_i \pmod q$. The proxy signature on $m$ is $(m, m_w, K, APSID, R, s)$.

**[Proxy Signature Verification Phase]**
Any verifier can identify the original signer and the actual proxy signers from $m_w$ and $APSID$, and validate the proxy signature by checking if

$$g^s = R^R \left( K(y_0 y_{SA})^{h(m_w||K)} \prod_{i=1}^{t} y_i \right)^{h(R||APSID||m)} \pmod p. \tag{7}$$

**Cryptanalysis of the HWW Scheme.** We will show that the HWW scheme is vulnerable against the conspiracy of the original signer and SA. A malicious original signer and SA can cooperatively generate a proxy signature on any message and claims that any $t$ or more than $t$ proxy signers are the actual proxy signers. In order to forge a proxy signature on any message $m$, $O$ chooses an "actual" proxy group $\{P_1, P_2, \cdots, P_t\}$, then computes $R = g^\beta$, $K = (\prod_{i=1}^{t} y_i)^{-1} g^\alpha \pmod p$, where $\alpha, \beta \in_R \mathbb{Z}_q^*$. After obtaining $x_{SA}$ from SA, $O$ computes

$$s = [\alpha + (x_0 + x_{SA})h(m_w||K)]h(R||APSID||m) + \beta R \pmod q. \tag{8}$$

Then, $(m, m_w, K, APSID, R, s)$ is a valid proxy signature. This is because:

$$\begin{aligned} g^s &= (g^\beta)^R [g^\alpha (y_0 y_{SA})^{h(m_w||K)}]^{h(R||APSID||m)} \pmod p \\ &= R^R \left( K(y_0 y_{SA})^{h(m_w||K)} \prod_{i=1}^{t} y_i \right)^{h(R||APSID||m)} \pmod p. \end{aligned}$$

### 3.3 On the Security of the YTH Scheme

**Review of the YTH Scheme.** The scheme [21] is composed of three phases.

**[Proxy Share Generation Phase]**
$O$ randomly chooses $k \in \mathbb{Z}_q$, and computes $K = g^k \pmod p$ and the proxy signature key $\sigma = x_0 h(m_w||K) + k \pmod q$. Then, $O$ broadcasts $(\sigma, m_w, K)$ to the proxy group $\{P_1, P_2, \cdots, P_n\}$.

$P_i$ uses $\sigma$ as the proxy share if the equation $g^\sigma = K y_0^{h(m_w||K)} \pmod p$ holds.

**[Proxy Signature Generation Phase]**
Without loss of generality, let $\{P_1, P_2, \cdots, P_t\}$ be the actual proxy signers.

First, each $P_i$ broadcasts $r_i = g^{k_i}$, where $k_i \in_R \mathbb{Z}_q^*$. Then $P_i$ computes

$$R = \prod_{j=1}^{t} r_j \pmod p, s_i = k_i R + (t^{-1}\sigma + x_i)h(R||APSID||m) \pmod q.$$

$P_i$ sends the individual proxy signature $s_i$ to the designated clerk. For each $i$, the clerk validates the individual proxy signature $s_i$ by checking:

$$g^{s_i} = r_i^R \left( ((K y_0^{h(m_w||K)})^{t^{-1}} y_i \right)^h (R||APSID||m) \pmod{p}. \tag{9}$$

If all $s_i$ on message $m$ are valid, the designated clerk computes $s = \sum_{i=1}^{t} s_i \pmod{q}$. $(m, m_w, K, APSID, R, s)$ is the proxy signature.

**[Proxy Signature Verification Phase]**
Any verifier can identify the original and the actual proxy signers from $m_w$ and $APSID$, and validate the proxy signature through the following equation.

$$g^s = R^R \left( K y_0^{h(m_w||K)} \prod_{i=1}^{t} y_i \right)^{h(R||APSID||m)} \pmod{p}. \tag{10}$$

**Cryptanalysis of the YTH Scheme.** We show that the YTH scheme is insecure against the original signer's forgery. In order to forge a proxy signature on any message $m$, a malicious original signer $O$ chooses a proxy group $APSID$ of not less than $t$ proxy signers. Assume the proxy group is $\{P_1, P_2, \cdots, P_t\}$. $O$ randomly chooses $\alpha$ and $\beta$ in $\mathbb{Z}_q^*$, and computes

$$K = (\prod_{i=1}^{t} y_i)^{-1} \cdot g^{\alpha} \pmod{p}, R = g^{\beta} \pmod{p}, \tag{11}$$

$$s = (\alpha + x_0 h(m_w||K)) h(R||APSID||m) + \beta R \pmod{q}. \tag{12}$$

Then, $(m, m_w, K, APSID, R, s)$ is a valid proxy signature on message $m$. This is because:

$$\begin{aligned} g^s &= g^{\beta R} (g^{\alpha} g^{x_0 h(m_w||K)})^{h(R||APSID||m)} \mod p \\ &= R^R (K y_0^{h(m_w||K)} \prod_{i=1}^{t} y_i)^{h(R||APSID||m)} \mod p. \end{aligned}$$

### 3.4 On the Security of the HLL Scheme

**Review of the HLL Scheme.** The HLL scheme comprises of the phases.

**[Secret Share Generation Phase]**
The proxy group generates a group private/public key pair $(v, y_G)$ as in the Sun-scheme. Here, each $P_i$ uses $f_i(z) = x_i + a_{i0} + a_{i1} z + a_{i2} z^2 + \cdots + a_{i,t-1} z^{t-1}$. The secret key shared is $v = \sum_{i=1}^{n} x_i$ and the public key is $y_G = \prod_{i=1}^{n} y_i \pmod{p}$. Each proxy signer $P_i$ obtains a secret key share $v_i = f(i) = \sum_{j=1}^{n} f_j(i) \pmod{q}$. The group publishes $A_j = g^{a_j} \pmod{p}, j = 0, 1, 2, \cdots, t-1$.

**[Proxy Share Generation Phase]**
$O$ first generates the proxy key $\sigma = h(m_w||K)x_0 + k(\text{mod } q)$, then distributes the proxy key $\sigma$ among the proxy group by executing Feldman's VSS scheme as follows. $O$ randomly chooses a polynomial of degree $t-1$:

$$f'(z) = \sigma + b_1 z + b_2 z^2 + \cdots + b_{t-1} z^{t-1} \pmod q.$$

$O$ computes and secretly sends $\sigma_i = f'(i) \pmod q$ to $P_i$ for $i = 1, 2, \cdots, n$. $O$ publishes $(m_w, K)$ and $B_j = g^{b_j} \pmod p$ for $j = 1, 2, \cdots, t-1$ .

$P_i$ accepts $(\sigma_i, m_w, K)$ if the equation holds.

$$g^{\sigma_i} = y_0^{h(m_w||K)} K \prod_{j=1}^{t-1} B_j^{i^j} \pmod p.$$

Then $P_i$ computes $\sigma_i' = \sigma_i + v_i \cdot h(m_w||K) \pmod q$ as his proxy share.

**[Proxy Signature Generation Phase]**
Assume that $\{P_1, P_2, \cdots, P_t\}$ are the actual proxy group. $P_i$ first generates the secret random share $v_i'$ as $P_i$ does in the Sun-scheme. Then $P_i$ computes the individual proxy signature $s_i = v_i' y + \sigma_i' h(APSID||m) \pmod q$ and sends $s_i$ to the proxy signers $P_j$ $(j = 1, 2, \cdots, t, j \neq i)$ in a secure manner. $P_j$ can verify the validity of $s_i$ by checking if the following equation holds:

$$g^{s_i} = \left[ y \left( \prod_{j=1}^{t-1} C_j^{i^j} \right) \left( \prod_{j=1}^{t} y_j \right) \right]^y \left[ \left( K y_0^{h(m_w||K)} \prod_{j=1}^{t-1} B_j^{i^j} \right)^{h(APSID||m)} \cdot \left( y_G A_0 \prod_{j=1}^{t-1} A_j^{i^j} \right)^{h(m_w||K)} \right] \pmod p. \tag{13}$$

By the Lagrange interpolation formula, any actual proxy signer can generate

$$s = f''(0) y + [f(0) + f'(0)] h(APSID||m).$$

The proxy signature on $m$ is $(m, m_w, K, APSID, y, A_0, s)$.

**[Proxy Signature Verification Phase]**
Any verifier can identify the original signer and the actual proxy signers from $m_w$ and $APSID$, and validate the proxy signature from the equation:

$$g^s = \left[ K A_0 y_0^{h(m_w||K)} \prod_{i=1}^{n} y_i \right]^{h(APSID||m)} \left( y \prod_{i=1}^{t} y_i \right)^y \pmod p. \tag{14}$$

**Cryptanalysis of the HLL Scheme.** We show that the HLL scheme is universally forgeable. Any adversary can impersonate any original signer and any $t$ or more than $t$ proxy signers to forge a proxy signature on any message. Given any message $m$ and any proxy group $\{P_1, P_2, \cdots, P_n\}$, the adversary chooses $\{P_1, P_2, \cdots, P_t\}$ as the actual proxy signers. The adversary first randomly chooses $\alpha$, $\beta$, $\gamma$ in $\mathbb{Z}_q^*$, and $y \in \mathbb{Z}_p^*$ and computes

$$K = (\prod_{i=1}^{n} y_i)^{-1} g^{\alpha} \pmod{p}, A_0 = (y_0^{h(m_{w||K})})^{-1} g^{\beta} \pmod{p} \tag{15}$$

$$s = (\alpha + \beta) h(APSID||m) + \gamma y \pmod{q}. \tag{16}$$

Then, $(m, m_w, K, APSID, y, A_0, s)$ is a valid proxy signature on message $m$. This is because it satisfies the following verification equation:

$$\begin{aligned} g^s &= \left(g^{\alpha} g^{\beta}\right)^{h(APSID||m)} g^{\gamma y} \pmod{p} \\ &= \left[K A_0 y_0^{h(m_w||K)} \prod_{i=1}^{n} y_i\right]^{h(APSID||m)} \left(y \prod_{i=1}^{t} y_i\right)^{y} \pmod{p}. \end{aligned}$$

### 3.5 On the Security of the TYH Scheme

**Review of the TYH Scheme.** The TYH scheme [18] involves four parties: the trusty share distribution center(SDC), the original group $G_o = \{O_1, O_2, \cdots, O_{n_1}\}$, the proxy group $G_p = \{P_1, P_2, \cdots, P_{n_2}\}$ and the verifier group $G_v = \{V_1, V_2, \cdots, V_{n_3}\}$. Each $V_i$ has private/public key pair $(x_{v_i}, y = g^{x_{v_i}} \pmod{p})$. The group $G_p$ and $G_v$ have private/public key pair $(x_P, y_P)$ and $(x_V, y_V)$, respectively, where $y_P = g^{x_P} \pmod{p}$, $y_V = g^{x_V} \pmod{p}$. These public keys are chosen by SDC and certified by CA. Let $t_1$, $t_2$ and $t_3$ be the threshold values about the original signer group, the proxy signer group, and the verifier group, respectively. The TYH scheme contains four phases.

**[Secret Share Generation Phase]**
SDC randomly chooses a polynomial $f_p(x) \in \mathbb{Z}_q[x]$ with degree $t_2 - 1$ and $f_v(x) \in \mathbb{Z}_q[x]$ with degree $t_3 - 1$ such that $f_p(0) = x_P$, $f_v(0) = x_V$. Then SDC distributes the secret shadow $f_p(y_{p_i})$ to $P_i$ and publishes $y_{fP_i} = g^{f_p(y_{p_i})}$, $1 \le i \le n_2$. Similarly, SDC distributes the secret shadow $f_v(y_{v_j})$ to $V_j$ and publishes $y_{fv_j} = g^{f_v(y_{v_j})}$, $1 \le j \le n_3$.

**[Proxy Share Generation Phase]**
Assume that $\{O_1, O_2, \cdot, O_{t_1}\}$ are the actual original signers AOSID.

Each $O_j$ in the AOSID chooses a random number $a_j \in \mathbb{Z}_q^*$ and broadcasts $k_j = g^{a_j} \pmod{p}$. Upon receiving $k_j$, $O_i$ computes

$$K = \prod_{i=1}^{t_1} k_i \pmod{p}, \sigma_i = a_i K + x_{o_i} h(K||m_w||AOSID) \pmod{q}.$$

$O_i$ sends $\sigma_i$ to the designated clerk. The clerk verifies its validity by checking

$$g^{\sigma_i} = k_i^K y_{o_i}^{h(K||m_w||AOSID)} \pmod{p}.$$

Then the clerk computes $\sigma = t_2^{-1} \sum_{i=1}^{t_1} \sigma_i \pmod{q}$ and broadcasts $(\sigma, m_w, K, AOSID)$ to $G_p$. Each $P_j \in G_p$ uses $\sigma$ as the proxy share.

**[Proxy Signature Generation Phase]**
Assume that $\{P_1, P_2, \cdots, P_{t_2}\}$ be the actual proxy group APSID.

First, each $P_k$ in the APSID chooses a random number $b_k \in \mathbb{Z}_q^*$ and broadcast $r_{P_k} = g^{b_k} (\text{mod } p)$. Next, each $P_k$ computes and broadcasts $r'_{p_k}$:

$$r'_{p_k} = (y_V)^{f_p(y_{p_k}) \prod_{l=1, l\neq k}^{t_2} (0 - y_{P_l})/(y_{p_k} - y_{p_l})} (\text{mod } p).$$

After receiving $r_{P_k}$ and $r'_{p_k}$, each $P_j$ in the APSID computes

$$R = \prod_{j=1}^{t_2} r_{P_j} (\text{mod } p), R' = \prod_{j=1}^{t_2} r'_{P_j} (\text{mod } p),$$

$$s_j = R' f_p(y_{p_j}) \prod_{k=1, k\neq j}^{t_2} \frac{(0 - y_{P_k})}{(y_{p_j} - y_{p_k})} + b_j R + (\sigma + x_{p_i}) h(R||APSID||m) (\text{mod } q).$$

$P_j$ sends the individual proxy signature to the designated clerk. The clerk validates $s_j$ by checking the equation

$$g^{s_j} = y_{fp_j}^{R' \prod_{k=1, k\neq j}^{t_2} (0 - y_{P_k})/(y_{p_j} - y_{p_k})} r_{p_i}^R \cdot \left( (K^K \prod_{i=1}^{t_1} y_{o_i}^{h(K||m_w||AOSID)})^{t_2^{-1}} y_{p_j} \right)^{h(R||APSID||m} (\text{mod } p). \quad (17)$$

If the equation above holds, the clerk computes $S = \sum_{i=1}^{t_2} s_i (\text{mod } q)$. The proxy signature on message $m$ is $(m_w, K, AOSID, R, S, APSID)$.

**[Proxy Signature Verification Phase]**
Any $t_3$ out of $n_3$ verifiers in the group $G_V$ can cooperatively check the validity of the proxy signature. Let $\{V_1, V_2, \cdots, V_{t_3}\}$ be the actual verifiers.

First, each $V_l$ $(1 \leq l \leq t_3)$ computes and broadcasts $r'_{v_l}$:

$$r'_{v_l} = (y_P)^{f_v(y_{v_l}) \prod_{k=1, k\neq l}^{t_3} (0 - y_{v_k})/(y_{v_l} - y_{p_k})} (\text{mod } p).$$

After receiving $r'_{v_l}$, each $V_l$ computes $R' = \prod_{l=1}^{t_3} r'_{v_l} (\text{mod } p)$. Then each $V_l$ can check the validity of the proxy signature through the equation:

$$g^S = y_p^{R'} R^R (K^K \prod_{i=1}^{t_1} y_{o_i}^{h(K||m_w||AOSID)} \prod_{j=1}^{t_2} y_{p_j})^{h(R||APSID||m)} (\text{mod } p). \quad (18)$$

**Cryptanalysis of the TYH Scheme.** We show that the TYH scheme is insecure against the original signers' forgery. After the malicious original signer group $AOSID$ obtain the proxy signature ($m_w$, $m$, $K$, $AOSID$, $R$, $S$, $APSID$) on message $m$, the original group $AOSID$ can generate another proxy signature on message $m$ without the agreement of the proxy group $APSID$.

The original group first cooperates to restore $\sigma$. Then the original group replaces $m_w$ with $m'_w$, executes all the steps except broadcasting to $G_p$ during

the proxy share generation phase, and obtains $(\sigma', m'_w, K', AOSID)$. Finally, any original signer in the actual original group $AOSID$ can compute

$$S' = S - t_2\sigma h(R||APSID||m) + t_2\sigma' h(R||APSID||m). \tag{19}$$

It is easy to verify that $(m'_w, m, K', AOSID, R, S', APSID)$ is a valid proxy signature.

## 4 Conclusions

In the paper, we analyze the security of some nonrepudiable threshold proxy signature schemes. We find that in the SUN scheme and the YTH scheme, a malicious original signer can forge a valid proxy signature on any message without the agreement of the proxy group. Furthermore, we show that the HLL scheme is universally forgeable, and the HWW scheme is insecure against the conspiracy of the original signer and the secret share dealer SA. In the TYH scheme, a malicious actual original signer group can generate another proxy signature on the same message without the agreement of the same proxy group after the proxy group signs message $m$. These proxy signature schemes cannot fulfil the claimed security properties. Therefore, it is desirable to design efficient nonrepudiable threshold proxy signature schemes which are provably secure in the formal security model proposed in [1] and [10].

## Acknowledgements

The authors are most grateful to the anonymous referees for their suggestions and to G.-L. Wang for his helpful discussions about this work.

## References

1. A.Boldyreva, A. Palacio, B. Warinschi, *Secure Proxy Signature Schemes for Delegation of Signing Rights*, IACR ePrint Archive, available at http://eprint.iacr.org/2003/096, 2003.
2. Y. Desmedt and Y. Frankel, *Threshold Cryptosystems*, Proc. Crypto1989, Lecture Notes in Computer Science, Vol. 435. Springer-Verlag, Berlin Heidelberg New York (1989), pp. 307-315.
3. P. Feldman, *A Practical Scheme for Non–Interactive Verifiable Secret Sharing*, Proc. 28th FOCS, IEEE, pp. 427-437, 1987.
4. C.-L. Hsu, T.-S. Wu, and T.-C. Wu, *New nonrepudiable threshold proxy signature schemem with known signers*, The Journal of Systems and Software 58(2001), pp. 119-124, 2001.
5. M.-S. Hwang, I.-C. Lin and K.-F. Lu, *A secure nonrepudiable threshold proxy signature scheme with known signers*, International Journal of Informatica 11(2), pp.1-8, 2000.
6. M.-S. Hwang, E.J.-L. Lu and I.-C. Lin, *A practical* $(t, n)$ *threshold proxy signature scheme bBased on the RSA cryptosystem*, IEEE Trans. Knowledge and Data Eng., Vol. 15, no. 16, pp.1552-1560, 2003.

7. S. J. Kim, S. J. Park, D. H. Won, *Proxy Signatures, revisited*. ICICS'97, Lecture Notes in Computer Science, Vol. 1334. Springer-Verlag, Berlin Heidelberg New York (1997), pp. 223-232.
8. B. Lee, H. Kim, and K. Kim, *Strong proxy signgture and its applications*, Proceedings of SCIS, 2001, pp. 603-608, 2001.
9. M. Mambo, K. Usuda and E. Okamoto, *Proxy signatures for delegating signing operation*, Proc. 3rd ACM Conference on Computer and Communications Security, ACM Press, pp. 48-57, 1996.
10. Tal Malkin, Satoshi Obana, and Moti Yung, *The hierarchy of key evovlving signatures and a characterization proxy signatures*, Proc. Eurocrypt2004, Lecture Notes in Computer Science, Vol. 3027. Springer-Verlag, Berlin Heidelberg New York (2004), pp. 306-322.
11. H.-U. Park and L.-Y. Lee, *A digital nominative proxy signature scheme for mobile communications*, ICICS 2001, Lecture Notes in Computer Science, Vol. 2229. Springer-Verlag, Berlin Heidelberg New York (2001), pp. 451-455.
12. T. P. Pedersen, *Non-interactive and information-theoretic secure verifiable secret sharing*, Proc. Crypto1991, Lecture Notes in Computer Science, Vol. 576. Springer-Verlag, Berlin Heidelberg New York (1991), pp. 129-140.
13. T. P. Pedersen. *Distributed Provers with Applications to Undeniable Signatures*. Proc. Eurocrypt1991, Lecture Notes in Computer Science, Vol. 547. Springer-Verlag, Berlin Heidelberg New York (1991), pp. 221-242.
14. T. P. Pedersen. *A Threshold Cryptosystem without a Trusted Party*, Proc. Eurocrypt1991, Lecture Notes in Computer Science, Vol. 547. Springer-Verlag, Berlin Heidelberg New York (1991), pp. 522-526.
15. A. Shamir, *How to Share a Secret*, Communications of the ACM, vol.22, No.11, pp. 612-613, 1979.
16. H. M. Sun, *An efficient nonrepudiable threshold proxy signatures with known signers*, Computer Communications 22(8),1999, pp. 717-722.
17. S.-F. Tzeng, M.-S. Hwang, and C.-Y. Yang, *An improvement of nonrepudiable threshold proxy signature schemem with known signers*, Computers & Security 23,2004, pp. 174-178.
18. S.-F. Tzeng, C.-Y. Yang, and M.-S. Hwang, *A nonrepudiable threshold multi-proxy multi-signature scheme with shared verification*, Future Generation Computer Systems 20, 2004, pp. 887-893.
19. G.-L. Wang, F. Bao, J.-Y. Zhou, and Robert H. Deng, *Comments on "A practical* $(t, n)$ *threshold proxy signature scheme based on the RSA cryptosystem"*, IEEE Trans. Knowledge and Data Eng., Vol. 16, no. 10, pp.13092-1311, 2004.
20. F.-Y. Yang, J.-K. Jan, and W.-J. Jeng, *Cryptanalysis of a threshold proxy signature with known signers*, IACR ePrint Archive, available at http://eprint.iacr.org/2004/313, 2004.
21. C.-Y. Yang, S.-F. Tzeng and M.-S. Hwang, *On the efficiency of nonrepudiable threshold proxy signatures with known signers*, The Journal of Systems and Software 22(9),2003, pp. 1-8.
22. K. Zhang, *Threshold proxy signature schemes*, Information Security Workshop, Japan, 1997, pp. 191-197.

# Token-Controlled Public Key Encryption

Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo

Centre for Information Security Research,
School of Information Technology and Computer Science,
University of Wollongong,
Wollongong NSW 2522, Australia
{baek, rei, wsusilo}@uow.edu.au

**Abstract.** "Token-controlled public key encryption" is a public key encryption scheme where individual message can be encrypted and sent to every receiver, but the receiver cannot decrypt the message until he/she is given an extra piece of information called a "*token*". The token will not reveal any information about the messages that the sender originally sent and the communication overhead for releasing the token is very small. Also, it is possible that a single token can control the decryption of a number of ciphertexts sent to multiple receivers. We formalize security model for such scheme and show efficient and provably secure constructions based on known computational assumptions in the random oracle model.

## 1 Introduction

Consider the following scenario. A company ABC wants to send a payslip to its employees, in such a way that each employee can decrypt the authorization code to obtain his/her cash *at the appointed time*. The description of the payslip (such as the amount, the detail of the employee, etc.) can be read when the payslip is received, but the authorization code that will allow the bank to transfer the money to the employee's account will only be available at the appointed time, i.e. on the pay-day. Intuitively, this problem can be solved easily by sending the payslip and authorization code on the pay-day to each employee. However, due to the large number of employees in the company, it would be more convenient if the information can be sent progressively, i.e. throughout the week before the pay day occurs. Nevertheless, the company do not want to allow their employees to obtain their salary before the pay day.

Formally, we would like to obtain a new public key encryption scheme, which we call a "*token-controlled public key encryption scheme*", where individual message can be encrypted and sent to every receiver, but the receiver cannot decrypt the message until he/she is given an extra piece of information, which we call a "*token*", from the third party that was appointed by the sender. Additionally, the token will not reveal any information about the messages that the sender originally sent. We also require that the communication and computational overhead for releasing the token to be small. For example, after its release by the

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 386–397, 2005.

company, *one* token whose size is similar to that of the security parameter within the system, e.g. 1024 bits, can control the decryption of the bank authorization codes different from employee to employee.

We argue that such a scheme has many other real world applications. Suppose there is a millionaire who would like to write a will for his three sons. This will is written, sealed and sent to his sons. However, he will not allow his sons to read the will before he passes away, and therefore, he will send the token used to create the sealed will to a lawyer that he trusts. On one hand, this lawyer cannot read the will because he has no access to this encrypted document, and on the other hand, the three sons cannot read the will since they require the key information that is held by the lawyer.

## 2 Related Work and Our Contributions

*Related Work.* The closest work related to our research is the "timed-release cryptography" a.k.a. "sending information into the future" discussed in the papers of May [8], Rivest et al. [10], and Di Crescenzo et al. [6]. The approaches made in those papers to realize the timed-cryptography can be categorized into a "computational approach" and an "agent-based approach". The idea of the computational approach of [10] is that the time to recover a secret is given by the minimum computational effort required by any machine to perform some computation which will enable one to recover the secret. In [10], a computational primitive based on the hardness of the integer factorization was proposed to realize the idea, subsequently, a timed-release encryption scheme was constructed. However, as discussed in the same paper, this primitive does not give precise time-release as computational capability can vary from machine to machine.

On the other hand, in the agent-based approach, a trusted third party is expected to release a secret at the appointed time. As a result, precise time-release is possible. The schemes in [8] and [6], and the second scheme in [10] follow this approach. In [8], it was suggested that a cryptographic key $K$ should be stored by a trusted agent while the encryption of the message $M$ with $K$, denoted by $C = E(K, M)$, is sent to the receiver. At time $t$, the agent releases $K$ which will enable the receiver to decrypt $C$. However, one of the drawbacks in this approach is that the agent must store the keys for all users. Rivest et al.'s [10] scheme resolves this problem by having the agent use its own key $S$ to encrypt the key $K$ requested by the sender and yield $C' = E(S, K)$. $C = E(K, M)$ and $C'$ are then sent to the receiver. At the time when the agent releases $S$, the receiver recovers $K$ from $C'$ and subsequently decrypt $C$. In our point of view, the main drawback of the approaches of [8] and [10] is that in order to protect confidentiality of the message $M$, the encryption key $K$ or the agent's secret key $S$ should be delivered to the receiver through secure channels after the release-time has passed. If $K$ and $S$ are just published as described in [8] and [10], no secrecy on the message $M$ is guaranteed.

In contrast to the above methods, the approach of [6] does not have this problem. In the timed-release encryption scheme proposed in [6], a plaintext message $M$

is encrypted by the receiver's public key $pk_R$ and the resulting ciphertext is re-encrypted under the agent's public key $pk_A$. The resulting encryption, which we denote by $C = E(pk_A, E(pk_R, M))$, together with the release-time is sent to the receiver. On receiving $C$ and the time information, the receiver interacts with the agent using the computationally intensive "conditional oblivious transfer protocol" which will give the receiver $E(pk_R, M)$ if the release-time is not less than the current time of the agent, otherwise, gives nothing to the receiver. Since the message is encrypted by the receiver's public key and this can only be decrypted by the receiver who has the corresponding private key after the release-time has passed, the problems in [8] and [10] do not happen. This property is in fact what we also need in our token-controlled public key encryption, but other requirements for it is not exactly the same as those for the scheme in [6] as outlined below.

*Our Contributions.* Based on the scenarios illustrated in Section 1 and the discussions on the related work, we summarize the features of our token-controlled public key encryption scheme:

1. A *token* independently chosen from a decryption key (a receiver's private key) is revealed by the third party so that the confidentiality of the plaintext message against parties except for the correct receiver is attained even after the token is released.
2. The receiver does not have to be involved in a computationally-heavy protocol to obtain a token.
3. A token can be *reusable* in the multiple receiver setting. That is, *one token* can control the decryption operation of multiple receivers without compromising security.

In the rest of the paper, we define a formal security model for the single receiver token-controlled public key encryption and then propose efficient and provably secure schemes in the random oracle model [3].

*Remark.* We remark that the security model and the scheme for the single user setting can readily be extended to the multiple receiver setting. In this setting, "token reusability" stated above is achieved. However, we omit them in the current version of the paper due to the lack of space. We also omit the proofs of the theorems presented in the current version, except for the proof of Theorem 1. The full version of this paper will contain all the omitted proofs.

## 3 Formal Security Model for the Single Receiver Setting

*Generic Description.* We begin with a high level description. In a token-controlled public key encryption scheme, the sender first picks a token at random from predefined token space. The sender then encrypts a plaintext message using the receiver's public key and the token, and sends the ciphertext to the receiver. Additionally, the sender gives the token to a "semi-trusted" third party. What we mean by "semi-trusted" here is that the third party is not required to store any private keys from the sender and the receiver, but is required to release a token honestly at the time previously requested by the sender. We can also

assume that this party may behave maliciously to break the confidentiality of a ciphertext given to the receiver. (This will be discussed further in Section 3). *Only when the token is released (or published)* by the third party, the receiver can decrypt the ciphertext.

Compared with the model in [6], our token-controlled public key encryption does not provide "sender-anonymity" against a third party in that the sender must contact the third party to hand in a token. This is different from the model in [6] which resulted in a scheme that requires a computationally intensive protocol between the receiver and the third party. We now formally define a token-controlled public key encryption scheme as follows.

**Definition 1 (TCPKE).** A Token-Controlled Public Key Encryption scheme denoted by "TCPKE" consists of the following algorithms.

- A Randomized Key Generation Algorithm $\mathsf{GK}^{\mathsf{TCPKE}}(k)$: Taking a security parameter $k \in \mathbb{N}$ as input, this algorithm returns a private and public key pair $(sk, pk)$. Note that $pk$ includes the security parameter, descriptions of a finite plaintext space $\mathcal{P}$, a finite token space $\mathcal{T}$, and a ciphertext space $\mathcal{C}$.
- A Randomized Token Generation Algorithm $\mathsf{GT}^{\mathsf{TCPKE}}(k)$: Taking a security parameter $k \in \mathbb{N}$ as input, this algorithm chooses a token $tk \in \mathcal{T}$ at random and returns it.
- A Randomized Encryption Algorithm $\mathsf{E}^{\mathsf{TCPKE}}(pk, tk, M)$: Taking $pk$, $tk$, and $M \in \mathcal{P}$ as input, this algorithm returns a ciphertext $C \in \mathcal{C}$ which is an encryption of $M$.
- A Decryption Algorithm $\mathsf{D}^{\mathsf{TCPKE}}(sk, tk, C)$: Taking $sk$, $tk$, and $C$ as input, this algorithm returns a decryption $D$, which is either a plaintext $M \in \mathcal{P}$ or a special symbol "*Reject*".

*Security against Outside Attackers: IND-TCPKE-T1CCA/T2CCA* In terms of security of TCPKE, we need to consider two types of attackers, "*outside*" and "*inside*" attackers. This categorization is based on whether the attacker possesses the receiver's private key or not.

[*Type-1 Outside Attacker.*] We can further categorize the outside attackers into "type-1 outside attackers" and "type-2 outside attackers". Holding the public key of the receiver, the type-1 outside attacker's goal is to break the confidentiality of a TCPKE ciphertext created under a fixed token called a "target token" which is not given to the attacker. We assume that the attack conducted by this attacker is very active. First, the attacker has access to a "*token-embedded*" encryption oracle, which, upon receiving a plaintext message, returns a corresponding ciphertext created under the target token and the public key of the receiver. Also, this attacker has access to a decryption oracle, which, upon receiving a token-ciphertext pair of the attacker's choice, returns a corresponding decryption. Note that this models a situation where the attacker can record all the previous tokens and ciphertexts communicated among the sender, the receiver, and the third party to attack current ciphertexts.

Below, we formally define a security notion for TCPKE against the type-1 outside attacker, which we call "IND-TCPKE-T1CCA".

**Definition 2 (IND-TCPKE-T1CCA).** Let $\mathsf{A}$ be an attacker whose running time is bounded by $t$ which is polynomial in a security parameter $k$. We now consider the following game:

**Phase 1**: The key generation algorithm $\mathsf{GK}^{\mathsf{TCPKE}}(k)$ and the token generation algorithm $\mathsf{GT}^{\mathsf{TCPKE}}(k)$ are run. A private and public key pair $(sk, pk)$ and a target token $tk^*$ are then generated. $pk$ is given to $\mathsf{A}$ while $tk^*$ and $sk$ are kept secret from $\mathsf{A}$.
**Phase 2**: $\mathsf{A}$ queries a number of plaintexts, each of which is denoted by $M$, to the token-embedded encryption oracle and obtains a corresponding ciphertext $C = \mathsf{E}^{\mathsf{TCPKE}}(pk, tk^*, M)$. $\mathsf{A}$ also queries a number of token-ciphertext pairs, each of which is denoted by $(tk, C)$, to the decryption oracle and obtains a corresponding decryption $D = \mathsf{D}^{\mathsf{TCPKE}}(sk, tk, C)$, which is either a plaintext or a "*Reject*" symbol.
**Phase 3**: $\mathsf{A}$ outputs two equal-length plaintexts $(M_0, M_1)$. $\beta \in \{0, 1\}$ is then chosen at random and a target ciphertext $C^* = \mathsf{E}^{\mathsf{TCPKE}}(pk, tk^*, M_\beta)$ is created and returned to $\mathsf{A}$.
**Phase 4**: $\mathsf{A}$ issues a number of encryption and decryption oracle queries as in Phase 2.
**Phase 5**: $\mathsf{A}$ outputs its guess $\tilde{\beta} \in \{0, 1\}$.

We define $\mathsf{A}$'s success by the probability $\mathbf{Succ}^{\mathrm{IND-TCPKE-T1CCA}}_{\mathsf{TCPKE},\mathsf{A}}(k) = 2 \cdot \Pr[\tilde{\beta} = \beta] - 1$. The $\mathsf{TCPKE}$ scheme is said to be IND-TCPKE-T1CCA secure if $\mathbf{Succ}^{\mathrm{IND-TCPKE-T1CCA}}_{\mathsf{TCPKE},\mathsf{A}}(k)$ is negligible in $k$.

Note in the above attack game that no restriction is imposed on the token and ciphertext pairs queried to the decryption oracle in Phase 4. As a result, it is possible that $(tk, C) = (tk^*, C^*)$. However, if this event happens, the attacker comes to know $\beta$ with probability 1 and hence the IND-TCPKE-T1CCA is broken. Nevertheless, this would show that the token must be chosen from an exponentially large space, which makes the probability that query $tk$ equals $tk^*$ is negligible. (Note that in the proof of security, this would be one of the "bad" events in the attack which contributes to the scheme's insecurity function, which we could bound, as we will see in the proof of Theorem 1).

[*Type-2 Outside Attacker.*] The type-2 outside attacker still does not have the receiver's private key but does *have the token* that the sender used to encrypt messages. Namely, the type-2 outside attacker is the "semi-trusted" third party that was appointed by the sender to release the token. However, we assume that this party can behave maliciously to break the confidentiality of the $\mathsf{TCPKE}$ ciphertexts. Like the type-1 outside attackers, the party can query token-ciphertext pairs of its choice to the decryption oracle of $\mathsf{TCPKE}$. An attack game for a security notion associated with the type-2 attacker, which we call, "IND-TCPKE-T2CCA", is almost identical to that of IND-TCPKE-T1CCA except that the type-2 attacker cannot query the target token-ciphertext pair $(tk^*, C^*)$ to the decryption oracle in Phase 4. (Since the type-2 attacker knows the target token, it is unreasonable to allow it).

*Security against Inside Attacker: IND-TCPKE-ISCPA.* As mentioned earlier, an "inside" attacker is assumed to possess the private key of the receiver but not the associated token. In other words, the inside attacker is the receiver itself. Having access to the token-embedded encryption oracle, the goal of the inside attacker is to defeat the confidentiality of a ciphertext created under the token. The security against this attack implies that the receiver cannot get any information about the plaintext message without getting the associated token.

We formalize the security notion for $\mathsf{TCPKE}$ against the inside attacker, which we call "IND-TCPKE-ISCPA". ("ISCPA" is read as "inside chosen plaintext attack").

**Definition 3 (IND-TCPKE-ISCPA).** Let $\mathsf{A}$ be an attacker whose running time is bounded by $t$ which is polynomial in a security parameter $k$. We now consider the following game:

**Phase 1**: The key generation algorithm $\mathsf{GK}^{\mathsf{TCPKE}}(k)$ is run and a private and public key pair $(sk, pk)$ is generated. Then, the token generation algorithm $\mathsf{GT}^{\mathsf{TCPKE}}(k)$ is run and a target token $tk^*$ is generated. The *private/public* key pair $(sk, pk)$ is given to $\mathsf{A}$ while $tk^*$ is kept secret from $\mathsf{A}$.
**Phase 2**: $\mathsf{A}$ queries a number of plaintexts, each of which is denoted by $M$, to the token-embedded encryption oracle and obtains a corresponding ciphertext $C = \mathsf{E}^{\mathsf{TCPKE}}(pk, tk^*, M)$.
**Phase 3**: $\mathsf{A}$ outputs two equal-length plaintexts $(M_0, M_1)$. $\beta \in \{0, 1\}$ is then chosen at random and a target ciphertext $C^* = \mathsf{E}^{\mathsf{TCPKE}}(pk, tk^*, M_\beta)$ is created and returned to $\mathsf{A}$.
**Phase 4**: $\mathsf{A}$ queries a number of plaintexts to the token-embedded encryption oracle as in Phase 2.
**Phase 5**: $\mathsf{A}$ outputs his guess $\tilde{\beta} \in \{0, 1\}$.

We define $\mathsf{A}$'s success by the probability $\mathbf{Succ}^{\mathrm{IND-TCPKE-ISCPA}}_{\mathsf{TPKE,A}}(k) = 2\Pr[\tilde{\beta} = \beta] - 1$. The $\mathsf{TCPKE}$ scheme is said to be IND-TCPKE-ISCPA secure if $\mathbf{Succ}^{\mathrm{IND-TCPKE-ISCPA}}_{\mathsf{TPKE,A}}(k)$ is negligible in $k$.

*Relations among IND-TCPKE-T1CCA/T2CCA and IND-CCA.* Once a token is revealed, the scheme $\mathsf{TCPKE}$ can be treated as a normal public key encryption scheme. We call this scheme "$\mathsf{PKE}_{\mathsf{TCPKE}}$", which can be defined as follows.

**Definition 4 ($\mathsf{PKE}_{\mathsf{TCPKE}}$).** A (normal) public key encryption scheme $\mathsf{PKE}_{\mathsf{TCPKE}}$ derived from $\mathsf{TCPKE}$ consists of the following algorithms.

- A Randomized Key Generation Algorithm $\mathsf{GK}^{\mathsf{PKE}}(k)$: This algorithm first computes $(sk, pk) = \mathsf{GK}^{\mathsf{TCPKE}}(k)$ and $tk = \mathsf{GT}^{\mathsf{TCPKE}}(k, pk)$, where $k \in \mathbf{N}$ is a security parameter. It then returns a private key $\overline{sk} = sk$ and a public key $\overline{pk} = pk||tk$.
- A Randomized Encryption Algorithm $\mathsf{E}^{\mathsf{PKE}}(\overline{pk}, M)$: This algorithm computes $C = \mathsf{E}^{\mathsf{TCPKE}}(pk, M)$ and returns a ciphertext $\overline{C} = C||tk$.
- A Decryption Algorithm $\mathsf{D}^{\mathsf{PKE}}(\overline{sk}, \overline{C})$: This algorithm first parses $\overline{C}$ as $C||tk$ and computes $D = \mathsf{D}^{\mathsf{TCPKE}}(sk, tk, C)$. It then returns $D$. (Note that $D$ is either a plaintext $M \in \mathcal{P}$ or a special symbol "*Reject*").

Now we present a reduction result regarding the relationship between IND-TCPKE-T1CCA of $\mathsf{TCPKE}$ and IND-CCA of $\mathsf{PKE}_{\mathsf{TCPKE}}$. (Note that the formal definition of IND-CCA of public key encryption in general can be found in usual cryptographic literature including [1]). The following theorem implies that the scheme $\mathsf{PKE}_{\mathsf{TCPKE}}$ is secure in the IND-CCA sense and the token is chosen from an exponentially large space, then $\mathsf{TCPKE}$ is IND-TCPKE-T1CCA secure.

**Theorem 1.** *Suppose that a token for the* $\mathsf{TCPKE}$ *scheme is uniformly chosen at random from a space* $\mathcal{T}$ *such that* $|\mathcal{T}| > 2^{l_T(k)}$ *where* $l_T : \mathbb{N} \rightarrow \mathbb{N}$ *denotes linear function determining the length of a token. Assume that an attacker* $\mathsf{A}$ *making* $q_{TE}$ *queries to the token-embedded encryption oracle and* $q_D$ *queries to the decryption oracle defeats the IND-TCPKE-T1CCA of the* $\mathsf{TCPKE}$ *scheme within time* $t$*. Then for the attacker* $\mathsf{A}$*, there exists an IND-CCA attacker* $\mathsf{B}$ *for the* $\mathsf{PKE}_{\mathsf{TCPKE}}$ *scheme, such that for any* $k \in \mathbb{N}$*,*

$$\mathbf{Succ}^{\mathrm{IND-TCPKE-T1CCA}}_{\mathsf{TCPKE},\mathsf{A}}(k) \leq \mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathsf{PKE}_{\mathsf{TCPKE}},\mathsf{B}}(k) + \frac{q_D}{2^{l_T(k)-1}}$$

*and* $t' = t + q_{TE}T_E$ *and* $q'_D = q_D$*, where* $t'$*,* $T_E$*, and* $q'_D$ *denote the running time of* $\mathsf{B}$*, the time required to encrypt a message using* $\mathsf{PKE}_{\mathsf{TCPKE}}$*, and the number of decryption oracle queries made by* $\mathsf{B}$ *respectively.*

The proof is given in Appendix A.

Regarding the relationship between IND-TCPKE-T2CCA and IND-CCA, we obtain a similar result. That is, if the scheme $\mathsf{PKE}_{\mathsf{TCPKE}}$ is secure in the IND-CCA sense then $\mathsf{TCPKE}$ is IND-TCPKE-T2CCA secure, which can be stated as follows.

**Theorem 2.** *Assume that an attacker* $\mathsf{A}$ *making* $q_D$ *queries to the decryption oracle defeats the IND-TCPKE-T2CCA of the* $\mathsf{TCPKE}$ *scheme within time* $t$*. Then for the attacker* $\mathsf{A}$*, there exists an IND-CCA attacker* $\mathsf{B}$ *for the* $\mathsf{PKE}_{\mathsf{TCPKE}}$ *scheme, such that for any* $k \in \mathbb{N}$*,*

$$\mathbf{Succ}^{\mathrm{IND-TCPKE-T2CCA}}_{\mathsf{TCPKE},\mathsf{A}}(k) \leq \mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathsf{PKE}_{\mathsf{TCPKE}},\mathsf{B}}(k)$$

*and* $t' = t$ *and* $q'_D = q_D$*, where* $t'$ *and* $q'_D$ *denote the running time of* $\mathsf{B}$ *and the number of decryption oracle queries made by* $\mathsf{B}$ *respectively.*

Intuitively, what the type-2 outside attacker can do to break the IND-TCPKE-T2CCA of the $\mathsf{TCPKE}$ scheme is equivalent to what an IND-CCA attacker for the $\mathsf{PKE}_{\mathsf{TCPKE}}$ scheme can do in that the type-2 outside attacker "sees" the target token used to create a target ciphertext, which are all available to the IND-CCA attacker for the $\mathsf{PKE}_{\mathsf{TCPKE}}$ scheme.

## 4 Realization of Token-Controlled Public Key Encryption

*Design Principles.* We note that our $\mathsf{TCPKE}$ schemes will have a property called "public checkability." In publicly checkable encryption [1, 13], the ciphertext validity check for chosen ciphertext security can be performed by anyone, even

who does not possess the decryption key. Hence, publicly checkable encryption schemes are useful in situations where a large number of incoming ciphertexts need to be checked and screened without having them to be decrypted as observed in [1]. Although this property is not a formal requirement of $\mathsf{TCPKE}$ as defined in Definition 1, it will also be useful in the situation of the token-controlled public key encryption, especially when the receivers do not have tokens to decrypt incoming ciphertexts yet, but want the validity of them to be checked. More precisely, the $\mathsf{PKE}_{\mathsf{TCPKE}}$ schemes derived from our $\mathsf{TCPKE}$ schemes will be publicly checkable without requiring the "token part" of the $\mathsf{PKE}_{\mathsf{TCPKE}}$ ciphertext, i.e. "$tk$" in $\overline{C} = C||tk$ (as described in Definition 4), to be involved in the validity checking for ensuring IND-CCA. (In other words, if $C$ has checked before the token is released, the $\overline{C} = C||tk$ does not require to be checked again in the future).

However, care must be taken when constructing such schemes. Consider the following $\mathsf{TCPKE}$ scheme, which is very similar to our second scheme, but turns out to be insecure in the IND-TCPKE-T2CCA sense: Suppose that a plaintext $M$ is encrypted to yield a ciphertext $C$ such that $C = (u, v, h, s) = (g^r, M \oplus H_1(u, y^r) \oplus H_2(\tau), H_3(v, g^z), z - hr)$, where $g$ is a generator of a group $\mathcal{G}$ of a prime order $q$; $r \in \mathbb{Z}_q^*$ and $z \in \mathbb{Z}_q^*$ are chosen at random; $\tau$ is a token chosen at random from an appropriate space. Suppose that $\tau$ is given to the (malicious) third party. Suppose also that $C$ is a target ciphertext and $\tau$ is a target token. By the rule of the attack game of IND-TCPKE-T2CCA, the party cannot issue $(\tau, C)$ as a decryption oracle query. But he can replace $\tau$ with $\tau'$ and queries $(\tau', C)$ to the decryption oracle. Since the first part $(u, v, h, s)$ is still valid, $C$ will pass the validity test which checks whether $h = H_3(v, g^s u^h)$, and hence the oracle will return $M' = M \oplus H_2(\tau) \oplus H_2(\tau')$. However, the attacker can compute $H_2(\tau)$ and $H_2(\tau')$ by himself, so the $M$ can easily be recovered from $M'$!

Another construction of a $\mathsf{TCPKE}$ scheme can be considered as follows. Let $(pk, sk)$ be keys for an IND-CCA secure public-key encryption scheme with encryption algorithm $E$, and let $E'$ be an encryption algorithm for an IND-CCA secure symmetric-key scheme. Then to encrypt a message $m$, choose a random key $k$ as a token and send the ciphertext $E_{pk}(E'_k(m))$. However, to realize the IND-CCA secure symmetric-key scheme, one may need to employe the technique presented in [7], which makes a scheme somewhat complicated.

We now present our two $\mathsf{TCPKE}$ schemes.

*Our First Scheme Based on Bilinear ElGamal Encryption + Short Signature.* Our first scheme $\mathsf{TCPKE}_{\mathsf{BS}}$ is based on a combination of the bilinear pairing version of the ElGamal encryption scheme [4] and the Short Signature scheme [5]. The term "bilinear pairing" used in this paper refers to the admissible bilinear map $\hat{e} : \mathcal{G} \to \mathcal{F}$ [4] where $\mathcal{G}$ and $\mathcal{F}$ are groups of order prime $q$, which has the following property: 1) Bilinear: $\hat{e}(aR_1, bR_2) = \hat{e}(R_1, R_2)^{ab}$, where $R_1, R_2 \in \mathcal{G}$ and $a, b \in \mathbb{Z}_q^*$; 2) Non-degenerate: $\hat{e}$ does not send all pairs of points in $\mathcal{G} \times \mathcal{G}$ to the identity in $\mathcal{F}$. (Hence, if $R$ is a generator of $\mathcal{G}$ then $\hat{e}(R, R)$ is a generator of $\mathcal{F}$).; 3)Computable: For all $R_1, R_2 \in \mathcal{G}$, the map $\hat{e}(R_1, R_2)$ is efficiently computable.

We use the Short Signature scheme proposed in [5] to convert the bilinear pairing version of the ElGamal encryption scheme to a publicly checkable IND-CCA secure scheme.

[*Description.*] The single-receiver token controlled public key encryption scheme $\mathsf{TCPKE}_{\mathsf{BS}}$ consists of the following algorithms:

- Randomized Key Generation Algorithm $\mathsf{GK}_{\mathsf{BS}}^{\mathsf{TCPKE}}(k)$: Choose two groups $\mathcal{G} = \langle P \rangle$ and $\mathcal{F}$ of the same prime order $q \geq 2^{l_q(k)}$ and chooses a generator $P$ of $\mathcal{G}$. Then, construct a bilinear pairing $\hat{e} : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{F}$ and choose hash functions $\mathsf{H}_1 : \mathcal{F} \times \mathcal{G} \rightarrow \{0,1\}^{l_M(k)}$ and $\mathsf{H}_2 : \mathcal{G}^* \times \{0,1\}^{l_M(k)} \rightarrow \mathcal{G}^*$. (Note that $l_q : \mathbb{N} \rightarrow \mathbb{N}$ and $l_M : \mathbb{N} \rightarrow \mathbb{N}$ denote linear functions determining the length of $q$ and message $M$ respectively). Then, choose $x \in \mathbb{Z}_q^*$ uniformly at random and compute $Y = xP$. Return a public key $pk = (\mathrm{k}, \mathcal{G}, \hat{e}, q, P, Y, H_1, H_2, l_q, l_M)$ and a private key $sk = (\mathcal{G}, \hat{e}, q, P, x, H_1, H_2, k, l_q, l_M, d_{\mathcal{T}})$, where $d_{\mathcal{T}}$ denotes a description of a token space $\mathcal{T}$.
- Randomized Token Generation Algorithm $\mathsf{GT}_{\mathsf{BS}}^{\mathsf{TCPKE}}(k)$: Choose $T \in \mathcal{G}^*(= \mathcal{T})$ uniformly at random and return a token $tk = T$.
- Randomized Encryption Algorithm $\mathsf{E}_{\mathsf{BS}}^{\mathsf{TCPKE}}(pk, tk, M)$: Choose $r \in \mathbb{Z}_q^*$ uniformly at random and subsequently compute $U = rP$, $\kappa = \hat{e}(T, Y)^r$, $K = H_1(\kappa, T)$, $V = K \oplus M$, $L = H_2(U, V)$, and $W = rL$. Return a ciphertext $C = (U, V, W)$.
- Decryption Algorithm $\mathsf{D}_{\mathsf{BS}}^{\mathsf{TCPKE}}(sk, tk, C)$: If $\hat{e}(P, W) = \hat{e}(U, H_2(U, V))$ return $V \oplus H_1(\hat{e}(T, U)^x, T)$, otherwise, return "*Reject*".

[*Security Analysis.*] *Security against Outside/Limited Inside Attackers.* As shown in Theorems 1 and 2, and IND-CCA of a public key encryption scheme $\mathsf{PKE}_{\mathsf{TCPKE}_{\mathsf{BS}}}$ derived from $\mathsf{TCPKE}_{\mathsf{BS}}$ following Definition 4 implies the security of $\mathsf{TCPKE}_{\mathsf{BS}}$ against outside/limited inside attackers in the single receiver setting. Hence, it is important to prove that $\mathsf{PKE}_{\mathsf{TCPKE}_{\mathsf{BS}}}$ is IND-CCA secure and token-reusable.

We now prove that the hardness of the Bilinear Diffie-Hellman (BDH) problem [4] (given $aP, bP, cP \in \mathcal{G}$, computes $\hat{e}(P, P)^{abc} \in \mathcal{F}$) is sufficient for the $\mathsf{PKE}_{\mathsf{TCPKE}_{\mathsf{BS}}}$ scheme to be IND-CCA secure in the random oracle model [3].

**Theorem 3.** *Assume that an attacker* $\mathsf{A}$ *making* $q_{H_1}$ *and* $q_{H_2}$ *queries to the random oracles* $H_1$ *and* $H_2$*, and* $q_D$ *oracle queries to the decryption oracle defeats the IND-CCA of the* $\mathsf{PKE}_{\mathsf{TCPKE}_{\mathsf{BS}}}$ *scheme within time* $t$*. Then, for the attacker* $\mathsf{A}$*, there exists an attacker* $\mathsf{B}$ *that breaks the BDH problem within time* $t'$ *such that for any* $k \in \mathbb{N}$*,*

$$\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathsf{PKE}_{\mathsf{TCPKE}_{\mathsf{BS}}},\mathsf{A}}(k) \leq 2\mathbf{Succ}^{\mathrm{BDH}}_{\mathcal{G},\mathsf{B}}(k) + \frac{q_D}{2^{k-1}}$$

*and* $t' = t + q_D(3T_{BP} + O(1))$*, where* $T_{BP}$ *denotes the time for computing the bilinear pairing.*

In terms of the security of $\mathsf{TCPKE}_{\mathsf{BS}}$ against inside attackers in the single and multiple receiver setting, we investigate IND-TCPKE-ISCPA and IND-MRTCPKE-ISCPA of $\mathsf{TCPKE}_{\mathsf{BS}}$.

We first review the "modified Generalized Bilinear Inversion (GBI)" problem presented in [2]. The mGBI problem refers to the computational problem in which an attacker, given a generator $P$ of $\mathcal{G}$ and $h \in \mathcal{F}$, is to find a pair $S \in \mathcal{G}$ such that $\hat{e}(S, P) = h$.

We then prove that the hardness of mGBI problem implies IND-TCPKE-ISCPA of $\mathsf{TCPKE}_{\mathsf{BS}}$.

**Theorem 4.** *Assume that an attacker* $\mathsf{A}$ *making* $q_{H_1}$ *and* $q_{H_2}$ *queries to the random oracles* $H_1$ *and* $H_2$ *defeats the IND-TCPKE-ISCPA of the* $\mathsf{TCPKE}_{\mathsf{BS}}$ *scheme within time* $t$*. Then, for the attacker* $\mathsf{A}$*, there exists an attacker* $\mathsf{B}'$ *that breaks the mGBI problem within time* $t'$ *such that for any* $k \in \mathbb{N}$*,*

$$\frac{1}{2}\mathbf{Succ}^{\mathrm{IND-TCPKE-ISCPA}}_{\mathsf{TCPKE}_{\mathsf{BS}},\mathsf{A}}(k) \leq \mathbf{Succ}^{\mathrm{mGBI}}_{\mathcal{G},\mathsf{B}}(k)$$

*and* $t' = t + O(k^3)$*.*

*Our Second Scheme Based on ElGamal Encryption + Schnorr Signature.* Our second scheme $\mathsf{TCPKE}_{\mathsf{ES}}$ modifies the combination of the normal ElGamal encryption scheme constructed using a subgroup of the multiplicative group $\mathbb{Z}_p^*$ with prime $p$ and the Schnorr [11] signature scheme [1, 12, 13].
[*Description*] The single-receiver token controlled public key encryption scheme $\mathsf{TCPKE}_{\mathsf{ES}}$ consists of the following algorithms:

- Randomized Key Generation Algorithm $\mathsf{GK}^{\mathsf{TCPKE}}_{\mathsf{ES}}(k)$: Choose a finite cyclic subgroup $\mathcal{G} = \langle g \rangle$ of a multiplicative group $\mathbb{Z}_p^*$ with prime $p$ such that $Ord_{\mathcal{G}}(g) = q$, where $q$ is a prime such that $|q| > 2^{l_q(k)}$. Choose hash functions $H_1 : \mathcal{G} \times \mathcal{G} \times \{0,1\}^* \to \{0,1\}^{l_M(k)}$ and $H_2 : \{0,1\}^{l_M(k)} \times \mathcal{G} \to \mathbb{Z}_q^*$. (Note that $l_q : \mathbb{N} \to \mathbb{N}$ and $l_M : \mathbb{N} \to \mathbb{N}$ denote linear functions determining the length of $q$ and message $M$ respectively). Choose $x \in \mathbb{Z}_q^*$ uniformly at random and compute $y = g^x$. Then, output a private and public key pair $(sk, pk)$ such that $sk = (k, \mathcal{G}, g, p, q, x, H_1, H_2, l_q, l_M)$ and $pk = (k, \mathcal{G}, g, p, q, y, H_1, H_2, l_q, l_M, d_{\mathcal{T}})$, where $d_{\mathcal{T}}$ denotes a description of a token space $\mathcal{T}$.
- Randomized Token Generation Algorithm $\mathsf{GT}^{\mathsf{TCPKE}}_{\mathsf{ES}}(k)$: Choose $\tau \in \{0,1\}^{l_T(k)}$ uniformly at random and return a token $tk = \tau$. (Note that $l_T : \mathbb{N} \to \mathbb{N}$ denotes a linear function determining the length of a token).
- Randomized Encryption Algorithm $\mathsf{E}^{\mathsf{TCPKE}}_{\mathsf{ES}}(pk, tk, M)$: Choose $r \in \mathbb{Z}_q^*$ at random. Compute $u = g^r$, $\kappa = y^r$, $K = H_1(u, \kappa, \tau)$, and $v = K \oplus M$. Choose $z \in \mathbb{Z}_q^*$ at random. Compute $w = g^z$, $h = H_2(v, w)$, and $s = z - hr$. Return a ciphertext $C = (u, v, h, s)$.
- Decryption Algorithm $\mathsf{D}^{\mathsf{TCPKE}}_{\mathsf{ES}}(sk, tk, C)$: If $h = H_2(v, g^s u^h)$, compute $\kappa = y^x$ and $K = H_1(u, \kappa, \tau)$, and return $M = K \oplus v$, otherwise, return "*Reject*".

[*Security Analysis.*] In the same way we analyzed the security of $\mathsf{TCPKE}_{\mathsf{BS}}$ against outside/limited inside attackers in the single receiver setting, we show that the public key encryption scheme $\mathsf{PKE}_{\mathsf{TCPKE}_{\mathsf{ES}}}$ derived from $\mathsf{TCPKE}_{\mathsf{ES}}$ following Definition 4 is IND-CCA secure.

We now prove that the intractability of the Gap Diffie-Hellman (GDH) problem [9], which is to solve the Computational Diffie-Hellman (CDH) problem with the help of Decisional Diffie-Hellman (DDH) oracle, is sufficient for the $\mathsf{PKE_{ES}}$ scheme to be IND-CCA secure in the random oracle model.

**Theorem 5.** *Assume that an attacker* $\mathsf{A}$ *making* $q_{H_1}$ *and* $q_{H_2}$ *queries to the random oracles* $H_1$ *and* $H_2$*, and* $q_D$ *oracle queries to the decryption oracle defeats the IND-CCA of the* $\mathsf{PKE_{TCPKE_{ES}}}$ *scheme within time* $t$*. Then, for the attacker* $\mathsf{A}$*, there exists an attacker* $\mathsf{B}$ *that breaks the GDH problem within time* $t'$ *such that for any* $k \in \mathbb{N}$*,*

$$\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathsf{PKE_{TCPKE_{ES}}},\mathsf{A}}(k) \leq 2\mathbf{Succ}^{\mathrm{GDH}}_{\mathcal{G},\mathsf{G}}(k) + \frac{q_D}{2^{k-1}}$$

*and* $t' = t + q_D(3T_{DDH} + O(1))$*, where* $T_{DDH}$ *denotes the running time of the DDH oracle.*

*Security against Inside Attackers.* IND-TCPKE-ISCPA and IND-TCPKE-ISCPA of the $\mathsf{TCPKE_{ES}}$ scheme depends solely on the assumption that the token is chosen uniformly at random from the space $\{0,1\}^{l_T(k)}$ and the hash function $H_1$ is a random oracle. The advantages of the attackers for those notions are abounded by $O(\frac{1}{2^{l_T(k)}})$, which will be negligible if $l_T(k)$ is large.

## References


1. M. Abe, *Combiniang Encryption and Proof of Knowledge in the Random Oracle Model*, Computer Journal 47(1), pp. 58–70, Oxford Uni. Press, 2004.
2. J. Baek and Y. Zheng, *Identity-based threshold signature scheme from the bilnear parings*, In IAS track of ITCC '04, pp. 124-128, IEEE Computer Society, 2004.
3. M. Bellare and P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, In ACM-CCCS '93, pp. 62–73, 1993.
4. D. Boneh and M. Franklin, *Identity-Based Encryption from he Weil Pairing*, In Crypto '01, LNCS 2139, pp. 213–229, Springer-Verlag, 2001.
5. D. Boneh, B. Lynn and H. Shacham, *Short Signatures from the Weil Pairing*, In Asiacrypt '01, LNCS 2248, pp. 566–582, Springer-Verlag, 2001.
6. G. Di Crescenzo, R. Ostrovsky, and S. Rajagopalan, *Conditional Oblivious Transfer and Timed-Release Encryption* , In Eurocrypt '99, LNCS 1592, pp. 74–89, Springer-Verlag, 1999.
7. A. Desai, *New Paradigms for Constructing Symmetric Encryption Schemes Secure against Chosen-Ciphertext Attack*, In Crypto '01, LNCS 1880, pp. 394–412, Springer-Verlag, 2000.
8. T. May, *Timed-Release Crypto*, Manuscript, 1993.
9. T. Okamoto and D. Pointcheval, *The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes*, In PKC '01, LNCS 1992, pp. 104–118, Springer-Verlag, 2001.
10. R. Rivest, A Shamir, and D. Wagner, *Time-Lock Puzzles and Timed-Release Crypto*, Technical Report, MIT/LCS/TR-684.
11. C. P. Schnorr, *Efficient Signature Generation for Smarts Cards*, Journal of Cryptology, Vol. 4, pp. 239–252, Springer-Verlag, 1991.

12. C. P. Schnorr and M. Jakobsson, *Security of Signed ElGamal Encryption* , In Asiacrypt '00, LNCS 1976, pp. 73–89, Springer-Verlag, 2000.
13. Y. Tsiounis and M. Yung: *On the Security of ElGamal-Based Encryption*, In PKC '98, LNCS 1431, pp. 117–134, Springer-Verlag, 1998.

## A Proof of Theorem 1

*Proof.* Suppose that $\mathsf{B}$ is given a private/public key pair $(\overline{sk}, \overline{pk})$ such that $\overline{sk} = sk$ and $\overline{pk} = pk||tk$, where $(sk, pk) = \mathsf{GK}^{\mathsf{TCPKE}}(k)$ and $tk = \mathsf{GT}^{\mathsf{TCPKE}}(k)$. $\mathsf{B}$ lets $tk^* = tk$. then gives $pk$ to $\mathsf{A}$ while keeps $sk$ and $tk^*$ secret. $\mathsf{B}$ then simulates the oracles that $\mathsf{A}$ has access to in the real attack as follows.

- Simulation of Token-Embedded Encryption Oracle: For each query $M$,
  - compute $\overline{C} = C||tk^* = \mathsf{E}^{\mathsf{PKE}}(\overline{pk}, M)$ and return $C$. (Note that $C = \mathsf{E}^{\mathsf{TCPKE}}(pk, tk, M)$).
- Simulation of Encryption Oracle: For two equal-length plaintexts $M_0$ and $M_1$ as a challenge,
  - forward $(M_0, M_1)$ to the encryption oracle of $\mathsf{PKE}_{\mathsf{TCPKE}}$ to obtain a (target) ciphertext $\overline{C^*} = C^*||tk^* = \mathsf{E}^{\mathsf{PKE}}(\overline{pk}, M_\beta)$ for random $\beta \in \{0, 1\}$ and return $C^*$. (Note that $C^* = \mathsf{E}(pk, tk^*, M_\beta)$).
- Simulation of Decryption Oracle: For each query $(C, tk)$,
  - if $(C, tk) = (C^*, tk^*)$, stop the simulation;
  - otherwise, forward $\overline{C} = C||tk$ to the decryption oracle of $\mathsf{PKE}_{\mathsf{TCPKE}}$ to obtain a decryption $D = \mathsf{D}^{\mathsf{PKE}}(\overline{sk}, \overline{C})$ and return $D$. (Note that $\mathsf{D}^{\mathsf{PKE}}(\overline{sk}, \overline{C}) = \mathsf{D}^{\mathsf{TCPKE}}(sk, tk, C)$).
- When $\mathsf{A}$ outputs a guess $\tilde{\beta} \in \{0, 1\}$ for the bit $\beta$, $\mathsf{B}$ returns $\tilde{\beta}$ as its guess.

*Analysis.* By the specifications presented above, the oracles that $\mathsf{A}$ has access to are perfectly simulated except for the case that $\mathsf{A}$ queries $(C^*, tk^*)$ to the decryption oracle. Let $\mathsf{TKBrk}$ be an event that $\mathsf{A}$ queries $(C, tk)$ such that $(C, tk) = (C^*, tk^*)$ in Phase 4. Since $\mathsf{B}$'s guess is exactly the same as $\mathsf{A}$'s guess $\tilde{\beta}$, if $\mathsf{TKBrk}$ does not happen, we have

$$\Pr[\tilde{\beta} = \beta | \neg \mathsf{TKBrk}] \leq \frac{1}{2} + \frac{1}{2}\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathsf{PKE}_{\mathsf{TCPKE}},\mathsf{B}}(k).$$

Thus, we obtain the following:

$$\begin{aligned}\frac{1}{2} + \frac{1}{2}\mathbf{Succ}^{\mathrm{IND-TCPKE-T1CCA}}_{\mathsf{TCPKE},\mathsf{A}}(k) = \Pr[\tilde{\beta} = \beta] &\leq \Pr[\tilde{\beta} = \beta | \neg \mathsf{TKBrk}] + \Pr[\mathsf{TKBrk}] \\ &\leq \frac{1}{2} + \frac{1}{2}\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathsf{PKE}_{\mathsf{TCPKE}},\mathsf{B}}(k) + \Pr[\mathsf{TKBrk}].\end{aligned}$$

Since we have assumed that the token $tk$ has been chosen uniformly at random from the space $\mathcal{T}$ such that $|\mathcal{T}| > 2^{l_T(k)}$ and the total $q_D$ decryption oracle queries are made by $\mathsf{A}$, $\Pr[\mathsf{TKBrk}] \leq q_D/2^{l_T(k)}$ and hence we get the bound in the theorem statement. Note that the running time of $\mathsf{B}$ is $t' = t + q_{TE}T_E$ where $T_E$ denotes the time required to encrypt a message using $\mathsf{PKE}_{\mathsf{TCPKE}}$. The number of decryption oracle queries made by $\mathsf{B}$ is $q'_D = q_D$. □

# A New Class of Codes for Fingerprinting Schemes

Marcel Fernandez, Miguel Soriano, and Josep Cotrina*

Department of Telematics Engineering. Universitat Politècnica de Catalunya, C/ Jordi Girona 1 i 3. Campus Nord, Mod C3, UPC. 08034 Barcelona. Spain
{marcelf, soriano, jcotrina}@entel.upc.es

**Abstract.** In this paper we discuss the problem of collusion secure fingerprinting. In the first part of our contribution we prove the existence of equidistant codes that can be used as fingerprinting codes. Then we show that by giving algebraic structure to the equidistant code, the tracing process can be accomplished by passing a modified version of the Viterbi algorithm through the trellis representing the code.

## 1 Introduction

In the multimedia content market, there is the need to protect both intellectual property and distribution rights against dishonest buyers. Encrypting the data only offers protection as long as the data remains encrypted, since once an authorized but fraudulent user decrypts it, nothing stops him from redistributing the data without having to worry about being caught.

The fingerprinting technique consists in making the copies of a digital object unique by embedding a different set of marks in each copy. Having unique copies of an object clearly rules out plain redistribution, but still a coalition of dishonest users can collude, compare their copies and by changing the marks where their copies differ, they are able to create a pirate copy that tries to disguise their identities. Thus, the fingerprinting problem consists in finding, for each copy of the object, the right set of marks that help to prevent collusion attacks. If the marks are the codewords of a collusion secure code, then it is guaranteed that at least one of the members of the coalition can be traced back with high probability.

The construction of collusion secure codes was first addressed in [1]. In that paper, Boneh and Shaw obtain a logarithmic length code by composing an inner binary code with an outer random code. In this paper we first discuss the use of equidistant codes as collusion secure fingerprinting codes against collusions of size 2. Secondly, we show how by giving structure to a code, the tracing of

* This work has been supported in part by the Spanish Research Council (CICYT) Projects TIC2002-00818 (DISQET) and TIC2003-01748 (RUBI) and by the IST-FP6 Project 506926 (UBISEC).

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 398–409, 2005.

dishonest users can be accomplished using a modified version of the Viterbi algorithm.

More precisely, for equidistant *parity check codes*, the elegant techniques from [8] can be used to construct the trellis of the code. The algorithm we present passes through the trellis and traces whenever possible the participants of a collusion.

The paper is organized as follows. In Section 2 we provide an overview of the coding theory concepts used throughout the paper. The use of equidistant codes as fingerprinting codes is discussed in Section 3. Section 4 deals with the tracing process and how it can be efficiently accomplished. In Section 5 we summarize our work.

## 2 Previous Results

### 2.1 Coding Theory Overview

Let $\mathbb{F}_q^n$ be the vector space over $\mathbb{F}_q$, then $C \subseteq \mathbb{F}_q^n$ is called a *code*. The field, $\mathbb{F}_q$ is called the *code alphabet*. A code $C$ is called a *linear code* if it forms a subspace of $\mathbb{F}_q^n$. If $q = 2$ then the code is called a binary code. The number of nonzero coordinates in $\mathbf{x}$ is called the *weight* of $\mathbf{x}$ and is commonly denoted by $w(\mathbf{x})$. The *Hamming distance* $\mathbf{d}(\mathbf{a}, \mathbf{b})$ between two words $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^n$ is the number of positions where $\mathbf{a}$ and $\mathbf{b}$ differ. If the dimension of the subspace is $k$, and its minimum Hamming distance is $d$, then we call $C$ an $[n,k,d]$-code. A code whose codewords are all the same distance apart is called an *equidistant code*.

A $(n-k) \times n$ matrix $\mathbf{H}$, is a *parity check matrix* for the code $C$, if $C$ is the set of codewords $\mathbf{c}$ for which $\mathbf{Hc} = \mathbf{0}$, where $\mathbf{0}$ is the all-zero $(n-k)$ tuple. Each row of the matrix is called a *parity check equation*. A code whose codewords satisfy all the parity check equations of a parity check matrix is called a *parity check code*.

For any two words $\mathbf{a}$, $\mathbf{b}$ in $\mathbb{F}_q^n$ we define the *set* of *descendants* $D(\mathbf{a}, \mathbf{b})$ as $D(\mathbf{a}, \mathbf{b}) := \{x \in \mathbb{F}_q^n : x_i \in \{a_i, b_i\}, 1 \leq i \leq n\}$. For a code $C$, the *descendant code* $C^*$ is defined as: $C^* := \bigcup_{\mathbf{a} \in C, \mathbf{b} \in C} D(\mathbf{a}, \mathbf{b})$.

If $\mathbf{c} \in C^*$ is a descendant of $\mathbf{a}$ and $\mathbf{b}$, then we call $\mathbf{a}$ and $\mathbf{b}$ *parents* of $\mathbf{c}$.

Note that the concepts of descendant and parents model the situation of a collusion attack, the descendant being the word in the pirate copy, and the parents being the participants in a collusion.

A code $C$ is $(2, 2)$-*separating* [6], if for any two disjoint subsets of codewords of size two, $\{\mathbf{a}, \mathbf{b}\}$ and $\{\mathbf{c}, \mathbf{d}\}$, where $\{\mathbf{a}, \mathbf{b}\} \cap \{\mathbf{c}, \mathbf{d}\} = \emptyset$, their respective sets of descendants are also disjoint, $D(\mathbf{a}, \mathbf{b}) \cap D(\mathbf{c}, \mathbf{d}) = \emptyset$.

The following results from [3] give a sufficient conditions for a linear code to be (2,2)-separating. Let $d_1$ and $m_1$ denote respectively the minimum and maximum weight of a non-zero codeword.

**Theorem 1 ([3]).** *If a code $C$ is a linear, binary $(2, 2)$-separating code, then* $d_1 < n - 2(k - 2)$.

**Theorem 2 ([3]).** *If a code satisfies $4d_1 > 2m_1 + n$, or if $4d_1 > 3m_1$ and it is linear, then it is (2,2)-separating.*

**Corollary 1 ([3]).** *All linear, equidistant codes are (2,2)-separating.*

The following result is a classic, and will be usefull throughout the paper. Let $C$ be an equidistant binary code, and let $\mathbf{z} \in C^*$. Then there are three possible configurations for the parents of $\mathbf{z}$.

1. In a star configuration, there is a single codeword, say $\mathbf{u}$, such that $\mathbf{d}(\mathbf{u}, \mathbf{z}) \leq (d/2) - 1$.
2. In a "degenerated" star configuration, there is a single pair of codewords, say $\{\mathbf{u}, \mathbf{v}\}$, such that $\mathbf{d}(\mathbf{u}, \mathbf{z}) = \mathbf{d}(\mathbf{v}, \mathbf{z}) = d/2$.
3. In a triangle configuration, there three possible pairs of codewords, say $\{\mathbf{u}, \mathbf{v}\}$, $\{\mathbf{u}, \mathbf{w}\}$ and $\{\mathbf{v}, \mathbf{w}\}$, such that $\mathbf{d}(\mathbf{u}, \mathbf{z}) = \mathbf{d}(\mathbf{v}, \mathbf{z}) = \mathbf{d}(\mathbf{w}, \mathbf{z}) = d/2$.

### 2.2 Trellis Representation of Block Codes

The contents of this section are based on [8].

For a binary linear block code, a *trellis* is defined as a graph in which the nodes represent states, and the edges represent transitions between these states. The nodes are grouped into sets $S_t$, indexed by a "time" parameter $t$, $0 \leq t \leq n$. The parameter $t$ indicates the *depth* of the node. The edges are unidirectional, with the direction of the edge going from the node at depth $t$, to the node at depth $t + 1$. Each edge is labeled using an element of $\mathbb{F}_2$.

In any depth $t$, the number of states in the set $S_t$ is at most $2^{(n-k)}$. The states at depth $t$ are denoted by $\mathbf{s}_t^i$, for certain values of $i$, $i \in \{0, 1, \ldots, 2^{(n-k)} - 1\}$. The states will be identified by binary $(n-k)$-tuples. In other words, if we order all the binary $(n-k)$-tuples from 0 to $2^{(n-k)} - 1$, then $\mathbf{s}_t^i$ corresponds to the $i$th tuple in the list. Using this order, for each set of nodes $S_t$, we can associate the set $I_t$ that consists of all the integers $i$, such that $\mathbf{s}_t^i \in S_t$. The set of edges incident to node $\mathbf{s}_t^i$ is denoted by $\mathcal{I}(\mathbf{s}_t^i)$.

In the trellis representation of a code $C$, each distinct path corresponds to a different codeword, in which the labels of the edges in the path are precisely the codeword symbols. The correspondence between paths and codewords is one to one, and it is readily seen from the construction process of the trellis, that we now present.

The construction algorithm of the trellis of a linear block code, uses the fact that every code word of $C$ must satisfy all the parity check equations imposed by the parity check matrix $\mathbf{H}$. In this case, the codewords are precisely the coefficients $c_1, c_2, \ldots, c_n$ of the linear combinations of the columns $\mathbf{h}_i$ of $\mathbf{H}$, that satisfy

$$c_1\mathbf{h}_1 + c_2\mathbf{h}_2 + \cdots + c_n\mathbf{h}_n = \mathbf{0}, \tag{1}$$

where $\mathbf{0}$ is the all zero $(n-k)$-tuple.

Intuitively, the algorithm first constructs a graph, in which all linear combinations of the columns of $\mathbf{H}$ are represented by a distinct path. Then removes all paths corresponding to the linear combinations that do not satisfy (1).

1. Initialization (depth $t = 0$):
   $S_0 = \{\mathbf{s}_0^0\}$, where $\mathbf{s}_0^0 = (0, \ldots, 0)$.
2. Iterate for each depth $t = 0, 1, \ldots, (n-1)$.
   (a) Construct $S_{t+1} = \{\mathbf{s}_{t+1}^0, \ldots, \mathbf{s}_{t+1}^{|I_{t+1}|}\}$, using
   $\mathbf{s}_{t+1}^j = \mathbf{s}_t^i + c_l \mathbf{h}_{t+1}$
   $\forall i \in I_t$ and $l = 0, 1$.
   (b) For every $i \in I_t$, according to 2a:
   – Draw a connecting edge between the node $\mathbf{s}_t^i$ and the 2 nodes it generates at depth $(t+1)$, according to 2a.
   – Label each edge $\theta_t^{i,j}$, with the value of $c_j \in \mathbb{F}_2$ that generated $\mathbf{s}_{t+1}^j$ from $\mathbf{s}_t^i$.
3. Remove all nodes that do not have a path to the all-zero state at depth $n$, and also remove all edges incident to these nodes.

According to the convention in 2b, for every edge $\theta_t^{i,j}$, we can define the function **label_of**$(\theta_t^{i,j})$ that, given a codeword $\mathbf{c} = (c_1, c_2, \ldots, c_n)$, returns the $c_j$ that generated $\mathbf{s}_{t+1}^j$ from $\mathbf{s}_t^i$

### 2.3 The Viterbi Algorithm

This section provides a brief overview of the Viterbi algorithm.

The Viterbi algorithm is a recursive optimal solution to the problem of estimating the state sequence of a discrete-time finite-state Markov process observed in memoryless noise [5]. In this scenario, given a sequence of observations, each path of the trellis has an associated "length". The Viterbi Algorithm (VA) identifies the state sequence corresponding to the minimum "length" path from time 0 to time $n$. The incremental "length" metric associated with moving from state $\mathbf{s}_t^i$ to state $\mathbf{s}_{t+1}^j$, is given by $l[\theta_t^{i,j}]$, where $\theta_t^{i,j}$ denotes the edge that goes from $\mathbf{s}_t^i$ to $\mathbf{s}_{t+1}^j$.

We consider time to be discrete. Using the notation of Section 2.2, the state $\mathbf{s}_t^i$ at time $t$ is one of a finite number $|I_t|$ of states, since $\mathbf{s}_t^i \in S_t$. In the trellises we deal with in this paper, there is only a single initial state $\mathbf{s}_0^0$, and a single final state $\mathbf{s}_n^0$. Since the process runs from time 0 to time $n$, the state sequence can be represented by a vector $\mathbf{s} = \langle \mathbf{s}_0^0, \ldots, \mathbf{s}_n^0 \rangle$.

Among all paths starting at node $\mathbf{s}_0^0$ and terminating at the node $\mathbf{s}_t^j$, we denote by $\psi_t^j$ the path segment with the shortest length. For a given node $\mathbf{s}_t^j$, the path $\psi_t^j$, is called the *survivor* path, and its length is denoted by $L[\psi_t^j]$. Note that, $L[\psi_t^j] := \min_{\theta_{t-1}^{i,j}} L[\psi_{t-1}^i] + l[\theta_{t-1}^{i,j}]$.

Due to the structure of the trellis, at any time $t = t_1$ there are at most $|S_{t_1}|$ survivors, one for each $\mathbf{s}_{t_1}^i$. The key observation is the following one [5]: the shortest complete path $\psi_n^0$ must begin with one of these survivors, if it did not, but passed through state $\mathbf{s}_{t_1}^l$ at time $t_1$, then we could replace its initial segment by $\psi_{t_1}^l$ to get a shorter path, which is a contradiction.

With the previous observation in mind, we see that for any time $(t-1)$, we only need to mantain $m$ survivors $\psi_{t-1}^m$ $(1 \leq m \leq |I_{t-1}|$, one survivor for each node), and their lengths $L[\psi_{t-1}^m]$. In order to move from time $t-1$ to time $t$:

- we extend the time $(t-1)$ survivors, one time unit along their edges in the trellis, this is denoted by $\psi_t^j = (\psi_{t-1}^i || \theta_{t-1}^{i,j})$.
- compute the new length $L[\psi_t^i, \theta_t^{i,j}]$, of the new extended paths, and for each node (state) we select as the time $t$ survivor the extended path with the shortest length.

The algorithm proceeds by extending paths and selecting survivors until time $n$ is reached, where there is only one survivor left.

**Viterbi Algorithm.**
*Variables:*
$t =$ time index; $\psi_t^j, \forall j \in I_t$ Survivor terminating at $\mathbf{s}_t^j$; $L[\psi_t^j], \forall j \in I_t$ Survivor length; $L[\psi_t^j, \theta_{t-1}^{i,j}]$ Length of the path $(\psi_{t-1}^i || \theta_{t-1}^{i,j})$.
*Initialization:*

$t = 0$;
$\psi_0^0 = \mathbf{s}_0^0$; $\quad \psi_t^j =$ arbitrary, $\quad t \neq 0, \forall j \in I_t$;
$L[\psi_0^0] = 0$; $\quad L[\psi_t^j] = \infty$, $\quad t \neq 0, \forall j \in I_t$.

*Recursion:* $(1 \leq t \leq n)$
for every $\mathbf{s}_t^j \in S_t$ do
for every $\mathbf{s}_{t-1}^i$, such that $\theta_{t-1}^{i,j}$ is defined, do
Compute $L[\psi_t^j, \theta_{t-1}^{i,j}] = L[\psi_{t-1}^i] + l[\theta_{t-1}^{i,j}]$
end_for
Find $L[\psi_t^j] = \min_{\theta_{t-1}^{i,j}} L[\psi_t^j, \theta_{t-1}^{i,j}]$
Store the tuple $(\psi_t^j, L[\psi_t^j])$
end_for
*Termination:*
At time $t = n$ the shortest complete path is stored as the survivor $\psi_n^0$.

## 3 Equidistant Codes as Fingerprinting Codes

In this section we discuss the use of equidistant codes as fingerprinting codes. Recall from Section 2 that given a descendant, there are three possible configurations for the parents of a descendant. Note that among these configurations, the only one that defeats the fingerprinting scheme is the triangle one, and therefore it should be difficult for the colluders to achieve it. Below, we show that the probability, that a collusion generates a descendant that "decodes" in a triangle configuration, can be made exponentially small by increasing the length (and reducing the rate) of the code.

Given two words $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$, we call the set $M(\mathbf{a}, \mathbf{b}) = \{i : a_i = b_i\}$ the set of *matching positions*. In the same fashion, we define the set of *unmatching positions* between $\mathbf{a}$ and $\mathbf{b}$, as $U(\mathbf{a}, \mathbf{b}) = \{i : a_i \neq b_i\}$

In the rest of the section we will suppose that $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ are codewords of $C$, and $\mathbf{z}$ is a descendant of $\mathbf{u}$ and $\mathbf{v}$.

We will need the following lemma about triangle configurations.

**Lemma 1.** *Let $C$ be an $[n, k, d]$ equidistant binary linear code. Let $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ be codewords of $C$, and let $\mathbf{z}$ be a descendant of $\mathbf{u}$ and $\mathbf{v}$. Then, in a triangle configuration we have that the set $U(\mathbf{w}, \mathbf{z})$ is a subset of $M(\mathbf{u}, \mathbf{v})$.*

*Proof.* Given two codewords $\mathbf{u}$ and $\mathbf{v}$, we have that in the $n - d$ positions in $M(\mathbf{u}, \mathbf{v})$, another codeword $\mathbf{w}$ is either equal to both $\mathbf{u}$ and $\mathbf{v}$, or different to both of $\mathbf{u}$ and $\mathbf{v}$. Since the code is equidistant this implies that in the $d$ positions in $U(\mathbf{u}, \mathbf{v})$, $\mathbf{w}$ is different to $\mathbf{u}$ and $\mathbf{v}$ in the same proportion, that of course is $d/2$. This in turn implies, that (to satisfy equidistancy) in the $n - d$ positions in $M(\mathbf{u}, \mathbf{v})$, $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ are different in exactly $d/2$ of these positions, that by construction of the descendant and the constraint of a triangle configuration, are precisely the positions in $U(\mathbf{w}, \mathbf{z})$. If follows that $U(\mathbf{w}, \mathbf{z}) \subset M(\mathbf{u}, \mathbf{v})$. ∎

The previous lemma is used in the next proposition to explicity state that in a triangle configuration, the positions in which $\mathbf{u}$ and $\mathbf{v}$ disagree, are precisely the ones in which the descendant $\mathbf{z}$ and $\mathbf{w}$ agree.

**Proposition 1.** *Let $C$ be an $[n, k, d]$ equidistant binary linear code. Let $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ be codewords of $C$, and let $\mathbf{z}$ be a descendant of $\mathbf{u}$ and $\mathbf{v}$. Then, if $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ form a triangle configuration we have that $U(\mathbf{u}, \mathbf{v}) \subset M(\mathbf{w}, \mathbf{z})$.*

*Proof.* From Lemma 1, we have that $|U(\mathbf{w}, \mathbf{z})| = d/2$ and $U(\mathbf{w}, \mathbf{z}) \subset M(\mathbf{u}, \mathbf{v})$, therefore, for a triangle configuration, in all the positions outside $M(\mathbf{u}, \mathbf{v})$, $\mathbf{w}$ and $\mathbf{z}$ must agree. ∎

**Lemma 2.** *There exists an $[n, k, d]$ equidistant binary linear code $C$, such that the inequality $d - k > 0$ is always satisfied. Moreover, we have that the number $d - k$ can be made arbitrary large by increasing $n$.*

*Proof.* The proof of the lemma uses the results from [2]. Note that a dual binary Hamming code of dimension $k$ has parameters $[2^k - 1, k, 2^{k-1}]$, therefore a linear increase in the dimension implies an exponential increase in the distance. Since every equidistant code is a sequence of dual binary Hamming codes [2], it is clear that such a code exists. ∎

Now we are in the position to show that in a fingerprinting scheme, the probability that a coalition of two codewords generates a descendant, that belongs to a triangle configuration can be made as small as desired. This was first shown in [4] for the particular case of dual binary Hamming codes. In the following proposition we generalize the results in [4] to any equidistant code.

**Proposition 2.** *Let $C$ be an $[n, k, d]$ equidistant binary linear code with minimum distance $d \geq 8$. Let $\mathbf{u}$ and $\mathbf{v}$ be codewords of $C$. Let $\mathbf{z}$ be a descendant of $\mathbf{u}$ and $\mathbf{v}$ generated as follows: in the positions where $\mathbf{u}$ and $\mathbf{v}$ differ, for the corresponding symbol of $\mathbf{z}$ choose either the symbol $\mathbf{u}$ or the symbol in $\mathbf{v}$ with equal probability. Then, the probability $p$ of $\mathbf{z}$ being at a distance exactly $d/2$ of another codeword $\mathbf{w}$ is $p \leq 2^{-(d-k)}$ that can be made arbitrarily small.*

*Proof.* Note that $p$ is the probability that $\mathbf{w}$ forms a triangle configuration with $\mathbf{u}$ and $\mathbf{v}$. From the definition of a descendant, we can see that when $\mathbf{u}$ and $\mathbf{v}$ generate the descendant they have to make a choice in the $d$ positions in $U(\mathbf{u},\mathbf{v})$. From Proposition 1 we see that for all of these positions the symbol chosen must be equal to the symbol in $\mathbf{w}$. The probability of this happening is $\left(\frac{1}{2}\right)^d$.

Since the total number of codewords in the code is $2^k$. The overall probability of generating a descendant that forms a triangle configuration is $p \leq \left(\frac{1}{2}\right)^d \cdot 2^k$ that using Lemma 2 can be made as small as desired. ∎

Unfortunately, there is a drawback in the previous proposition, since it implicitely considers that the strategy of the coalition $\mathbf{u}$ and $\mathbf{v}$ creating the descendant is a very poor one. Note that, for instance, using that strategy the descendant can even be one of the parents!!.

Now, we assume a more realistic strategy for the coalition and impose that the contribution to the descendent from $\mathbf{u}$ and $\mathbf{v}$ in the $d$ positions in $U(\mathbf{u},\mathbf{v})$, is exactly the same, i.e. $d/2$.

The following proposition simply states that for equidistant codes with distance $d \geq 8$, the number of descendants in $D(\mathbf{u},\mathbf{v})$ at distance exactly $d/2$ of $\mathbf{u}$ and $\mathbf{v}$ is greater than the number of codewords in $C$. Intuitively this means that there are many descendants that do not form a triangle configuration.

**Proposition 3.** *Let $C$ be an $[n,k,d]$ equidistant binary linear code with minimum distance $d \geq 8$, and let $\mathbf{u}$ and $\mathbf{v}$ be two codewords of $C$. Then, $\binom{d}{d/2} > 2^k$.*

*Proof.* We have that,

$$\begin{aligned}\binom{d}{d/2} &= \frac{d(d-1)(d-2)\cdots[d-(\frac{d}{2}-1)]}{\frac{d}{2}(\frac{d}{2}-1)(\frac{d}{2}-2)\cdots[\frac{d}{2}-(\frac{d}{2}-1)]} \\ &= 2^{d/2}\frac{(d-1)(d-3)\cdots[d-(\frac{d}{2}-1)]}{(d-\frac{d}{2})(d-\frac{d}{2}-2)\cdots[d-\frac{d}{2}-(\frac{d}{2}-2)]} \\ &\geq 2^{d/2}\cdot 2^{d/4}\end{aligned}$$

Note that this upper bound is by no means tight, but it will suffice for our purposes. Therefore we need to show that

$$2^{d/2}\cdot 2^{d/4} > 2^k \tag{2}$$

From Theorem 1 and Theorem 2, we have that for equidistant codes $d < n - 2(k-2)$ and that $d > n/2$: $2(k-2) < n-d < 2d-d \Rightarrow 2(k-2) < d \Rightarrow k < \frac{d}{2}+2$. It follows that (2) is satisfied as long as $d \geq 8$. ∎

Next theorem is the center technical result of this section.

**Theorem 3.** *Let $C$ be an $[n,k,d]$ equidistant binary linear code with minimum distance $d \geq 8$. Let $\mathbf{u}$ and $\mathbf{v}$ be codewords of $C$. Let $\mathbf{z}$ be a descendant of $\mathbf{u}$ and $\mathbf{v}$ generated as follows: in the $d$ positions where $\mathbf{u}$ and $\mathbf{v}$ differ, choose $d/2$ symbols*

*from* $\mathbf{u}$ *and* $d/2$ *symbols from* $\mathbf{v}$*. Then, the probability* $p$ *of* $\mathbf{z}$ *being at a distance exactly* $d/2$ *of another codeword* $\mathbf{w}$ *is* $p < \left(\frac{1}{2}\right)^{\frac{3d}{4}} \cdot 2^k$ *that can be made arbitrarily small.*

*Proof.* The reasoning of the proof follows the same pattern as the one in the proof of Proposition 2. The number of descendants that can be generated by choosing $d/2$ symbols from each parent is $\binom{d}{d/2}$. From Proposition 1 and since there are $2^k$ codewords, the probability of obtaining a descendant that forms a triangle configuration is $p = \dfrac{2^k}{\binom{d}{d/2}} < \left(\frac{1}{2}\right)^{\frac{3d}{4}} \cdot 2^k$ From Proposition 3 this probability is less than 1 if $d \geq 8$, and using Lemma 2 it can be made as small as desired.

∎

## 4 Tracing: Identifying the Guilty

We now tackle the problem of how to recover the guilty in case of a collusion attack. As stated before, this is the same as searching for the parents of a descendant $\mathbf{z}$.

In order to do the search efficiently, we will add structure to an equidistant code, and work with an equidistant parity check matrix $[n, k, d]$ code $C$. Such a code can be represented by a trellis using the results in Section 2.2.

It is well known that for binary linear equidistant codes $d$ is an even number. And since we don't know in advance if we have to deal with a star, degenerated star or triangle configuration, we have to design an algorithm that outputs all codewords of a (2,2)-separating code within distance $d/2$ of $\mathbf{z}$. Since the error correcting bound of the code is $\lfloor \frac{d-1}{2} \rfloor$ we have that in the cases, "degenerated" star and triangle, we need to correct one more than the error correcting bound of the code. As it is shown below, this can be done by modifying the Viterbi algorithm.

### 4.1 Tracing Algorithm

In [8] it is shown that maximum likelihood decoding of any $[n, k, d]$ block code can be accomplished by applying the VA to a trellis representing the code. However, the algorithm discussed in [8] falls into the category of unique decoding algorithms since it outputs a single codeword, and is therefore not fully adequate for our purposes. In this section we present a modified version of the Viterbi algorithm that when applied to a descendant, outputs a list that contains all codewords within distance $d/2$ of the descendant. If the list is of size 3, then there are three possible pairs of parents, whose intersection is disjoint. In a fingerprinting scheme, this basically means that the colluders cannot be traced. The algorithm we present falls into the category of list Viterbi decoding algorithms [7].

We first give an intuitive description of the algorithm.

Recall that, in order to search for the parents of a given descendant $\mathbf{z}$, we find, either the unique codeword at a distance less or equal than $\frac{d}{2}-1$ of $\mathbf{z}$, or the two or three codewords at a distance $\frac{d}{2}$ of $\mathbf{z}$. Let $\mathbf{z}=(z_1,z_2,\ldots,z_n)$ be a descendant. Let $\boldsymbol{\theta}_c=\{\theta_0^{0,l},\ldots,\theta_{t-1}^{i,j},\ldots\theta_{n-1}^{k,0}\}$ the sequence of edges in the path associated with codeword $\mathbf{c}=(c_1,\ldots,c_t,\ldots,c_n)$. As defined in Section 2.2, we have that $\mathbf{label\_of}(\theta_{t-1}^{i,j})=c_t$. Each distinct path of the trellis corresponds to a distinct codeword, and since we need to search for codewords within a given distance of $\mathbf{z}$, it seems natural to define the "*length*" of the edge $\theta_{t-1}^{i,j}$, $l[\theta_{t-1}^{i,j}]$, as $l[\theta_{t-1}^{i,j}]:=\mathbf{d}(z_t,c_t)=\mathbf{d}(z_t,\mathbf{label\_of}(\theta_{t-1}^{i,j}))$.

Since we expect the algorithm to return all codewords within distance $d/2$ of $\mathbf{z}$, we can have more than one "survivor" for each node. For node $\mathbf{s}_t^j$, we denote the $l$th "*survivor*" as $\psi_t^{j,l}$.

Using the above "length" definition for $l[\theta_{t-1}^{i,j}]$, the length of the path $\psi_t^{j,c}$ associated with codeword $\mathbf{c}$, as the Hamming distance between $\mathbf{z}$ and $\mathbf{c}$, both truncated in the first $t$ symbols is $L[\psi_t^{j,c}]:=\mathbf{d}(\mathbf{z},\mathbf{c})=\sum_{m=1}^{t}\mathbf{d}(z_m,\mathbf{label\_of}(\theta_{m-1}^{i,j}))$.

Then, whenever $L[\psi_t^{j,c}]>2^{r-2}$ we can remove the path $\psi_t^{j,c}$ from consideration. Note that, for a given node the different "survivors" do not necessarily need to have the same length. For each node (state) $\mathbf{s}_t^j$, in the trellis, we maintain a list $\Psi_t^j$ of tuples $(\psi_t^{j,k},L[\psi_t^{j,k}])$, $k\in\{1,\ldots,|\Psi_t^j|\}$, where $\psi_t^{j,k}$ is a path passing through $\mathbf{s}_t^j$ and $L[\psi_t^{j,k}]$ is its corresponding length.

**Tracing Viterbi Algorithm. (TVA)**

*Variables:* $t=$ time index; $\psi_t^{j,m}$, $\forall j\in I_t$ $m$th survivor terminating at $\mathbf{s}_t^j$; $L[\psi_t^{j,m}]$, $\forall j\in I_t$ $m$th survivor length; $L[\psi_t^{j,m},\theta_{t-1}^{i,j}]$ Length of the path $(\psi_{t-1}^{i,k}||\theta_{t-1}^{i,j})$; $\Psi_t^j$, $\forall j\in I_t$ List of "survivors" terminating at $\mathbf{s}_t^j$.

*Initialization:*

$t=0$;
$\psi_0^{0,1}=\mathbf{s}_0^0$; $L[\psi_0^{0,1}]=0$; $\Psi_0^0=\{(\psi_0^{0,1},L[\psi_0^{0,1}])\}$;
$\Psi_t^j=\{\emptyset\}$ $\forall t\neq 0$

*Recursion:* $(1\leq t\leq n)$
  for every $\mathbf{s}_t^j\in S_t$ do
    $m:=0$
    for every $\mathbf{s}_{t-1}^i$ such that $\theta_{t-1}^{i,j}$ is defined do
      for every $\psi_{t-1}^{i,k}\in\Psi_{t-i}^i$
        Compute $L[\psi_t^{j,m},\theta_{t-1}^{i,j}]=L[\psi_{t-1}^{i,k}]+l[\theta_{t-1}^{i,j}]$
        if $L[\psi_t^{j,m}]<=d/2$
          add $(\psi_t^{j,m},L[\psi_t^{j,m}])$ to $\Psi_t^j$
          $m:=m+1$

*Termination:*

The codewords associated with each path $\psi_n^{0,m}\in\Psi_n^0$ are all within distance $d/2$ of $\mathbf{z}$.

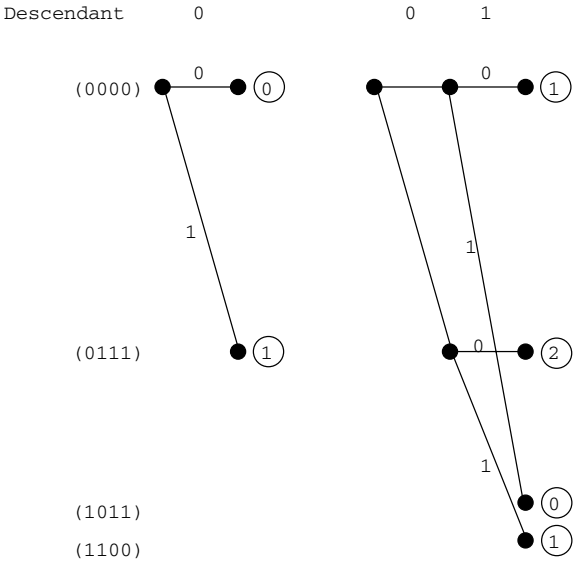



**Fig. 1.** TVA. Survivors at time $t = 1$ and $t = 2$

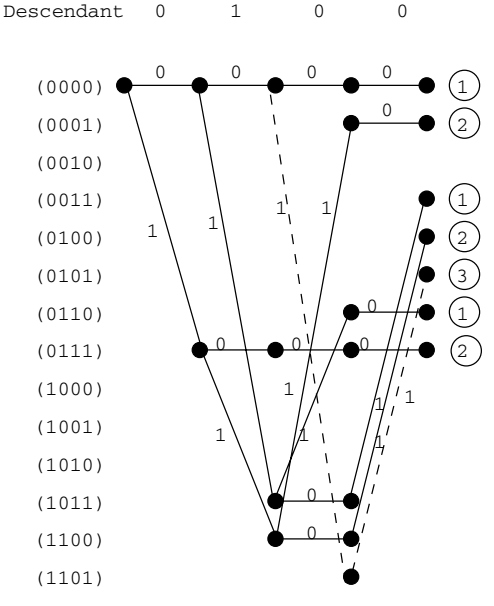



**Fig. 2.** TVA. Survivors at time $t = 4$

### 4.2 Example

If one needs to work with a binary equidistant parity check code, the dual binary Hamming $[2^r - 1, r, 2^{r-1}]$ code, $S_r$, probably comes as the most natural choice.

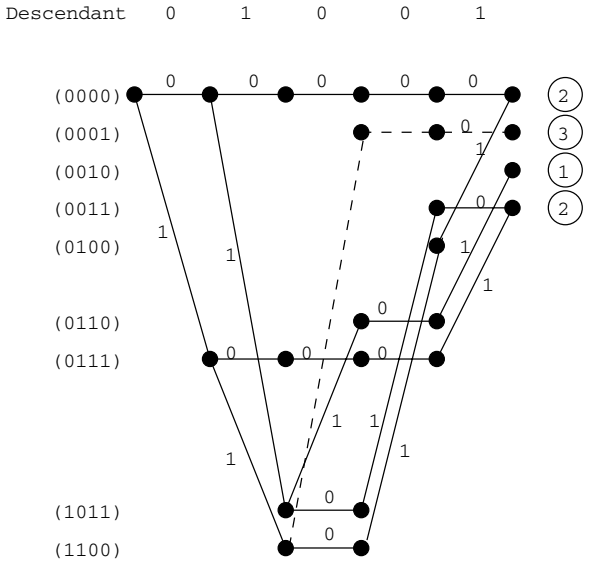



**Fig. 3.** TVA. Survivors at time $t = 5$

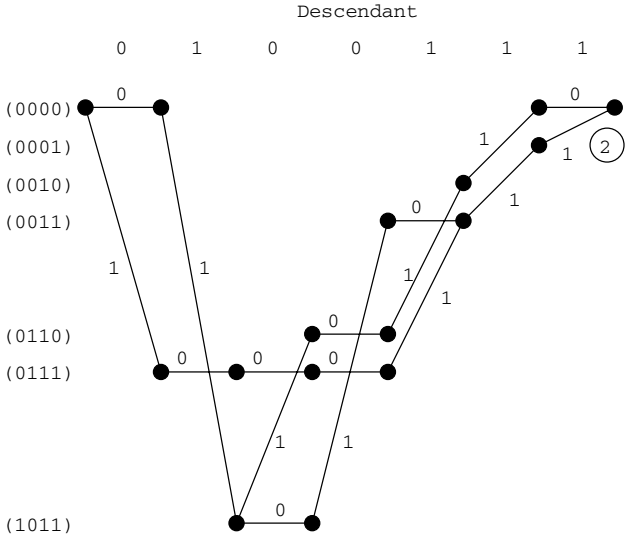



**Fig. 4.** TVA. Termination

The $S_r$ code, consists of $\mathbf{0}$ and $2^r - 1$ codewords of weight $2^{r-1}$, with every pair of codewords the same distance apart.

As an example we take the dual binary Hamming code $S_r$ with $r = 3$. This code is a $[7, 3, 4]$ code, and has the following parity check matrix.

$$\mathbf{H}_{S_3} = \begin{pmatrix} 0\,1\,1\,1\,0\,0\,0 \\ 1\,0\,1\,0\,1\,0\,0 \\ 1\,1\,0\,0\,0\,1\,0 \\ 1\,1\,1\,0\,0\,0\,1 \end{pmatrix}$$

Given a descendant $\mathbf{z}$, we need to search for codewords within distance 2 of $\mathbf{z}$. The search for parents of the descendant $(0,1,0,0,1,1,1)$, using the tracing Vitervi algorithm, is depicted in Figures 1 through 4. Note that a path is removed from consideration whenever its "length" amounts to 3. At time $n$ we have three surviving codewords $(0,1,0,1,0,1,1)$, $(0,1,1,0,1,1,0)$ and $(1,0,0,0,1,1,1)$, each one corresponding to a possible parent. Note that in this case the fingerprinting scheme is defeated.

## 5 Conclusions

This paper presents a new class of codes as fingerprinting codes. It is shown that in fact this new family of codes allow to trace the guilty with high probability in collusions of size 2, even when the colluders use their best strategy. When the equidistant code is also a parity check code, tracing the guilty in the fingerprinting scheme consists in decoding a block code beyond its error correction bound. We present an efficient tracing algorithm for this process.

## References

1. D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. *Advances in Cryptology-Crypto'95, LNCS*, 963:452–465, 1995.
2. A. Bonisoli. Every equidistant linear code is a sequence of dual hamming codes. *Ars Combinatoria*, 18:181–196, 1984.
3. G. Cohen, S. Encheva, and H. G. Schaathun. On separating codes. Technical report, ENST, Paris, 2001.
4. J. Domingo-Ferrer and J. Herrera-Joancomartí. Simple collusion-secure fingerprinting schemes for images. In *ITCC'00*, pages 128–132. IEEE Computer Society, 2000.
5. G. D. Forney. The Viterbi algorithm. *Proc. IEEE*, 61:268–278, 1973.
6. Y. L. Sagalovich. Separating systems. *Probl. Inform. Trans.*, 30(2):14–35, 1994.
7. Nambirajan Seshadri and Carl-Erik W. Sundberg. List Viterbi decoding algorithms with applications. *IEEE Trans. Comm.*, 42:313–323, 1994.
8. Jack K. Wolf. Efficient maximum likelihood decoding of linear block codes using a trellis. *IEEE Trans. Inform. Theory*, 24:76–80, 1978.

# $t$-Out-of-$n$ String/Bit Oblivious Transfers Revisited

Qianhong Wu[1], Bo Qin[1,2], Changjie Wang[3], Xiaofeng Chen[4], and Yumin Wang[1]

[1] State Key Lab. of ISN, Xidian Univ., Xi'an City 710071, P. R. China
woochanhoma@hotmail.com, ymwang@xidian.edu.cn
[2] School of Science,Xi'an Univ. of Tech. Xi'an City 710048, P. R. China
qinbo_77@hotmail.com
[3] Department of Computer Science and Engineering, Chinese Univ. of Hong Kong, Shatin, Hong Kong
cjwang@cse.cuhk.edu.hk
[4] School of Information Science and Technology, Sun Yat-sen Univ., Guangzhou 510275, P. R. China
isschxf@zsu.edu.cn

**Abstract.** In this paper, we focus on lowering the complexity of $t$-out-of-$n$ string/bit OTs for large $t$. The notion of oblivious public-key cryptosystem (OPKC) is introduced, in which Bob possesses $n$ public keys but only $t$ private keys and no one knows which $t$ private keys Bob possesses. If the sender, say, Alice, encrypts each message using the $n$ oblivious public keys, resp., the receiver, Bob, can obtain only $t$ messages by $t$ decryptions with his known $t$ private keys. This approach can be directly applied to $t$-out-of-$n$ bit OT. However, it is very inefficient due to heavy message expansion and many encryption/decryption operations. To construct $t$-out-of-$n$ bit OT, we introduce bit oblivious public-key cryptosystem (BOPKC), which is a special public-key cryptosystem with a message space of $n$ bits, and the private key only enables its owner to decrypt $t$ bits of $n$ secret bits. After an offline generation of such a BOPKC, it requires only one encryption, one decryption and one ciphertext. Finally, we show the concrete implementations of OPKC/BOPKC based on ElGamal/Paillier cryptosystem, and efficient $t$-out-of-$n$ string/bit OTs are achieved.

## 1 Introduction

Introduced by Rabin [21], oblivious transfer has been studied extensively in various flavor and security models (e.g., [2], [3], [4], [6], [7], [11], [12], [18], [22]). In particular, 1-out-of-2 bit OT (where $m_1$ and $m_2$ are bits) attracts much attention from researchers since it is the basic oblivious transfer scheme to which 1-out-of-2 string OT and 1-out-of-$n$ OT schemes are reduced. A generic approach

Robert H. Deng et al. (Eds.): ISPEC 2005, LNCS 3439, pp. 410–421, 2005.

to construct $t$-out-of-$n$ $k$-bit string OT is first to construct a basic 1-out-of-2 bit OT and then to construct 1-out-of-2 $k$-bit string OT by $k$ calls to the underlying 1-out-of-2 bit OT, and then to construct the 1-out-of-$n$ $k$-bit OT by invoking the basic 1-out-of-2 string OT for many runs, typically, $n$ or $\log_2 n$ runs (e.g., [4], [11], [18]), and then to construct the $t$-out-of-$n$ $k$-bit string OT by invoking the 1-out-of-$n$ $k$-bit string OT for $t$ runs. Clearly, this method requires at least $O(kt\log_2 n)$ calls of the basic 1-out-of-2 bit OT. The reduction approaches are widely studied (e.g., [4], [11], [18]). 1-out-of-$n$ string OT schemes are also possible to be built from basic techniques directly (e.g., [22], [23]). Efficient direct constructions of $t$-out-of-$n$ string OTs have been proposed (e.g.,[17],[25])(unfortunately, the schemes in [17]have been shown insecure [24]).

The oblivious transfer has been employed in many applications in cryptographic studies and protocol design such as secure multiparty computation, private information retrieval (PIR), fair electronic contract signing, oblivious secure computation (e.g., [5], [9], [12], [14]), and etc. Although the work by Ishai *et al* [15] shows it is possible to extend OT efficiently, i.e. implement large numbers of OTs from relatively small numbers of OTs, the result of Impagliazzo and Rudich [16] implies that it is unlikely that oblivious transfer could be based on more efficient one-way functions or other private-key cryptographic primitives. And hence the complexity of OTs may be quite demanding and they are likely to be the bottleneck in many applications.

### 1.1 Our Works and Comparison

Following the work by Bellare and Micali [6], we introduce the notion of oblivious public-key cryptosystem(OPKC), in which Bob has $n$ public keys but only $t$ private keys and which $t$ ones he knows is unknown for others. It is convenient to realize efficient $t$-out-of-$n$ string OT schemes by letting the sender, say, Alice, encrypt each message using the receiver's $n$ oblivious public keys respectively and send them to Bob. On receiving the $n$ ciphertexts from Alice, Bob can get only $t$ messages because he knows only $t$ private keys. After a preprocessing phase to generate such oblivious public keys, Alice needs exactly $n$ encryptions and Bob needs $t$ decryptions.

This construction can be directly applied to $t$-out-of-$n$ bit OT. Such a construction, however, is inefficient due to heavy message expansion and many encryption/decryption operations. We construct efficient $t$-out-of-$n$ bit OT by using bit oblivious public-key cryptosystem (BOPKC), in which Bob has a public encryption algorithm with a message space of $n$ bits and his private key only enables him to decrypt $t$ bits of the $n$ secret bits. In this scenario, after generating the BOPKC, Alice needs only one encryption and Bob needs one decryption. The complexity is independent of $t$.

For concrete implementations, we demonstrate how to obtain OPKC/BOPKC from ElGamal/Paillier cryptosystem. The proposed $t$-out-of-$n$ string OT is a generalization version of [6] due to Bellare *et al*, in which 1-out-of-2 OT and 2-out-of-3 OT were considered. Then we extend Stern's 1-out-of-$n$ OT [22] to $t$-out-of-$n$ bit OT without increasing complexity. Since the Fiat-Shamir heuristic

are applied in the OPKC/BOPKC generation stage to achieve non-interaction, the random oracle methodology are inexplicitly used in the security arguments. However, at a cost of three rounds of communication with the techniques in [10], the random oracle can be removed and the security of the protocols is achieved in the general model.

## 2 Definitions and General Constructions

### 2.1 *t*-Out-of-*n* String/Bit Oblivious Transfer

The $t$-out-of-$n$ (string/bit) OT is a natural extension of the 1-out-of-$n$ (string/bit) OT. We follow the definition in [23]. For a more strict definition, see [2].

**Definition 1.** *A $t$-out-of-$n$ (string/bit) OT is a two-party protocol in which Alice possesses $n$ (string/bit) secrets $m_1, m_2, \cdots, m_n$ and Bob has his secret choices $\sigma = \{i_1, \cdots, i_t\} \subseteq \{1, \cdots, n\}$. A $t$-out-of-$n$ OT scheme should satisfy the following requirements:*

- **Completeness:** *If both Alice and Bob follow the protocol, Bob gets $t$ secrets $m_j$ for $j \in \sigma$ after executing the protocol with Alice.*
- **Receiver's privacy:** *After executing the protocol with Bob, Alice shall not learn which $t$ secrets Bob has received.*
- **Sender's privacy:** *After executing the protocol with Alice, Bob gets no information about the other $n - t$ secrets $m_j$ for $j \notin \sigma$ or their combinations.*

The above definition does not address the issue of whether Alice is committed to her input prior to Bob's input. Similarly, another issue not handled is whether Alice knows which inputs and whether her input is independent of Bob's input. Although these requirements are significant, we do not require it since it seems that it is best to handle these issues at the level of the application, as suggested in [18].

### 2.2 *t*-Out-of-*n* Oblivious Public-Key Cryptosystem

**Definition 2.** *A $t$-out-of-$n$ oblivious public-key cryptosystem contains the following three algorithms:*

- $\mathbf{G}(1^\ell)$**:** *A probabilistic polynomial time algorithm which takes security parameter $\ell$ and outputs $\{K_i\}_1^n$ and $\{s_{i_1}, \cdots, s_{i_t}\}$, where $i_j \in \{1, 2, \cdots, n\}$ for $j = 1, 2, \cdots, t$.*
- $\mathbf{E}(\cdot, \cdot)$**:** *A (probabilistic) polynomial time algorithm which takes inputs $m_i \in \{0,1\}^k$ and $K_i$, and outputs $c_i \in \{0,1\}^l$, where $c_i = E(m_i, K_i)$ for $i \in \{1, 2, \cdots, n\}$.*
- $\mathbf{D}(\cdot, \cdot)$**:** *A deterministic polynomial time algorithm which takes inputs $c_j, s_j$ and outputs $m_j = D(c_j, s_j)$ for $j \in \sigma$ and null or random string in $\{0,1\}^k$ for $j \notin \sigma$, where $\sigma = \{i_1, i_2, \cdots, i_t\}$.*

Apart from the general security requirements of ordinary public-key cryptosystems, a $t$-out-of-$n$ oblivious public-key cryptosystem should meet the following two security properties:

- ***Choice Counter:*** A (an interactive) polynomial time algorithm which determines whether $|\sigma| \leq t$ or not, where $|\sigma|$ is the cardinality of $\sigma$.
- ***Choice Ambiguity:*** For any $\tau \subseteq \{1, 2, \cdots, n\}$ with $1 \leq |\tau| \leq t$, there exists no polynomial time algorithm to determine $\tau \subseteq \sigma$ or not.

Informally speaking, in a $t$-out-of-$n$ oblivious public-key cryptosystem, the owner of $n$ public keys has at most $t$ private keys and which $t$ ones he knows is unknown for others. Hence, the owner of $n$ public keys may be required to provide sufficient witnesses to show he knows only $t$ private keys without leaking which $t$ ones, for instance, in zero-knowledge manner.

### 2.2.1 General Construction of $t$-Out-of-$n$ String OT Using OPKC

The OPKC model allows a more efficient implementation of $t$-out-of-$n$ string OT for large $t$.

**[Offline Stage]**

- Bob runs $\mathrm{G}(1^\ell)$ and obtains public keys $\{K_i\}_1^n$ and private keys $\{s_{i_1}, \cdots, s_{i_t}\}$. Bob sends $\{K_i\}_1^n$ to Alice.
- Alice runs the choice counter algorithm to verifies that the number of private keys known by Bob is no more than $t$. If it is not the case, Alice aborts the following protocol.

**[Online Stage]**

Bob's Private Inputs: secrete choices $\sigma$; Alice's Private Inputs: secrete messages $m_i \in \{0,1\}^k$, $i = 1, 2, \cdots, n$. Bob's outputs: $m_i \in \{0,1\}^k$, $i \in \sigma$; Alice's outputs: null.

- Bob selects a uniformly random permutation $\pi$ on $\{1, 2, \cdots, n\}$ such that $\pi\{i_1, \cdots, i_t\} = \sigma$. Bob sends $\pi$ to Alice.
- Alice computes $C_{\pi(1)} = E(m_{\pi(1)}, K_{\pi(1)}), \cdots, C_{\pi(n)} = E(m_{\pi(n)}, K_{\pi(n)})$ and sends them to Bob.
- Bob obtains $m_j = D(C_j, s_j)$ for $j \in \sigma$.

For security, we reasonably assume that, without the private key corresponding to a public key, it is infeasible to extract the plaintext from the ciphertext generated with the underlying encryption algorithm $\mathrm{E}(\cdot, \cdot)$ and the public key. Since Bob knows $t$ private keys, he will obtain $t$ messages and the completeness of OT follows. Alice cannot learn which $t$ secrets Bob has received because the OPKC is of choice ambiguity, that is, Alice does not learn which $t$ private keys Bob knows. Hence, the receiver's privacy is achieved. The choice counter algorithm convinces one that Bob does not know the other $n - t$ private keys, and hence he cannot learn more than $t$ messages. After the offline stage, Bob needs $t$ decryptions and $n\mathtt{log}n$ bits and Alice $n$ encryptions and $nl$ bits. The computation overhead of selecting a random permutation is negligible.

### 2.3 $t$-Out-of-$n$ Bit Oblivious Public-Key Cryptosystem

The above construction can be directly applied to $t$-out-of-$n$ bit OT in which each secret is just one bit 0 or 1. In this case, one can part the plaintext space into two distinct sets $I_0$ and $I_1$ and denote the bit 0 by any message in $I_0$ and 1 by the message in $I_1$. However, it is impractical because of heavy message expansion and many encryption/decryption opertaitons. To achieve efficient implementation, we propose the notion of $t$-out-of-$n$ bit oblivious public-key cryptosystem (BOPKC).

**Definition 3.** *A $t$-out-of-$n$ bit oblivious public-key cryptosystem contains the following three algorithms:*

- **$\mathbf{G}(1^\ell)$:** *A probabilistic polynomial time algorithm which takes security parameter $\ell$ and outputs a public key $K$ and a secrete key $s$.*
- **$\mathbf{E}(\cdot,\cdot)$:** *A (probabilistic) polynomial time algorithm which takes inputs $K$ and $(m_1,\cdots,m_n) \in \{0,1\}^n$, outputs $c \in \{0,1\}^l$, where $c = E(m_1,\cdots,m_n,K)$ for $i \in \{1,2,\cdots,n\}$.*
- **$\mathbf{D}(\cdot,\cdot)$:** *A deterministic polynomial time algorithm which takes inputs $c, s$, and outputs $m_j = D(c,s)$ for $j \in \sigma$ and null or random bit in $\{0,1\}$ for $j \notin \sigma$, where $\sigma = \{i_1, i_2, \cdots, i_t\}$ is the choices of the owner of $K$.*

Apart from the general security requirements of ordinary public-key cryptosystems, a $t$-out-of-$n$ bit oblivious public-key cryptosystem should meet the following two security properties:

- ***Choice Counter:*** A (an interactive) polynomial time algorithm which determines whether $|\sigma| \leq t$ or not.
- ***Choice Ambiguity:*** For any $\tau \subseteq \{1,2,\cdots,n\}$ with $1 \leq |\tau| \leq t$, there exists no polynomial time algorithm to determine $\tau \subseteq \sigma$ or not.

Coarsely speaking, a $t$-out-of-$n$ bit oblivious pubic-key cryptosystem is a special public-key cryptosystem. It has a message space of $n$ bits. The private key only enables its owner to decrypt $t$ bits of an encrypted $n$-bit message without leaking which $t$ bits they are. Similarly, the owner may be required to show sufficient witnesses to convince one that it is the case. Obviously, when $t = n$, a $t$-out-of-$n$ BOPKC is an ordinary public-key cryptosystem.

#### 2.3.1 Construction of BOPKC Using Homomorphic Encryption

We show a BOPKC can be constructed using the standard public-key cryptosystem with certain properties. Indeed, if the message space of the underlying public-key cryptosystem has an efficient weak base (defined below), a $t$-out-of-$n$ BOPKC can be built from an ordinary public-key cryptosystem which is randomizable, homomorphic and semantically secure. We review here these related notions in brief.

A probabilistic encryption algorithm $E(\cdot)$ is randomizable if there exists a probabilistic polynomial-time algorithm $rand(\cdot)$ such that for any message $m$,

any instance $c_0 \longleftarrow E(m)$, $rand(c_0)$ is identically distributed to $E(m)$, that is, $rand(\cdot)$ maps one ciphertext instance to a uniformly random ciphertext of the same plaintext. An encryption algorithm $E(\cdot)$ is homomorphic if for any $m_0, m_1 \in M$ satisfying $m_0 \oplus m_1 \in M$, where $M$ is the message space, it holds that $E(m_0) \otimes E(m_1) = E(m_0 \oplus m_1)$, where $\oplus$ and $\otimes$ are two efficient operations defined in message space and ciphertext space, respectively. A probabilistic encryption algorithm $E(\cdot)$ is semantically secure if for any given message $a$ and $b \in M$, there exists no polynomial-time algorithm to distinguish $E(a)$ from $E(b)$ with non-negligible probability.

The construction involves also a weak base of the message space. $(b_1, \cdots, b_n) \in M^n$ is a weak base of the message space $M$ on operator $\oplus$ if for any $(x_1, \cdots, x_n) \in \{0,1\}^n, b = x_1 b_1 \oplus \cdots \oplus x_n b_n \in M$ has a unique representation, where $0 \in M$ and for any $b \in M, 0b = 0, 0 \oplus b = b$. Here, we do not differentiate 0 from a string of all 0's without confusion. A weak base $(b_1, \cdots, b_n) \in M^n$ is efficient if for any $b = x_1 b_1 \oplus \cdots \oplus x_n b_n \in M$, there exists a polynomial-time algorithm in $n$ to extract $(x_1, \cdots, x_n) \in \{0,1\}^n$ from $b$.

If a probabilistic encryption algorithm $E(\cdot)$ is randomizable, homomorphic and semantically secure, and the message space $M$ has an efficient weak base $(b_1, \cdots, b_n) \in M^n$, where $n \leq \mathtt{log}_2|M|$ and $|M|$ denotes the cardinality of $M$, one can construct a secure BOPKC with a new message space $\{0,1\}^n$.

- Select $(a_1, \cdots, a_n) \in M^n$, where $a_j = b_j$ for $j \in \{i_1, i_2, \cdots, i_t\} \subseteq \{1, \cdots, n\}$ and $a_j = 0$ for $j \notin \{i_1, i_2, \cdots, i_t\}$.
- Compute and publish $E(a_1), \cdots, E(a_n)$ as the public key. The private key is $\{a_j | j \in \sigma\}$.
- Prove that $n - t$ entities in $\{a_1, \cdots, a_n\}$ are 0's in zero-knowledge manner.
- Encryption: $c = rand(E(a_1)^{m_1} \otimes \cdots \otimes E(a_n)^{m_n})$, where $m_i \in \{0,1\}$, $i = 1, 2, \cdots, n$.
- Decryption: $D(c) = m_1 a_1 \oplus \cdots \oplus m_n a_n = \bigoplus_{j \in \sigma} m_j a_j$. Extract $m_j$ from $D(c)$ for $j \in \{i_1, i_2, \cdots, i_t\}$.

### 2.3.2 Construction of $t$-Out-of-$n$ Bit OT Using BOPKC

Now it is straightforward to construct $t$-out-of-$n$ bit OT using the above BOPKC.

**[Offline Stage]**

- Bob runs $\mathrm{G}(1^\ell)$ and obtains the public key $(E(a_1), \cdots, E(a_n))$ and the private key $\{a_j | j \in \sigma\}$. Bob sends $(E(a_1), \cdots, E(a_n))$ to Alice.
- Alice runs the choice counter algorithm to verifies that the number of Bob's choices is no more than $t$. If it is not the case, Alice aborts the following protocol.

**[Online Stage]**

Bob's Private Inputs: secrete choices $\sigma$; Alice's Private Inputs: secrete messages $m_i \in \{0,1\}$, $i = 1, 2, \cdots, n$. Bob's outputs: $m_i \in \{0,1\}$, $i \in \sigma$; Alice's outputs: null.

- Bob selects a uniformly random permutation $\pi$ on $\{1, 2, \cdots, n\}$ such that $\pi\{i_1, \cdots, i_t\} = \sigma$. Bob sends $\pi$ to Alice.
- For any $m_i \in \{0, 1\}$, where $m_i \in \{0, 1\}$, $i = 1, 2, \cdots, n$, Alice computes and sends $c = rand(E(a_{\pi(1)})^{m_{\pi(1)}} \otimes \cdots \otimes E(a_{\pi(n)})^{m_{\pi(n)}})$ to Bob, where $E(m)^1 = E(m)$, $E(m)^0 = 1$ and $1 \otimes E(m) = E(m)$ for any $m \in M$.
- Bob computes $D(c) = m_{\pi(1)}a_{\pi(1)} \oplus \cdots \oplus m_{\pi(n)}a_{\pi(n)} = \bigoplus_{j\in\sigma} m_j a_j$ and runs an efficient algorithm to extract $m_j$ for $j \in \sigma$.

Notice that for $E(a_j)$, the $j$-th bit is chosen by Bob if $j \in \sigma$, i.e., $a_j = b_j$, while it is out of choice if $j \notin \sigma$, i.e., $a_j = 0$. However, Alice cannot tell $E(b_j)$ from $E(0)$ because $E(\cdot)$ is semantically secure. Bob's proof in zero-knowledge manner does not leak any information about his choice. Hence, Alice cannot know which $t$ bits Bob has received. The receiver cannot learn the other $n - t$ bits because $c = E(\bigoplus_{j\in\sigma} m_j a_j)$ does not contain any information about the other $n - t$ bits out of choice.

After offline stage to generate BOPKC, Alice needs one encryption and $l$ bits, and Bob needs one decryption (the cost of computing a random permutation, $E(a_{\pi(1)})^{m_{\pi(1)}} \otimes \cdots \otimes E(a_{\pi(n)})^{m_{\pi(n)}}$ and extracting $m_j$ from $D(c)$ for $j \in \sigma$ is negligible in our implementation ) and $n\log n$ bits. The online complexity is almost optimal because at least one encryption and one decryption are needed and to represent the $t$ choices, Bob needs at least $\min\{t\log n, (n - t)\log n\}$ bits. The complexity is independent of $t$ and hence it is extremely preferable for large $t$. To our best knowledge, it is the most efficient construction for $t$-out-of-$n$ bit OT.

## 3 The Schemes

We only specify the construction of OPKC/BOPKC. The implementation of $t$-out-of-$n$ string/bt OT can be achieved by directly following the general construction. Our construction requires some standard zero-knowledge proofs of knowledge of discrete logarithms [8] and 1-out-of-n discrete logarithm [1](also in appendix for self-contain reason ).

### 3.1 OPKC Based on ElGamal Cryptosystem

The following scheme is an extension of the work due to Bellare *et al* [6] with judicious zero-knowledge proofs. In their original work, very special instances, i.e. noninteractive 1-out-of-2 and 2-out-of-3 string OTs are considered. Naor *et al* [19] extended the basic 1-out-of-2 OT to 1-out-of-$n$ OT and reduction method were used to improve its efficiency. The reduction method cannot be applied to the following scheme. In their instances, zero-knowledge proofs are unnecessary.

Let $G$ be a cyclic group of order $q$ where $q$ is a large prime, and $g$ is a generator of $G$ and $y = g^s$. The ElGamal public key is $(G, g, y)$ and the private key is $s$ [13]. The message space is $G\backslash\{0\}$. To encrypt a message $m$ in $G\backslash\{0\}$, one randomly selects $r \in Z_q^*$ and computes $a = g^{-r}$, $b = my^r$. The decryption is $m = ba^s$. We use a slightly modified version in the following.

Suppose that $h$ is an independent generator of $G$, which can be generated using a random oracle or provided by Alice, i.e. the discrete logarithm $\log_g h$ is unknown for Bob. Let $m_i \in \{0,1\}^k$ and $H : G \to \{0,1\}^k$ be a cryptographic hash function. Bob can generate a OPKC as follows.

**−Generation of oblivious public keys:**

Randomly select $s_1, \cdots, s_n \in Z_q^*$ and determine the choice set $\sigma = \{i_1, \cdots, i_t\} \subseteq \{1, \cdots, n\}$. Compute $y_j = h^{r_j} g^{s_j}, j = 1, \cdots, n$, where $r_j = 0$ if $j \in \sigma$ and $r_j = 1$ if $j \notin \sigma$. The public key is $(G, g, h, y_1, \cdots, y_n)$. The private key is $\{s_j\}$ for $j \in \sigma$.

**−Proofs of valid public keys:**

$$ZK\{r_j | y_j = h^{r_j} g^{s_j} \wedge r_j \in \{0,1\}\},\ j = 1, \cdots, n \text{ ([1] or Appendix)},$$
$$ZK\{s_1 + \cdots + s_n | y_1 \cdots y_n / h^{n-t} = g^{s_1 + \cdots + s_n}\} \text{([8])}.$$

**−Encryption:** $u \leftarrow Z_q^*$, $a = g^{-u}, b_j = m_j \oplus H(y_j^u), j = 1, \cdots, n$.

**−Decryption:** $m_j = b_j \oplus H(a^{s_j})$, $j \in \sigma$.

Clearly, the decryption procedure is correct. Now we consider the security of the above OPKC. The security proofs are of sketch.

**Theorem 1.** *The verification algorithm of the above zero-knowledge proofs is a valid choice counter.*

*Proof.* The second knowledge proof convinces one that $y_1 \cdots y_n / h^{n-t} = g^{s_1 + \cdots + s_n}$. Because Bob does not know $\log_g h$, it follows that $r_1 + \cdots + r_n = n - t$. The first knowledge proof convinces one that $r_j \in \{0,1\}$. Hence, there are $t$ 0's in $\{r_1, \cdots, r_n\}$. If $r_j = 0$, it is a valid ElGamal encryption and the corresponding ciphertexts can be decrypted using the known private key $s_j$. If $r_j = 1$, the corresponding ciphertext cannot be decrypted because he does know the private key. The verification algorithm of the above zero-knowledge proofs is a polynomial time algorithm.

**Theorem 2.** *The choice ambiguity is unconditional.*

*Proof.* It is sufficient to prove that for any $j$, an unbounded adversary cannot determine whether $r_j$ is 1 or 0. From lemma 1 (Appendix 2), Alice can obtain no information about $r_j$ from the knowledge proofs even if she has unlimited computational power. Let $u = \log_g h$ and the adversary kows it. Hence, $y_j = h^{r_j} g^{s_j} = h^0 g^{s_j + u r_j} = h^1 g^{s_j - u + u r_j}$ also leaks no information about whether $r_j$ is 1 or 0. Therefore, Bob's choice is unconditionally ambiguous.

### 3.2 BOPKC Based on Paillier Cryptosystem

Our scheme is probably most relate to the ANDOS scheme [22] by Stern, in which $t$ is limited to 1 and the cut-and-choose technique is used to obtain an oblivious

public key. Our extension does not introduce additional overhead when 1 is improved to $t$. More efficient and standard zero-knowledge proofs are integrated.

In [20], Paillier proposed a homomorphic public key cryptosystem. It is specified in brief. Set $N = pq$, where $p$ and $q$ are large primes, $2^l < N < 2^{l+1}$. As usual, $\phi$ denotes Euler Totient function and $\lambda$ Carmichael function taken on $N$, i.e., $\phi = (p-1)(q-1)$ and $\lambda = \mathtt{lcm}(p-1, q-1)$ in the present case. Recall that $|Z^*_{N^2}| = N\phi$ and that for any $w \in Z^*_{N^2}$, $w^\lambda = 1 \bmod N$ and $w^{N\lambda} = 1 \bmod N^2$. Denote by $\Theta_\alpha \subset Z^*_{N^2}$ the set of elements of order $N\alpha$ and by $Q$ their disjoint union for $\alpha\lambda$. $S_n = \{0 < u < N^2 | u = 1 \bmod N\}$ is multiplicative subgroup of integers modulo $N^2$ over which the function $L$ such that for $\forall u \in S_n, L(u) = (u-1)/N$ is well-defined. Let $g \in \Theta_\alpha$ for some $1 \leq \alpha \leq \lambda$. To encrypt a message $0 \leq m < N$, one randomly selects $r \in \{1, 2, \cdots, N-1\}$, and computes $c = g^{m+Nr} \bmod N^2$. Using the private key $\alpha$, one recovers the message $m = L(c^\alpha \bmod N^2)/L(g^\alpha \bmod N^2) \bmod N$.

Clearly, the Paillier cryptosystem is randomizable and homomorphic. It is semantically secure if and only if Decisional Partial Discrete Logarithm Problem is hard [20]. Notice that $\{2^0, 2^1, \cdots, 2^l\}(l < \mathtt{log}_2 N)$ is an efficient weak base of the plaintext space $0 \leq m < N$. Hence, a BOPKC can be constructed using the proposed method.

**−Generation of oblivious public key:**

Set $a_j = 0$ if $j \in \sigma$ and $a_j = 1$ if $j \notin \sigma$, $j = 1, 2, \cdots, n$. Randomly select $0 \leq r_j < N$ and compute $b_j = g^{a_j + Nr_j} \bmod N^2$. The oblivious public key is $\{N, g, b_1, \cdots, b_n\}$, and the private key is $\alpha$ and $\{a_j\}$ for $j \in \sigma$.

**−Proofs of valid public key:**

$$ZK\{r_1 + \cdots + r_n | b_1 \cdots b_n / g^t = g^{N(r_1 + \cdots + r_n)} \bmod N^2\} \text{ ([8])},$$
$$ZK\{a_j | b_j = g^{a_j + Nr_j} \bmod N^2 \wedge a_j \in \{0, 1\}\} \text{ ([1]or Appendix)}, j = 1, \cdots, n,$$

**−Encryption:** $c_j = b_j^{2^j} \bmod N^2$ for $j = 1, \cdots, n$. $v \leftarrow \{0, 1, \cdots, N-1\}, c = (c_1^{x_1} \cdots c_n^{x_n}) g^{vN} \bmod N^2$.

**−Decryption:** $x_{i_1} 2^{i_1} + \cdots + x_{i_t} 2^{i_t} = L(c^\alpha \bmod N^2)/L(g^\alpha \bmod N^2) \bmod N$.

The decryption algorithm is trivially correct. Now we consider the security of the scheme and the proofs are sketched.

**Theorem 3.** *The verification algorithm of the above zero-knowledge proofs is a valid choice counter.*

*Proof.* Similar to that in Theorem 2 but notice that $a_j + Nr_j$ is uniquely determined by $b_j = g^{a_j + Nr_j} \bmod N^2$.

**Theorem 4.** *The receiver's choices are ambiguous if the underlying Paillier System is semantically secure.*

*Proof.* The zero-knowledge proofs of receiver do not leak more information than the fact that there are $n-t$ 0's and $t$ 1's in $a_1, \cdots, a_n$. Since Bob's choice, 0 or 1, on the $j$-th bit is encrypted with Pailier cryptosystem, to extract the choice, an adversary has to determine whether Bob's $j$-th public-key component is $E(0)$ or $E(1)$. Hence, the receiver's choices are ambiguous if the underlying Paillier System is semantically secure.

### 3.3 Efficiency Analysis

There is a trivial method to lower the online complexity of $t$-out-of-$n$ OTs by running a random OT protocol before the choice set is decided. Hence, to achieve a fair comparison, we also compare the total complexity of our schemes with the most efficient reduction method.

**Table 1.** Comparison of communication complexity of $t$-out-of-$n$ OTs (bits)

| | Sender | Receiver | Typical Total Value |
|---|---|---|---|
| Our $k$-bit string OT | $nk+\ell$ | $(n+3)(\ell+\delta)+n\log n$ | $n(1184+k+\log n)+4576$ |
| Our bit OT | $2\ell$ | $n(2\ell+\delta+\log n)+2\ell+5\delta$ | $n(2208+\log n)+4896$ |
| OT with Reduction | $4t\ell\log n+nk$ | $(t+3)\ell\log n$ | $(5120t+3072)\log n+nk$ |

**Table 2.** Comparison of computation complexity of $t$-out-of-$n$ OTs (exponentiations)

| | Sender | Receiver | Typical Total Value |
|---|---|---|---|
| Our $k$-bit string OT | $2n+3$ | $2n+t+1$ | $4n+t+4$ |
| Our bit OT | $n+3$ | $2n+2$ | $3n+5$ |
| OT with Reduction | $4t\log n$ | $2t\log n$ | $6t\log n$ |

Here, $\ell$ is the binary length of the element in group $G$ or the RSA modulus, which is typically 1024. $\delta$ is the binary length of the output of the hash function involved in the zero-knowledge proofs, which is typically 160. When computing the complexity of $t$-out-of-$n$ OTs constructed with reduction method (with the reduction method, the complexity of $t$-out-of-$n$ bit OT is close to that of $t$-out-of-$n$ string OT), the most efficient reduction technique suggested in [18] and the most efficient 1-out-of-2 string OT [23] which we know are considered. Clearly, from the above tables, our schemes are more efficient when $t \geq n/\log n$. Further more, our schemes can work like a special public-key cryptosystem and the main overhead are in the preprocessing stage to generate the OPKC/BOPKC. And hence they enjoy two additional advantages, especially for our $t$-out-of-$n$ bit OT scheme. (1)They are very applicable to the *multi-sender-to-one-receiver* settings. (2)If the same OT is invoked many times, i.e. the receiver's choice does not change, our schemes provide a very efficient solution in this scenario.

## 4 Conclusions

We focus on $t$-out-of-$n$ string/bit OTs and lower their complexity. We introduce two models, namely, oblivious public-key cryptosystem and bit oblivious public-key cryptosystem, to implement $t$-out-of-$n$ bit/string OTs securely and efficiently. Concrete OPKC and BOPKC are proposed based on ElGamal/Pailler cryptosystem. $t$-out-of-$n$ bit/string OTs using these two models enjoy more efficiency for large $t$. They are especially applicable to the *multi-sender-to-one-receiver* application and the situation that the same OT is invoked many times.

## References

1. M. Abe, M. Ohkubo, and K. Suzuki. *1-out-of-n Signatures from a Variety of Keys*. Asiacrypt'02, Springer-Verlag, pp.415-432, 2002.
2. D. Beaver. *How to Break a "Secure" Oblivious Transfer Protocols*. Eurocrypt'92, LNCS 658, Springer-Verlag, pp.285-196, 1993.
3. G. Brassard, C. Crépeau, J.-M. Roberts. *All-or-Nothing Disclosure of Secrets*, Crypto'86, LNCS 263, Springer-Verlag, pp.234-238, 1987.
4. G. Brassard, C. Crépeau, M. Santha. *Oblivious Transfer and Intersecting Codes*. IEEE Trans. on Inf. Th., special issue in coding and complexity, Vol. 42, No. 6, pp.1769-1780, 1996.
5. M. Ben-Or, S. Goldwasser, A. Wigderson. *Completeness Theorems for Non-cryptographic Fault-tolerant Distributed Computation*. Proceedings of the 20th ACM Symposium on the Theory of Computing, pp.1-10, 1988.
6. M. Bellare, S. Micali. *Non-interactive Oblivious Transfer*. Crypto'89, LNCS 435, Springer-Verlag, pp.547-557, 1990.
7. C. Cachin. *On the Foundations of Oblivious Transfer*. Eurocrypt'98, LNCS 1403, Springer-Verlag, pp.361-374, 1998.
8. D. Chaum, J. H. Evertse, V. de Graaf. *An Improved Protocol for Demonstrating Possession of Discrete Logarithm and Some Generalizations*. Eurocrypt'87. Springer-Verlag, pp.127-141, 1987.
9. B. Chor, O. Goldreich, E. Kushilevitz, M. Susdan. *Private Information Retrieval*. Journal of the ACM 45(6), pp.965-982, 1998.
10. D. Chaum and T.P. Pedersen. *Wallet databases with observers*. Proc. CRYPTO'92, LNCS 740, pp. 89-105. Springer-Verlag, 1993.
11. C. Crépeau. *Equivalence between Two Flavors of Oblivious Transfers*. Crypto'87, LNCS 293, Springer-Verlag, pp.350-354, 1988.
12. S. Even, O. Goldreich, A. Lempel. *A Randomized Protocol for Signing Contracts*. Communications of the ACM 28, pp.637-647, 1985.
13. T. ElGamal. *A public-key cryptosystem and a signature scheme based on discrete logarithms*. Crypto'84, Springer-verlag pp.10-18, 1985.
14. O. Goldreich, R. Vainish. *How to Solve any Protocol Problem: An Efficient Improvement*. Crypto'87, LNCS 293, Springer-Verlag, pp.73-86, 1988.
15. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank1. *Extending Oblivious Transfers E.ciently*. CRYPTO 2003, LNCS 2729, pp.145-161, 2003.
16. R. Impagliazzo and S. Rudich. *Limits on the Provable Consequences of One-Way Permutations*. 20th ACM Symp. on the Theory of Computing, pp.44-61, 1989.

17. Y. Mu, J. Zhang, and V. Varadharajan. *m out of n oblivious transfer*. ACISP 2002, LNCS 2384, Springer-Verlag, pp.395-405, 2002.
18. M. Naor, B. Pinkas. *Oblivious Transfer and Polynomial Evaluation*. Proceedings of the 31st ACM Symposium on Theory of Computing, pp.145-254, 1999.
19. M. Naor, B. Pinkas. *Efficient Oblivious Transfer Protocols*. Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA) 2001, pp.448-457, 2001.
20. P. Paillier. *Public-key cryptosystems based on composite degree residuosity classes*. Eurocrypt'99, Springer-Verlag, pp.223-238, 1999.
21. M. Rabin. *How to Exchange Secrets by Oblivious Transfer*. Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981.
22. J. P. Stern. *A New and Efficient All-or-nothing Disclosure of Secrets Protocol*. Asiacrypt'98, LNCS 1514, Springer-Verlag, pp. 357-371, 1998.
23. W. Tzeng. *Efficient 1-out-of-n Oblivious Transfer Schemes*. PKC'02, LNCS 2274, Springer-Verlag, pp. 159-171, 2002.
24. G. Yao, F. Bao and R. H. Deng. *Security Analysis of Three Oblivious Transfer Protocols*. In the Proceedings of CCC'03, Progress in Computer Science and Applied Logic, BirkHauser Verlag, 2003. http://ccc.ustc.edu.cn/abstract/abstract.htm.
25. Q. Wu, J. Zhang , Y. Wang. *Practical t-out-n oblivious transfer and its applications*. ICICS 2003, LNCS 2836, Springer-verlag, pp.226-237, 2003

## Appendix: Knowledge Proof of 1-Out-of-$n$ D-Log

Let $h \in G$ be a generator of finite cyclic group$G$. $\varepsilon, \rho, \delta$ are security parameters and $H(\cdot) : 0,1^* \rightarrow \{0,1\}^\delta$ is a publicly available cryptographic hash function. The orders of finite cyclic groups involved in the following knowledge proofs are unknown for the participants. The proof is derived from the ones with order-known version by a careful choice of security parameters.

The following knowledge proof $ZK\{x, r|y = xh^r \wedge x \in \{a_1, a_2, \cdots, a_t\}\}$ is from [1] in which it works as 1-out-of-$n$ signature. It allows Alice to prove to Bob that she knows integers $x, r$ satisfying $y = xh^r$ and $x \in \{a_1, a_2, \cdots, a_t\}$, where $y, a_1, a_2, \cdots, a_t$, and $t$ are publicly known. Assume $x = a_k$, where $k \in \{1, 2, \cdots, t\}$.

- Alice randomly selects $s \in \pm\{0,1\}^{\varepsilon(\delta+\rho)+1}$ and $c_1, \cdots, c_{k-1}, c_{k+1}, \cdots, c_t \in \pm\{0,1\}^\delta$, and computes

$$c = H(h^s(y/a_1)^{c_1} \cdots (y/a_{k-1})^{c_{k-1}}(y/a_{k+1})^{c_{k+1}} \cdots (y/a_t)^{c_t}),$$
$$c_k = c \oplus c_1 \oplus \ldots \oplus c_{k-1} \oplus c_{k+1} \oplus \cdots \oplus c_t (\text{Here } \oplus \text{ means bitwise XOR}),$$
$$u = s - c_k r \text{ (in } Z).$$

  Then Alice sends $(u, c_1, \cdots, c_t)$ to Bob.
- Bob is convinced if $c_1 \oplus \cdots \oplus c_t = H(h^u(y/a_1)^{c_1} \cdots (y/a_t)^{c_t})$.

The following two lemmas are trivially derived from [1].

**Lemma 1.** *Bob cannot extract any information about $x$ even if he has unlimited computational power.*

**Lemma 2.** *Alice cannot cheat Bob if the discrete logarithm problem is hard.*

# Author Index