INFORMATION SECURITY 2011

Proceedings of the
Ninth Australasian Information Security Conference
(AISC 2011), Perth, Australia,
17-20 January 2011

Colin Boyd and Josef Pieprzyk, Eds.

Volume 116 in the Conferences in Research and Practice in Information Technology Series. Published by the Australian Computer Society Inc.

Published in association with the ACM Digital Library.
Table of Contents


Preface .............................................................................................................................. vii
Programme Committee .................................................................................................. viii
Organising Committee .................................................................................................... ix
Welcome from the Organising Committee ................................................................. x
CORE - Computing Research & Education ............................................................... xi
ACSW Conferences and the Australian Computer Science Communications .......... xii
ACSW and AISC 2011 Sponsors .................................................................................... xiv

Contributed Papers

An Anonymous Authentication Protocol with Single-database PIR ................................. 3
Toru Nakamura, Shunsuke Inenaga, Kensuke Baba, Daisuke Ikeda and Hiroto Yasuura

Cube Attack in Finite Fields of Higher Order ............................................................... 9
Andrea Agnesse and Marco Pedicini

Secure Two-Party Association Rule Mining ............................................................... 15
Md. Golam Kaosar, Russell Paulet and Xun Yi

Detection of Anomalies from User Profiles Generated from System Logs .................. 23
Malcolm Corney, George Mohay and Andrew Clark

Identification of Potential Malicious Web Pages ..................................................... 33
Van Lam Le, Ian Welch, Xiaoying Gao and Peter Komisarczuk

IEEE 802.11 Chipset Fingerprinting by the Measurement of Timing Characteristics ........ 41
Gnther Lackner and Peter Teufl

A Modal Logic for Information System Security ....................................................... 51
Yun Bai and Khaled Khan

Detection of Fast Flux Service Networks ............................................................. 57
Scott Campbell, Steve Chan and Jason Lee

Development and Evaluation of a Secure, Privacy Preserving Combinatorial Auction .... 67
Ben Palmer, Kris Bubendorfer and Ian Welch

Comparison of Low-Latency Anonymous Communication Systems - Practical Usage and Performance 77
Thorsten Ries, Radu State and Andriy Panchenko

Author Index ..................................................................................................................... 87
Preface

The Australasian Information Security Conference (AISC) 2011 was held on 18th-19th January 2011 in Perth, Australia, as a part of the Australasian Computer Science Week 2011. AISC grew out of the Australasian Information Security Workshop and officially changed the name to Australasian Information Security Conference in 2008. The main aim of the AISC is to provide a venue for Australasian and other researchers to present their work on all aspects of information security and promote collaboration between academic and industrial researchers working in this area.

This year we received 22 submissions from Austria, Australia, Germany, Italy, Japan, Luxembourg, New Zealand and United States. After a thorough refereeing process we accepted 10 papers for presentation at AISC 2011. We extend our thanks to all the AISC 2011 authors for their quality submissions and all the members of the Program Committee and additional referees for their expert reviews.

Following AISC tradition from previous years, we have selected a paper for the Best Student Paper Prize. Papers can be considered for this award only if the major contribution is due to a student author, who must be the first author of the paper. This year the award went to Ben Palmer from the Victoria University of Wellington for the paper "Development and Evaluation of a Secure, Privacy Preserving Combinatorial Auction" by Ben Palmer, Kris Bubendorfer and Ian Welch. Our hearty congratulations to Ben and his co-authors on this fine achievement!

The invited keynote address for AISC 2011 was presented by Gene Tsudik, Professor of Computer Science at the University of California, Irvine. We are very grateful to Gene for supporting AISC and delivering his lecture on usable security.

Special thanks go to Gleb Sechenov for his excellent work on maintaining the AISC 2011 website. We used Easychair software to manage the AISC submissions and reviews. We found this software very helpful and easy to use and we thank the maintainers of the service for this opportunity.

Last but not least we extend our gratitude to the ACSW 2011 chair Mihai Lazarescu and other members of the organising committee for their hard work and their continuous and invaluable support throughout the preparation of the conference.

Colin Boyd
Queensland University of Technology

Josef Pieprzyk
Macquarie University

AISC 2011 Programme Chairs
January 2011
Programme Committee

Chairs

Colin Boyd, Queensland University of Technology, Australia
Josef Pieprzyk, Macquarie University, Australia

Members

Joonsang Baek, Institute for Infocomm Research, Singapore
Lynn Batten, Deakin University, Australia
Ljiljana Brankovic, University of Newcastle, Australia
Raymond Choo, Australian Institute of Criminology, Australia
Asha Rao, RMIT University, Australia
Jason Reid, Queensland University of Technology, Australia
Ray Hunt, University of Canterbury, New Zealand
Ron Steinfeld, Macquarie University, Australia
Jill Slay, University of South Australia, Australia
Willy Susilo, University of Wollongong, Australia
Clark Thomborson, The University of Auckland, New Zealand
Ian Welch, Victoria University of Wellington, New Zealand
Huaxiong Wang, Nanyang Technology University, Singapore
Duncan S. Wong, City University of Hong Kong, Hong Kong SAR, China
Yang Xiang, Deakin University, Australia
Xun Yi, Victoria University, Australia
Organising Committee

Chair
Assoc. Prof. Mihai Lazarescu

Co-Chair
Assoc. Prof. Ling Li

Finance
Mary Simpson
Mary Mulligan

Catering and Booklet
Mary Mulligan
Dr. Patrick Puersum
Assoc. Prof. Mihai Lazarescu

Sponsorship and Web
Dr. Patrick Peursum
Dr. Aneesh Krishna

Registration
Mary Mulligan
Dr. Patrick Puersum

DVD and Signage
Dr. Patrick Puersum
Mary Mulligan

Venue
Dr. Mike Robey

Conference Bag
Dr. Sieteng Soh
Welcome from the Organising Committee

On behalf of the Australasian Computer Science Week 2011 (ACSW2011) Organising Committee, we welcome you to this year’s event hosted by Curtin University. Curtin University’s vision is to be an international leader shaping the future through its graduates and world class research. As Western Australia’s largest university, Curtin is leading the state in producing high quality ICT graduates. At Curtin Computing, we offer both world class courses and research. Our Computing courses cover three key areas in IT (Computer Science, Software Engineering and Information Technology), are based on the curricula recommendations of IEEE Computer Society and ACM, the largest IT professional associations in the world, and are accredited by the Australian Computer Society. Curtin Computing hosts a top level research institute (IMPCA) and offers world class facilities for large scale surveillance and pattern recognition.

We welcome delegates from over 18 countries, including Australia, New Zealand, USA, U.K., Italy, Japan, China, Canada, Germany, Spain, Pakistan, Austria, Ireland, South Africa, Taiwan and Thailand. We hope you will enjoy the experience of the ACSW 2011 event and get a chance to explore our wonderful city of Perth. Perth City Centre is located on the north bank of the Swan River and offers many fun activities and a wealth of shopping opportunities. For panoramic views of Perth and the river, one can visit Kings Park or enjoy a relaxing picnic in one of the many recreational areas of the park.

The Curtin University campus, the venue for ACSW2011, is located just under 10km from the Perth City Centre and is serviced by several Transperth bus routes that travel directly between Perth and Curtin University Bus Station, as well as several other routes connecting to nearby train services.

ACSW2011 consists of the following conferences:

- Australasian Computer Science Conference (ACSC) (Chaired by Mark Reynolds)
- Australasian Computing Education Conference (ACE) (Chaired by John Hamer and Michael de Raadt)
- Australasian Database Conference (ADC) (Chaired by Heng Tao Shen and Athman Bouguettaya)
- Australasian Information Security Conference (AISC) (Chaired by Colin Boyd and Josef Piepryzk)
- Australasian User Interface Conference (AUIC) (Chaired by Christof Lutteroth)
- Australasian Symposium on Parallel and Distributed Computing (AusPDC) (Chaired by Jinjun Chen and Rajiv Ranjan)
- Australasian Workshop on Health Informatics and Knowledge Management (HIKM) (Chaired by Kerryn Butler-Henderson and Tony Sahama)
- Computing: The Australasian Theory Symposium (CATS) (Chaired by Taso Viglas and Alex Potanin)
- Australasian Computing Doctoral Consortium (ACDC) (Chaired by Rachel Cardell-Oliver and Falk Scholer).

The nature of ACSW requires the co-operation of numerous people. We would like to thank all those who have worked to ensure the success of ACSW2011 including the Organising Committee, the Conference Chairs and Programme Committees, our sponsors, the keynote speakers and the delegates. Many thanks go to Alex Potanin for his extensive advice and assistance and Wayne Kelly (ACSW2010 chair) who provided us with a wealth of information on the running of the conference. ACSW2010 was a wonderful event and we hope we will live up to the expectations this year.

Assoc. Prof. Mihai Lazarescu and Assoc. Prof. Ling Li
Department of Computing, Curtin University
ACSW2011 Co-Chairs
January, 2011
CORE welcomes all delegates to ACSW2011 in Perth. CORE, the peak body representing academic computer science in Australia and New Zealand, is responsible for the annual ACSW series of meetings, which are a unique opportunity for our community to network and to discuss research and topics of mutual interest. The original component conferences ACSC, ADC, and CATS, which formed the basis of ACSW in the mid 1990s now share this week with six other events - ACE, AISC, AUIC, AusPDC, HIKM, ACDC, which build on the diversity of the Australasian computing community.

In 2011, we have again chosen to feature a small number of plenary speakers from across the discipline: Heng To Shen, Gene Tsudik, and Dexter Kozen. I thank them for their contributions to ACSW2011. I also thank the keynote speakers invited to some of the individual conferences. The efforts of the conference chairs and their program committees have led to strong programs in all the conferences again, thanks. And thanks are particularly due to Mihai Lazarescu and his colleagues for organising what promises to be a strong event.

In Australia, 2009 saw, for the first time in some years, an increase in the number of students choosing to study IT, and a welcome if small number of new academic appointments. Also welcome is the news that university and research funding is set to rise from 2011-12. However, it continues to be the case that per-place funding for computer science students has fallen relative to that of other physical and mathematical sciences, and, while bodies such as the Australian Council of Deans of ICT seek ways to increase student interest in the area, more is needed to ensure the growth of our discipline.

During 2010, CORE continued to negotiate with the ARC on journal and conference rankings. A key aim is now to maintain the rankings, which are widely used overseas as well as in Australia. Management of the rankings is a challenging process that needs to balance competing special interests as well as addressing the interests of the community as a whole.

CORE’s existence is due to the support of the member departments in Australia and New Zealand, and I thank them for their ongoing contributions, in commitment and in financial support. Finally, I am grateful to all those who gave their time to CORE in 2010; in particular, I thank Alex Potanin, Jenny Edwards, Alan Fekete, Aditiya Ghose, Leon Sterling, and the members of the executive and of the curriculum and ranking committees.

Tom Gedeon
President, CORE
January, 2011
ACSW Conferences and the
Australian Computer Science Communications

The Australasian Computer Science Week of conferences has been running in some form continuously since 1978. This makes it one of the longest running conferences in computer science. The proceedings of the week have been published as the Australian Computer Science Communications since 1979 (with the 1978 proceedings often referred to as Volume 0). Thus the sequence number of the Australasian Computer Science Conference is always one greater than the volume of the Communications. Below is a list of the conferences, their locations and hosts.

2012. Volume 34. Host and Venue - RMIT University, Melbourne, VIC.
2011. Volume 33. Host and Venue - Curtin University of Technology, Perth, WA.
2010. Volume 32. Host and Venue - Queensland University of Technology, Brisbane, QLD.
2008. Volume 30. Host and Venue - University of Wollongong, NSW.
2007. Volume 29. Host and Venue - University of Ballarat, VIC. First running of HDKM.
2006. Volume 28. Host and Venue - University of Tasmania, TAS.
1998. Volume 20. Hosts - University of Western Australia, Murdoch University, Edith Cowan University and Curtin University. Venue - Perth, WA.
1995. Volume 17. Hosts - Flinders University, University of Adelaide and University of South Australia. Venue - Glenelg, SA.
1990. Volume 12. Host and Venue - Monash University, Melbourne, VIC. Joined by Database and Information Systems Conference which in 1992 became ADC (which stayed with ACSW) and ACIS (which now operates independently).
1989. Volume 11. Host and Venue - University of Wollongong, NSW.
1987. Volume 9. Host and Venue - Deakin University, VIC.
1986. Volume 8. Host and Venue - Australian National University, Canberra, ACT.
1983. Volume 5. Host and Venue - University of Sydney, NSW.
1982. Volume 4. Host and Venue - University of Western Australia, WA.
1981. Volume 3. Host and Venue - University of Queensland, QLD.
1980. Volume 2. Host and Venue - Australian National University, Canberra, ACT.
1979. Volume 1. Host and Venue - University of Tasmania, TAS.
1978. Volume 0. Host and Venue - University of New South Wales, NSW.
Conference Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Conference Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACDC</td>
<td>Australasian Computing Doctoral Consortium</td>
</tr>
<tr>
<td>ACE</td>
<td>Australasian Computer Education Conference</td>
</tr>
<tr>
<td>ACSC</td>
<td>Australasian Computer Science Conference</td>
</tr>
<tr>
<td>ACSW</td>
<td>Australasian Computer Science Week</td>
</tr>
<tr>
<td>ADC</td>
<td>Australasian Database Conference</td>
</tr>
<tr>
<td>AISC</td>
<td>Australasian Information Security Conference</td>
</tr>
<tr>
<td>AUIC</td>
<td>Australasian User Interface Conference</td>
</tr>
<tr>
<td>APCCM</td>
<td>Asia-Pacific Conference on Conceptual Modelling</td>
</tr>
<tr>
<td>AusPDC</td>
<td>Australasian Symposium on Parallel and Distributed Computing (replaces AusGrid)</td>
</tr>
<tr>
<td>CATS</td>
<td>Computing: Australasian Theory Symposium</td>
</tr>
<tr>
<td>HIKM</td>
<td>Australasian Workshop on Health Informatics and Knowledge Management</td>
</tr>
</tbody>
</table>

Note that various name changes have occurred, which have been indicated in the Conference Acronyms sections in respective CRPIT volumes.
We wish to thank the following sponsors for their contribution towards this conference.

CORE - Computing Research and Education, www.core.edu.au


Australian Computer Society, www.acs.org.au

Curtin University of Technology, www.curtin.edu.au

QUT Information Security Institute, www.isi.qut.com
CONTRIBUTED PAPERS
An Anonymous Authentication Protocol with Single-database PIR

Toru Nakamura\(^1\)  Shunsuke Inenaga\(^1\)  Kensuke Baba\(^2\)  Daisuke Ikeda\(^1\)
Hiroto Yasuura\(^1\)

\(^1\) Graduate School/Faculty of Information Science and Electrical Engineering, Kyushu University
Moto’oka 744, Nishi-ku, Fukuoka, 819-0395, Japan
Email: \{toru, inenaga, yasuura\}@soc.aist.kyushu-u.ac.jp
daisuke@inf.kyushu-u.ac.jp
\(^2\) Research and Development Division, Kyushu University Library
10-1, Hakozaki 6, Higashi-ku, Fukuoka, 812-8581, Japan
Email: baba@lib.kyushu-u.ac.jp

Abstract

This paper focuses on anonymous authentication systems in multi-service environment, in which service providers communicate with the central manager in every authentication. Such systems have a merit that the central manager can easily update the database of user information by comparison to the existing anonymous authentication systems without communications between service providers and the central manager. The purpose of this paper is to realize a practical authentication protocol for such systems which satisfies four requirements for security and privacy protection, that is, correctness, impersonation resistance against passive insider, anonymity against central manager, and anonymity against service providers. The existing protocol consists of a multi-database PIR scheme, in which there are copies of the same database and none of these copies are allowed to communicate with each other. This paper proposes an authentication protocol which consists of the single-database PIR scheme proposed by Kushilevitz and Ostrovsky. This protocol also realizes all these requirements in the random oracle model. This protocol is more practical since using a single database implies the above-mentioned assumptions for multi-database PIR schemes are not required any more.

1 Introduction

With the increase of the number of services, users are forced to manage more pairs of a user ID (pseudonym) and a password. Hence much attention is recently paid to authentication systems in multi-service environment, which enable each user to have only a pair in order to use multiple services with a central manager. For example, single-sign-on systems such as Microsoft’s .NET Passport, Shibboleth, and OpenID, have been popular. In this paper, we focus on issues about user privacy such that activity or preference of a user can be revealed by (1) service providers or (2) a central manager. If a user submits his/her ID to multiple service providers and the central manager, information about what, when, and how often a user accesses can be collected. In order to solve such issues, an authentication protocol with anonymity against (1) service providers and (2) a central manager is essential. Authentication systems in multi-service environment can be classified according to which service providers must communicate with the central manager in every authentication. With respect to authentication systems without such communications, some protocols to realize the both kinds of anonymity are known, such as group signature schemes (Chaum & van Heyst 1991), anonymous credential schemes (Camenisch & Lysyanskaya 2002), and dynamic ID based anonymous authenticated key exchange schemes (Liao & Wang 2009). However, such protocols have a drawback that it is difficult for the central manager to deal with frequent queries to update the database of user information. Hence we focus on authentication systems with communications between service providers and the central manager. The requirements for an authentication system considered in this paper are the following.

- **Correctness**: if a user sends an authentication request with the valid password, every service provider accepts the request.
- **Impersonation resistance against passive insider**\(^1\): even if an adversary is a service provider, the adversary cannot impersonate a legitimate user.
- **Anonymity against service provider**: it is difficult for any service providers to obtain any information about a user ID.
- **Anonymity against central manager**: it is difficult for any central manager to obtain any information about a user ID.

There are few schemes which satisfy the previous requirements, as far as we know. Nakamura et al. (Nakamura et al. 2009) proposed an anonymous authentication protocol which satisfies all the requirements previously described. This protocol is based on private information retrieval (PIR) schemes (Chor et al. 1998)(Kushilevitz & Ostrovsky 1997). PIR schemes contribute for protecting privacy of a client who makes a query to a database server. Using a PIR scheme, the client can reconstruct an element from the answer which the database server has generated with the query, without the index of the element being revealed to the database server. The authentication protocol consists of a multi-database PIR scheme (Chor et al. 1998). This scheme requires the assumption that there are copies of the same database and none of these copies are allowed to communicate with each other. However, the assumption is not practical.

In this paper, we propose an authentication protocol with a single-database PIR scheme, which does not require copies of the same database. The protocol is called Single-database PIR based Anonymous Authentication Protocol (SPAAP for short). The first single-database

---

\(^1\) In this paper, a “passive and insider adversary” means that an adversary who is restricted to eavesdropping on messages that the service provider obtains.
PIR scheme, which is based on the quadratic residuosity assumption, is proposed by Kushilevitz and Ostrovsy (Kushilevitz & Ostrovsky 1997). The basic idea of realizing the authentication protocol is that (1) a user makes the query related his/her ID and encrypts the query with the public-key of the central manager, (2) the central manager decrypts the query and makes the answer related to the information to verify the user, and (3) the service provider reconstructs the information from the answer, where IDs correspond to indices of the database. If the service provider cannot obtain the ID, it is impossible to realize anonymity against service providers. However, original Kushilevitz and Ostrovsky’s single-database PIR scheme requires an index to reconstruct the element from the answer. Hence the single-database PIR scheme cannot be applied to our protocol. In this paper, we use the special version, in which an element of the database can be reconstructed without the index. Furthermore, we prove that SPAAP satisfies all the requirements under the quadratic residuosity assumption and the random oracle assumption (Bellare & Rogaway 1993).

SPAAP is more practical than the existing protocol (Namakura et al. 2009) since using a single database implies the assumptions for multi-database PIR schemes and not any more. Therefore, this paper contributes development of anonymous authentication systems in which service providers need to communicate with the central manager from the view point of reducing the impractical assumption.

The organization of this paper is shown as follows. In section 2, we provide some necessary definitions. In section 3, we introduce the definitions of the four requirements of anonymous authentication protocols. In section 4, we show the definition of the special version of single-database PIR and the detail of SPAAP. In section 5, we prove that SPAAP satisfies all the requirements.

2 Preliminaries

2.1 Notations

Let \( \mathbb{Z} \) denote the set of integers and \( \mathbb{N} \) denote the set of natural numbers. For a finite set \( X \), let \( |X| \) denote the number of elements which \( X \) contains. For \( x \in \mathbb{Z} \), let \( |x| \) denote the binary length of \( x \). For \( k \in \mathbb{N} \), let \( [k] = \{1, 2, \ldots, k\} \). For \( a, b \in \mathbb{Z} \), let \( a \mid b \) mean that \( b \) is divisible by \( a \). Let \( x \circ y \) be the concatenation of bit strings \( x \) and \( y \). We denote any polynomial of \( n \in \mathbb{N} \) by \( p(n) \), and some polynomial by \( \text{poly}(n) \).

An interactive Turing machine (ITM) (Goldreich 2001) is a Turing machine which has a pair of communication tapes in addition to a common input tape, a local input tape, an output tape, and a work tape. A joint computation of two ITMs is a sequence of pairs of the local configurations. The output of a joint computation is the output of one of the ITMs. The output of a Turing machine \( A \) on an input \( x \) is denoted by \( A(x) \). We denote by \( \langle A, B \rangle \) a joint computation of Turing machines \( A \) and \( B \), and by \( A(y), B(z) \) its output on a common input \( x \), a local input \( y \) for \( A \), and a local input \( z \) for \( B \). We sometimes omit the brackets if the input is empty. In the rest of this paper, we sometimes call a Turing machine \( A \) an “algorithm” \( A \) and a joint computation \( \langle A, B \rangle \) a “protocol” \( \langle A, B \rangle \). The idea of a joint computation of two ITMs can be extended straightforwardly to that of three ITMs by two pairs of communication tapes.

For random variables \( X, Y \) distributed over a set \( Z \), let

\[
\text{Pr}[X = Y] = \sum_{x,y \in Z} \text{Pr}[X = x] \cdot \text{Pr}[Y = y] \cdot \chi(x, y),
\]

where \( \chi \) is a predicate such that \( \chi(a, b) = 1 \) if \( a = b \), and \( \chi(a, b) = 0 \) otherwise. The output of a probabilistic algorithm \( A \) is determined by given inputs and random sources (called coin tosses). Assuming that coin tosses are given as local inputs, we can regard a probabilistic algorithm as a deterministic algorithm. Let \( A_\rho \) be a deterministic algorithm corresponding to a probabilistic algorithm \( A \). We assume that coin tosses \( \rho \) is a \( t \)-bit string. For random variables \( X, Y \) distributed over a set \( Z \) and \( x, y \in Z \), let

\[
\text{Pr}[A(x) = y] = \frac{\{ \{ A_\rho(x, \tau) = y \} \}}{2^t},
\]

\[
\text{Pr}[A(X) = y] = \sum_{x \in Z} \text{Pr}[X = x] \cdot \text{Pr}[A(x) = y],
\]

\[
\text{Pr}[A(X) = Y] = \sum_{x, y \in Z} \text{Pr}[X = x] \cdot \text{Pr}[Y = y].
\]

2.2 Indistinguishability

Definition 1 For any \( m \in \mathbb{N} \), two sequences of random variables \( X = (X^{(1)}, X^{(2)}, \ldots, X^{(m)}) \) and \( Y = (Y^{(1)}, Y^{(2)}, \ldots, Y^{(m)}) \) whose elements are distributed over \( \{0, 1\}^{\text{poly}(k)} \) are (computationally) indistinguishable if for any \( k \in \mathbb{N} \), any probabilistic polynomial-time algorithm \( B \)

\[
\text{Pr}[B(1^k, X^{(1)}, X^{(2)}, \ldots, X^{(m)}) = 1] - \text{Pr}[B(1^k, Y^{(1)}, Y^{(2)}, \ldots, Y^{(m)}) = 1] < \frac{1}{p(k)}.
\]

Definition 2 A sequence of random variables \( X \) which are distributed over \( \{0, 1\}^{\text{poly}(k)} \) is constructible if there exists a probabilistic polynomial-time algorithm \( S \) such that for any \( k \in \mathbb{N} \), the sequence of random variables \( S(1^k) \) and \( X \) are identically distributed.

Lemma 1 For any \( k \in \mathbb{N} \), any \( m \in \mathbb{N} \), any constructible sequences of random variables \( X = (X^{(1)}, X^{(2)}, \ldots, X^{(m)}) \) and \( Y = (Y^{(1)}, Y^{(2)}, \ldots, Y^{(m)}) \) distributed over \( \{0, 1\}^{\text{poly}(k)} \), if for any \( i \in [m] \), \( X^{(i)} \) and \( Y^{(i)} \) are indistinguishable, then \( X \) and \( Y \) are indistinguishable.

Proof: This can be proven easily by the standard hybrid argument (Goldreich 2001).

2.3 Quadratic Residuosity Assumption

For \( a \in \mathbb{Z} \), let \( \{a\} = \{x \in \mathbb{Z} | x \equiv a \mod n \} \) (\( \{a\} \) is called the residue class modulo \( n \) containing \( a \)). For \( n \in \mathbb{N} \), let

\[
\mathbb{Z}_n^* = \{x | 1 \leq x \leq n, \gcd(n, x) = 1\}.
\]

The quadratic residuosity predicate \( \mathcal{W}_n \) is defined as follows:

\[
\mathcal{W}_n(y) = \begin{cases} 
0 & \text{if } \exists \omega \in \mathbb{Z}_n^* \text{ such that } \omega^2 = y \mod n \\
1 & \text{otherwise}
\end{cases}
\]

For a positive odd \( n \), let \( \left( \frac{x}{n} \right) \) denote the Jacobi symbol of \( x \mod n \). Let

\[
\mathbb{Z}_n^{+1} = \{x \in \mathbb{Z}_n^* | \left( \frac{x}{n} \right) = +1\}.
\]
Let \( QR_n^+ = \{ x \in \mathbb{Z}_n^+ | W_n(x) = 0 \} \), \( QNR_n^+ = \{ x \in \mathbb{Z}_n^+ | W_n(x) = 1 \} \).

Informally, the Quadratic Redudosity Assumption is the assumption that there is no probabilistic polynomial-time algorithm for computing the predicate \( W_n(x) \). We show the definition of the assumption as follows.

**Definition 3 (Quadratic Redudosity Assumption)** For \( k \in \mathbb{N} \), let \( I_k = \{ n | n = \alpha \cdot \beta, \alpha \) and \( \beta \) are distinct primes, \( \| \alpha \| = \| \beta \| = k \} \). For any \( k \in \mathbb{N} \), any probabilistic polynomial-time algorithm \( B \):

\[
\Pr[B(N, X) = W_N(X)] < \frac{1}{2} + \frac{1}{p(k)},
\]

where \( N \) is a random variable uniformly distributed over \( I_k \) and \( X \) is a random variable uniformly distributed over \( Z_n^+ \).

### 3 Requirements of Anonymous Authentication Protocol

In this section, we introduce the authentication model which we assume in this paper and the definitions of the four requirements of anonymous authentication protocols.

#### 3.1 Authentication Model

In this paper, we assume an authentication model which consists of the following three types of entities.

- **User**: Let \( m \) be the number of the users. Each user is assigned the unique identifier \( i \in [m] \) and has a password \( x_i \in \{0, 1\}^\ell \) for a natural number \( \ell \). (Note that \( \ell \) is a polynomial of a security parameter \( k \).)
- **Service provider**: A service provider verifies whether the entity who has sent an authentication request is truly the legitimate user.
- **Central manager**: A central manager stores the sequence \( x = (x_1, x_2, ..., x_m) \) of the passwords of the users. We assume that each password is a random string.

Throughout this paper, we assume that

- each user can communicate only with service providers,
- each service provider can communicate with users and the central manager, and
- the central manager can communicate only with service providers.

Fig.1 is the authentication model that describes which pairs of entities can communicate each other. (\( U \): a user, \( S \): a service provider, \( C \): a central manager)

- **Correctness**: for any \( k, \ell, m \in \mathbb{N} \), any \( i \in [m] \), any \( x = \{x_i | i \in [m], x_i \in \{0, 1\}^\ell \} \),

\[
\Pr[(P(1^k, i, x_i), V(1^k), M(1^k, x)) = 1] > 1 - \frac{1}{p(k)}.
\]

- **Impersonation resistance against passive insider**: for any \( k, \ell, m \in \mathbb{N} \) any \( i \in [m] \), and any probabilistic polynomial-time algorithm \( B \),

\[
\Pr[(B(1^k, T_1), V(1^k), M(1^k, x)) = 1] < \frac{1}{p(k)},
\]

where \( X \) is a random variable uniformly distributed over \( \{0, 1\}^m \) and \( T_1 \) is a random variable which means a transcript of \( V \)'s local tape and read tapes after running \( (P(i, x), V, M(x)) \) where \( x \) is a sample from \( X \).

- **Anonymity against central manager**: for any \( k, \ell, m \in \mathbb{N} \) any \( i, j \in [m] \), any \( z, z' \in \{0, 1\}^\ell \), and any probabilistic polynomial-time algorithm \( B \),

\[
| \Pr[B(1^k, T_2) = 1] - \Pr[B(1^k, T_3) = 1] | < \frac{1}{p(k)},
\]

where \( X \) is a random variable uniformly distributed over \( \{0, 1\}^m \) and \( T_2 \) is a random variable which means a transcript of \( M \)'s local tape and read tapes after running \( (P(i, z), V, M(x)) \) where \( x \) is a sample from \( X \). Similarly, \( T_3 \) means a transcript after running \( (P(j, z'), V, M(x)) \).

- **Anonymity against service provider**: for any \( k, \ell, m \in \mathbb{N} \) any \( i, j \in [m] \), any \( z, z' \in \{0, 1\}^\ell \), and any probabilistic polynomial-time algorithm \( B \),

\[
| \Pr[B(1^k, T_4) = 1] - \Pr[B(1^k, T_5) = 1] | < \frac{1}{p(k)},
\]

where \( X \) is a random variable uniformly distributed over \( \{0, 1\}^m \) and \( T_4 \) is a random variable which means a transcript of \( V \)'s local tape and read tapes after running \( (P(i, z), V, M(x)) \) where \( x \) is a sample from \( X \). Similarly, \( T_5 \) means a transcript after running \( (P(j, z'), V, M(x)) \).

### 3.2 Requirements

We show the four requirements which an anonymous authentication protocol \( \langle P, V, M \rangle \) should satisfy as follows.
4 Our Approach: SPAAP

In this section, we show the anonymous authentication protocol which satisfies all the requirements, called SPAAP. We construct SPAAP with a special version of Kushilevitz and Ostrovsky’s single-database PIR schemes (Kushilevitz & Ostrovsky 1997), in which an element of the database can be reconstructed without the index.

4.1 Kushilevitz and Ostrovsky’s PIR scheme

For the ease of explanation, we assume that an element of a database is a bit, that is, a database is denoted by \( x = x_1 \oplus x_2 \oplus \cdots \oplus x_m \in \{0, 1\}^m \). We note that it is easy to modify this simpler scheme to treat a database of \( \ell \)-bit strings (for example, repeating this simpler scheme for \( \ell \) times).

- **Query algorithm** \( Q(\cdot, \cdot) \): \( Q \) is a probabilistic algorithm which receives \( 1^k \) and an index \( i \in [m] \) (\( k \) is a security parameter) as inputs. First, \( Q \) randomly chooses distinct primes \( \alpha \) and \( \beta \) whose length is \( k/2 \). Next, \( Q \) uniformly and randomly chooses \( m \) primes \( y_1, \ldots, y_m \in \mathbb{Z}_n^+ \). Finally, \( Q \) outputs \( y_1, \ldots, y_m \) as a query and \( (\alpha, \beta) \) as a secret.

- **Answer algorithm** \( A(\cdot, \cdot, \cdot) \): \( A \) is a deterministic algorithm which receives \( 1^k \), a database \( x \in \{0, 1\}^m \), and a query \( y_1, \ldots, y_m \in \mathbb{Z}_n^+ \) as inputs. \( A \) computes
  \[
  w_i = \begin{cases} 
  y_i^2 & \text{if } x_i = 0 \\
  y_i & \text{if } x_i = 1.
  \end{cases}
  \]

  Then, \( A \) outputs as an answer
  \[
  z = \prod_{i=1}^{m} w_i.
  \]

- **Reconstruct algorithm** \( \mathcal{R}(\cdot, \cdot, \cdot) \): \( \mathcal{R} \) is a deterministic algorithm which receives \( 1^k \), a secret \( (\alpha, \beta) \), and answer \( z \in \mathbb{Z}_n^+ \) as inputs. \( \mathcal{R} \) outputs 1 if \( W_n(z) = 1 \), and outputs 0 otherwise.

The PIR scheme satisfies the following properties under the quadratic residuosity assumption.

- **correctness**: for any \( k, m \in \mathbb{N} \), any \( x = \{x_i | i \in [m], x_i \in \{0, 1\}\} \), and any \( i \in [m] \),
  \[
  \Pr[\mathcal{R}(1^k, Q^2(1^k, i), A(x, Q^2(1^k, i))) = x_i] > 1 - \frac{1}{p(k)}. \tag{1}
  \]

- **privacy**: for any \( k, m \in \mathbb{N} \), any \( i, j \in [m] \), and any probabilistic polynomial-time algorithm \( B \),
  \[
  |Pr[B(1^k, Q^2(1^k, i)) = 1] - Pr[B(1^k, Q^2(1^k, j)) = 1]| < \frac{1}{p(k)}. \tag{2}
  \]

We prove the following lemma with respect to the PIR scheme. This lemma also holds in the modified scheme for a database \( x = \{x_i | i \in [m], x_i \in \{0, 1\}\} \) of \( \ell \)-bit strings. In the rest of paper, a PIR scheme means the modified scheme.

**Lemma 2** If \( (Q, A, \mathcal{R}) \) is the previous described PIR scheme, the following proposition holds: for any \( k, m \in \mathbb{N} \), any \( i, j \in [m] \), and any probabilistic polynomial-time algorithm \( B \),
\[
\Pr[B(1^k, Q^2(1^k, i), A(1^k, X, Q^2(1^k, i))) = 1] = \Pr[B(1^k, Q^2(1^k, j), A(1^k, X', Q^2(1^k, j))) = 1] = 0,
\]
where \( X, X' \) are random variables uniformly and independently distributed over \( \{0, 1\}^m \).

**proof**: Let \( I'_k = \{(\alpha, \beta) | (\alpha, \beta) \text{ are distinct primes, } ||\alpha|| = ||\beta|| = k\} \). \( Q^2(1^k, i) \) and \( Q^2(1^k, j) \) are (information theoretical) indistinguishable because both of them are random variables uniformly distributed over \( I'_k \).

Let \( n = \alpha \cdot \beta \), each \( U = U_1 \oplus U_2 \oplus \cdots \oplus U_m \) and \( U' = U_1' \oplus U_2' \oplus \cdots \oplus U_m' \) be a random variable uniformly distributed over \( \{1, 2\}^m \). For \( 1 \leq i \leq m - 1 \), let each \( Y_i \) and \( Y_i' \) be a random variable uniformly distributed over \( QR_{n}^\pm \). Let each \( V \) and \( V' \) be a random variable uniformly distributed over \( QR_{n}^\pm \). In the PIR scheme, \( (A(1^k, X, Q^2(1^k, i))) \) corresponds to \( Y_i U_1 \cdots Y_i U_m \). Similarly, \( A(1^k, X', Q^2(1^k, j)) \) corresponds to \( Y_i' U_1' \cdots Y_i' U_m' \).

Since multiplication is commutative,
\[
\Pr[B(1^k, Y_i U_1 \cdots Y_i U_m)] = 1 = \sum_{u \in \{1, 2\}^m} \sum_{b = 1}^{m-1} \sum_{y \in QR_{n}^\pm} \Pr[U = u] \cdot \Pr[V = v] \cdot \prod_{c=1}^{m} \Pr[Y_c = y_c] \cdot \Pr[B(1^k, y_1 U_1' \cdots y_m U_m')] = 1 = \sum_{u' \in \{1, 2\}^m} \sum_{b = 1}^{m-1} \sum_{y' \in QR_{n}^\pm} \Pr[U' = u'] \cdot \Pr[V' = v'] \cdot \prod_{c=1}^{m} \Pr[Y_c' = y_c'] \cdot \Pr[B(1^k, y_1 U_1' \cdots y_m U_m')] = 1.
\]

Hence \( A(1^k, X, Q^2(1^k, i)) \) and \( A(1^k, X', Q^2(1^k, j)) \) are (information theoretical) indistinguishable in the PIR scheme. By Lemma 1,
\[
\Pr[B(1^k, Q^2(1^k, i), A(1^k, X, Q^2(1^k, i))) = 1] = \Pr[B(1^k, Q^2(1^k, j), A(1^k, X', Q^2(1^k, j))) = 1] = 0.
\]

\[\square\]

4.2 SPAAP

We use a public-key encryption scheme and a random oracle as a hash function in order to construct SPAAP.

We show the definition of a public-key encryption scheme (Goldreich 2001) as follows.

**Definition 4** A semantically secure public-key encryption scheme is a triple \((G, E, D)\) of probabilistic polynomial-time algorithms satisfying the following conditions.


• On input $1^k$, algorithm $G$ outputs a pair of bit strings.
• For any pair of $(e, d)$ in the range of $G(1^k)$, and any $\gamma \in \{0, 1\}^*$,
  \[
  \Pr[D(d, E(e, \gamma)) = \gamma] = 1.
  \]
\[
(3)
\]
• For any $k \in \mathbb{N}$ any $x, y \in \{0, 1\}^{\text{poly}(k)}$, and any probabilistic polynomial-time algorithm $B$,
  \[
  |\Pr[ B(1^k, H(x)) = 1] - \Pr[ B(1^k, H(y)) = 1]| < \frac{1}{p(k)}. \]
\[
(4)
\]
In this paper, we assume that we can regard any hash function as a random oracle (that is, the random oracle model) (Bellare & Rogaway 1993). This assumption is called the random oracle assumption. In the random oracle model, all entities can interact with a random oracle $H$, that is a single function which is uniformly chosen from all possible functions. We note that if the random oracle $H$ receives the same input, $H$ answers the same output. We assume that the random oracle outputs $m$ bit strings on inputs $\ell$ bit strings, where $\ell$ and $m$ are polynomials of a security parameter $k$. The following lemma holds.

**Lemma 3** For any $k \in \mathbb{N}$, any $x, y \in \{0, 1\}^{\text{poly}(k)}(x \neq y)$, and probabilistic polynomial-time algorithm $B$,\[
|\Pr[B(1^k, H(x)) = 1] - \Pr[B(1^k, H(y)) = 1]| = 0.
\]

SPAAP $(\mathcal{P}, \mathcal{V}, \mathcal{M})$, which satisfies the all requirements; correctness, impersonation resistance against passive insider, anonymity against central manager, and anonymity against service provider, is shown as follows, where $(\mathcal{Q}, \mathcal{A}, \mathcal{R})$ is the Kushilevitz and Ostrovsky’s PIR scheme which described in the previous section.

1. $\mathcal{M}$ computes $(e, d) \leftarrow G(1^k)$ and publishes $e$.
2. $\mathcal{P}$ computes $(q, s) \leftarrow Q(1^k, i)$ and sends $(E(e, q), s)$ to $\mathcal{V}$.
3. $\mathcal{V}$ sends $E(e, q)$ to $\mathcal{M}$.
4. $\mathcal{M}$ obtains $q$ by decrypting $E(e, q)$. $\mathcal{M}$ randomly chooses $c \in \{0, 1\}^{\ell}$ for any $j \in [m]$ computes $x_j' \leftarrow H(x_j, c)$. Let $x' = (x_1', x_2', \ldots, x_m')$. $\mathcal{M}$ computes $a \leftarrow A(1^k, x', q)$ and sends $(c, a)$ to $\mathcal{V}$.
5. $\mathcal{V}$ computes $x_i' \leftarrow R(1^k, s, a) = H(x_i, c)$ and sends $c$ to $\mathcal{P}$.
6. $\mathcal{P}$ computes $z' \leftarrow H(z, c)$ where $z$ is a candidate password, and sends $z'$ to $\mathcal{V}$.
7. $\mathcal{V}$ outputs 1 if $z' = x_i'$, and outputs 0 otherwise.

5 Security Analysis

**Theorem 1** SPAAP has correctness under the quadratic residuosity assumption and the random oracle assumption.

**proof:** In Step 2, $q$ is always decrypted by Equality (3). In Step 5, the probability that $x_i' = H(x_i, c)$ is higher than $1 - 1/p(k)$ by Inequality (1). Hence if $z = x_i$, the probability that $z' = x_i'$ is higher than $1 - 1/p(k)$. \[\Box\]

**Theorem 2** SPAAP has impersonation resistance against passive insider under the quadratic residuosity assumption and the random oracle assumption.

**proof:** The main idea of this proof is that an adversary who has no pre-knowledge can simulate the transaction which is given to the service provider.

We prove that by contradiction. It is clearly (information theoretic) hard for any adversary to impersonate a legitimate user, if the adversary can obtain no pre-knowledge about $x$. That is, for any probabilistic polynomial-time algorithm $B$,
\[
\Pr[ B(1^k, V(1^k), M(1^k, X)) = 1] = \frac{1}{2^\ell} < \frac{1}{p(k)},
\]
\[
(5)
\]
where $X$ is a random variable uniformly distributed over $(\{0, 1\}^\ell)^m$.

The random variable $T_1$ is $\{E^1(1^k), Q^1(i), Q^2(i), c, A(\langle H(x_1, c), \ldots, H(x_m, c)\rangle, Q^1(i), H(z, c))\}$, where $x_1, \ldots, x_m$ are samples from $\{0, 1\}^{\ell}$, and $c$ is a sample from $\{0, 1\}^{\ell}$. Let $T'_1$ be $\{E^1(1^k), 1^{Q^1(i)}\}, Q^1(i), c, A(y_1, \ldots, y_m, Q^1(i), u\}$, where $y_1, \ldots, y_m$ are samples from $\{0, 1\}^{\ell}$, and $c$ and $u$ are samples from $\{0, 1\}^{\ell}$. By Inequality (4), $E^1(1^k)$ and $Q^1(i)$ are indistinguishable. By the basic property of a random oracle, $A(\langle H(x_1, c), \ldots, H(x_m, c)\rangle, Q^1(i))$ and $A(y_1, \ldots, y_m, Q^1(i))$ are indistinguishable. By Lemma 1, $T_1$ and $T'_1$ are indistinguishable. We assume that SPAAP does not have impersonation resistance against passive insider, that is, there exists some polynomial $q$ and some probabilistic polynomial-time algorithm $D$ such that
\[
\Pr[ D(1^k, T_1, V(1^k), M(1^k, X)) = 1] \geq \frac{1}{q(k)}.
\]
\[
(6)
\]
We derive contradiction by constructing a probabilistic polynomial-time algorithm $D'$ which takes $1^k$ as an input and uses the algorithm $D$ as a subroutine. $D'$ proceeds as follows.

1. $D'$ computes $(e, d) \leftarrow G(1^k)$ and randomly chooses $c, y, u$.
2. $D'$ computes $t_2 = \{E^1(1^k), 1^{Q^1(i)}\}, Q^2(i), c, A(y, Q^1(i), u\}$.
3. $D'$ outputs $D(1^k, t_2)$.

By Inequality (6), it holds that
\[
\Pr[ D'(1^k, V(1^k), M(1^k, X)) = 1] \geq \frac{1}{q(k)},
\]
because $T_1$ and $T'_1$ are indistinguishable. This contradicts to Inequality (5).

**Theorem 3** SPAAP has anonymity against central manager under the quadratic residuosity assumption and the random oracle assumption.

**proof:** We prove that by contradiction. The random variable $T_2$ is $\{G^2(1^k), E^1(1^k), Q^1(i)\}$ and random variable $T'_2$ is $\{G^2(1^k), E^1(1^k), Q^1(j)\}$. We assume that SPAAP does not have anonymity against central manager, that is, there exists some polynomial $q$ and some probabilistic polynomial-time algorithm $D$ such that
\[
\Pr[ D(1^k, T_2) = 1] - \Pr[ D(1^k, T'_2) = 1] \geq \frac{1}{q(k)}.
\]
\[
(7)
\]
We derive contradiction by constructing a probabilistic polynomial-time algorithm $D'$ which takes $1^k$ and $y$ as inputs and uses the algorithm $D$ as a subroutine. $D'$ proceeds as follows.
1. \( D' \) computes \((e, d) \leftrightarrow G(1^k) \).

2. \( D' \) outputs \( D(d, E(e, y)) \).

By Inequality (7), it holds that

\[
\begin{align*}
&\left| \Pr[D(1^k, Q_1(1^k, i)) = 1] \right| - \Pr[D(1^k, Q_1(1^k, j)) = 1]\right| \geq \frac{1}{p(k)}.
\end{align*}
\]

This contradicts to Inequality (2). \( \square \)

**Theorem 4** SPAAP has anonymity against service provider under the quadratic residuosity assumption and the random oracle assumption.

**proof:** The random variable \( T_4 \) is \( \{E(G^1(1^k), Q^1(i)), Q^2(i), c, A((H(x_1, c), \ldots, H(x_m, c)), Q^1(i)), H(z, c)\} \), and the random variable \( T_5 \) is \( \{E(G^1(1^k), Q^1(j)), Q^2(j), c, A((H(x_1, c), \ldots, H(x_m, c)), Q^1(j)), H(z', c)\} \) where \( x_1, \ldots, x_m \) are samples from \( \{0, 1\}^l \), and \( e \) is a sample from \( \{0, 1\}^l \). By Inequality (4), \( E(G^1(1^k), Q^1(i)) \) and \( E(G^1(1^k), Q^1(j)) \) are indistinguishable. By Lemma 2 and the basic property of a random oracle, \( Q^2(i), A((H(x_1, c), \ldots, H(x_m, c)), Q^1(i)) \) and \( Q^2(j), A((H(x_1, c), \ldots, H(x_m, c)), Q^1(j)) \) are indistinguishable. By Lemma 3, \( H(z, c) \) and \( H(z', c) \) are (information theoretical) indistinguishable. Therefore, \( T_4 \) and \( T_5 \) are indistinguishable by Lemma 1. \( \square \)

6 Conclusions

In this paper, we proposed SPAAP, which consists of the special version of the single-database PIR scheme proposed by Kushilevitz and Ostrovsky, in which an element of the database can be reconstructed without the index. We proved that SPAAP satisfies all the requirements: correctness, impersonation resistance against passive insider, anonymity against central manager, and anonymity against service providers under the quadratic residuosity assumption and the random oracle assumption. SPAAP is more practical than the existing protocol (Nakamura et al. 2009) since using a single database implies the assumptions for multi-database PIR schemes are not required any more.

Acknowledgements

This work was in part supported by CREST-DVLSI of JST. We are grateful for their support.

References


Cube attack in finite fields of higher order

Andrea Agnesse\(^1\)  Marco Pedicini\(^2\)

\(^1\) Dipartimento di Matematica, Università Roma Tre
Largo San Leonardo Murialdo 1, Rome, Italy

\(^2\) Istituto per le Applicazioni del Calcolo “Mauro Picone”
Consiglio Nazionale delle Ricerche
Viale Manzoni 30, 00185 Rome, Italy

Email: marco@iac.cnr.it

Abstract

We present in full details a version of the Dinur-Shamir Cube Attack (Dinur & Shamir 2009) for a generic finite field of order \(q\). In particular, when applied to multivariate monomials of degree \(d\) in \(k < d\) variables, the attack acts exactly in the same way if the selected monomial was using the degree \(k\) monomial in the same \(k\) variables.

Keywords: Algebraic cryptanalysis, Cube Attack.

1 Introduction

The Cube Attack is a new cryptographic attack based on multivariate polynomials over \(\mathbb{F}_2\) suitable for both block and stream ciphers. In (Dinur & Shamir 2009), authors introduced this methodology as a variant of algebraic attacks at aiming a way to distill from a cryptographic encoding function a set of linear relations involving secret parameters (e.g., key bits) by means of tweakable ones (e.g., plaintext or initial vectors). The basic requirement for the attack is the possibility of describing the cryptographic scheme as a function in \(m + n\) variables that can be partitioned in public \(x_1, \ldots, x_n\) (i.e., that can be chosen during the attack, therefore tweakable in accord with (Dinur & Shamir 2009)) and private variables \(k_1, \ldots, k_m\) (i.e., those variables that have to be determined during the attack). Note that public variables can represent bits of the initial vector but in other scenarios, they could be bits of the key, or bits of the plaintext, see for instance (Aumasson et al. 2009), (Joux 2009).

As usual, the goal of the attack is to find the value of the private variables: by obtaining from the enciphering function enough linear relations that have to be satisfied by these variables and having a way to connect them with ciphertext. Whenever the number of independent linear relations is equal to the number of variables, the system can be solved. In order to do this the attacker has to evaluate the enciphering function by choosing assignments for both public and private variables. The values to be used are determined as an application of the following two theorems:

**Theorem 1** For every polynomial \(p\) and for any sub-set of indices of variables \(I\), we define

\[
p_I := \sum_{v \in C_I} p|_{x_i=x_{i+1}}
\]

where \(C_I\) is the set of \(n\)-tuples such that the elements of index \(i \in I\) take all the possible combinations of values 0/1, while the ones with index \(i \notin I\) remain undetermined as a variable \(x_i\). So each element of \(C_I\) is a formal combination of boolean values and variables, and \(p_I\) is a polynomial which does not depend on variables with index in \(I\).

Then \(p_I = p_{S(I)}\) where \(p_{S(I)}\) is the quotient of the euclidean division of \(p\) by \(t_I := \prod_{i \in I} x_i\).

The quotient \(p_{S(I)}\) is called superpolynomial of the term \(t_I\). If for some index set \(I\), the corresponding polynomial \(p_{S(I)}\) is linear, then \(t_I\) is called maxterm of \(p\), and the following holds:

**Theorem 2** Let \(t_I\) be a maxterm of a polynomial \(p\), so that its superpoly is \(p_{S(I)} = a_0 + a_1 x_1 + \ldots + a_n x_n\), and let \(X\) and \(X_j\) be the sets \(X = \{x \in \mathbb{F}_2^n : x_i = 0 \text{ for all } i \notin I\}\) and \(X_j = \{x \in \mathbb{F}_2^n : x_i = 0 \text{ for all } i \notin I \cup \{j\}\}\) and \(x_j = 1\). Then

1. \(a_0 = \sum_{x \in X} p|_x\)
2. \(a_j = a_0 + \sum_{x \in X_j} p|_x\) for all \(j \notin I\).

The two theorems above can be easily proven considering that in \(\mathbb{F}_2\) the sum equals the difference and the fact that the characteristic of the field is 2. In the next sections we show how the theorems can be easily generalized to every finite field \(\mathbb{F}_p\).

In Section 2, we introduce the attack as presented in (Dinur & Shamir 2009). In Section 3, we describe the various phases of the attack when the polynomial representation of the enciphering function is available, while in Section 4 we describe a strategy to perform the attack in a realistic scenario, i.e., when we do not have the explicit expression of the polynomial but we have only access to it. Therefore, it can be accessed as a “black box” function. In Section 5, we present the original contribution of this paper by discussing the cube attack in \(\mathbb{F}_2\) that was only claimed as possible in (Dinur & Shamir 2009).

2 Scenario of the attack

The basic requirement for the attack is that the cryptographic scheme can be expressed as a multivariate function in \(m + n\) variables over \(\mathbb{F}_2\). Then we
may think this enciphering function as a polynomial $p(v_1, \ldots, v_m, x_1, \ldots, x_n)$. The crucial point in the cube attack is that there are variables that can be chosen, these variable are here denoted by $v_1, \ldots, v_m$ and they are called public variables. These variables are for instance, known plaintext variables or they are variables associated to bits of the initial vector. On the other hand, there are variables that the attacker cannot control, they are denoted by $x_1, \ldots, x_n$ and in typical cases they are the secret variables which contain the key bits.

The aim of the attack is to “solve” the polynomial, i.e., to find the values of the secret variables.

The attack can be divided in two distinct phases: in the preprocessing phase the goal is to derive from the polynomial $p$ only linear relations containing only the secret variables to create a solvable linear system of equations; at this scope the attacker can evaluate the polynomial by suitably choosing both public and secret variables. The matrix associated to the found linear system can easily be inverted. Then, in the online phase, secret variables can be attacked by using the ciphertext to compute known terms of the found linear system.

In this way the secret variables, which are set to evaluate the polynomial by suitably choosing both public and secret variables. The matrix associated to the found linear system can easily be inverted. Then, in the online phase, secret variables can be attacked by using the ciphertext to compute known terms of the found linear system.

In this way the secret variables, which are set to evaluate the polynomial by suitably choosing both public and secret variables. The matrix associated to the found linear system can easily be inverted. Then, in the online phase, secret variables can be attacked by using the ciphertext to compute known terms of the found linear system.

In this way the secret variables, which are set to evaluate the polynomial by suitably choosing both public and secret variables. The matrix associated to the found linear system can easily be inverted. Then, in the online phase, secret variables can be attacked by using the ciphertext to compute known terms of the found linear system.

### Definition 1
A term $t_I$ is a maxterm if $deg(p_{S(I)}) \equiv 1$.

Given a subset $I$ of indices of size $k$, say $I = \{i_1, \ldots, i_k\}$, we define the cube $C_I$ as the set of $n$-tuples such that the elements of index $j \in I$ takes all the possible combinations of values 0/1, while the ones with index $j \not\in I$ remain undetermined as a variable $x_j$. So each element of $C_I$ is a formal combination of boolean values and variables.¹

¹For instance: if $n = 3$ and $I = \{1, 2\}$, then $C_I = \{(0, 0, x_3), (0, 1, x_3), (1, 0, x_3), (1, 1, x_3)\}$.
Note 1. Note that the search of the matrix of coefficients $A$ of system (2) (and its eventually inversion) can be done in a preprocessing phase, since it is independent of the key used in the encryption.

Example 1. Let us consider the polynomial

$$p(v_1, v_2, v_3, v_4, x_1, x_2) = v_1 v_2 v_3 + v_1 v_2 v_4 + v_2 v_3 v_4 + v_1 v_2 x_1 + v_1 v_2 x_2 + v_2 v_3 x_1 + v_2 v_3 x_2 + v_1 v_2 + v_3 x_2 + v_3 x_1 + v_1 + v_3 + x_1 x_2 + 1$$

over $\mathbb{F}_2$. We want to recover the key $k = (0, 1)$.

In the preprocessing phase we look for maxterms which are product of only the public variables $v_1, \ldots, v_4$. Such maxterms are

$$t_{1,2} = v_1 v_2, \quad t_{2,3} = v_2 v_3, \quad t_{1,3} = v_1 v_3, \quad t_{1,4} = v_1 v_4,$$

while their superpolies, with the eventually other public variables set to zero, are

$$p_{S(1,2)} = x_2 + 1, \quad p_{S(2,3)} = x_1, \quad p_{S(1,3)} = x_1 + x_2 + 1, \quad p_{S(1,4)} = x_2.$$

Among them, we choose two superpolies in order to create a square linear system with a unique solution. In this example we choose $p_{S(1,2)}$ and $p_{S(1,3)}$.

The preprocessing phase ends by setting out the system

$$\begin{align*}
x_2 &= p_{S(1,2)}(k) + 1 \\
x_1 + x_2 &= p_{S(1,3)}(k) + 1
\end{align*}$$

and calculating the inverse of the matrix of coefficients,

$$A^{-1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \in M_2(\mathbb{F}_2).$$

In the online phase we can evaluate the polynomial in some chosen values of only the public variables, and it is sufficient to find the free terms of the system (3), as

$$p_{S(1,2)}(k) = \sum_{v \in C_{1,2}} p|v(k)$$

and

$$p_{S(1,3)}(k) = \sum_{v \in C_{1,3}} p|v(k).$$

Finally, we are able to recover the key used by solving system (3) in polynomial time with the use of the matrix $A^{-1}$ previously calculated,

$$k = A^{-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = (0, 1).$$

Generic ciphers usually implement the Shannon idea of confusion/diffusion and they have very complicated and huge polynomial representations; as a consequence, any polynomial representing a cipher should be so chaotic that it is correct to suppose that its structure is indistinguishable from a random polynomial of a certain degree $d$, i.e., polynomial in which each monomial of degree at most $d$ can occur with probability $\frac{1}{2}$. We actually need only a weaker condition on the polynomial, since we are merely interested in maxterms which are products just of public variables.

Definition 2. A polynomial $p$ of $m$ public variables, $n$ secret variables and degree $d$ is a d-random polynomial if each term of degree $d$ which is the product of $d - 1$ public variables and one secret variable is independently chosen to occur with probability $\frac{1}{2}$.

Note that in a d-random polynomial each term being product of $d - 1$ public variables is a maxterm with probability $1 - 2^{-d}$, as it is not a maxterm only if all the terms which contain the same $d - 1$ public variables and any secret variables do not appear in $p$.

Thus, with the hypothesis above, written the system (2) as $Ax = b$, we can suppose that every entry in the binary matrix $A$ is chosen randomly. So, in order to estimate the probability that $A$ is invertible, we use the following lemma:

Lemma 1. The probability that a random matrix $A \in M_n(\mathbb{F}_2)$ is invertible is

$$\prod_{i=1}^{n} \left(1 - \frac{1}{2^n}\right).$$

Proof: We recall that a square matrix $A$ of order $n$ is invertible if and only if its rank is maximum and equals to $n$, i.e., if all its rows are linearly independent. This means that, for $i = 1, \ldots, n$, the $i$th row must be linearly independent from all the previous ones, so that it can be chosen in $2^n - 2^{i-1}$ different ways out of the $2^n$ possible $n$-tuples of elements in $\mathbb{F}_2$. Thus, the probability that $A$ is invertible is

$$\prod_{i=1}^{n} \frac{2^n - 2^{i-1}}{2^n} = \prod_{i=1}^{n} \left(1 - \frac{1}{2^{i-1}}\right) = \prod_{i=1}^{n} \left(1 - \frac{1}{2}\right).$$

□

Remark 1. It is easy to show that the sequence $S_n = \prod_{i=1}^{n} \left(1 - \frac{1}{2^i}\right)$ is decreasing and converge to a positive value, as stated by the equivalence

$$\prod_{i=1}^{\infty} (1 - \theta_i) > 0 \iff \sum_{i=1}^{\infty} \theta_i < \infty,$$

where $\theta_i \in [0, 1]$. As you can see from Figure 1, this value is approximated by 0.28879.

Figure 1: Probability $\Pr[\det(A) \neq 0]$ expressed as a function of the order of the matrix $A$.

This means that the probability that system (2) has a unique solution can be made arbitrary close to 1 increasing the number of maxterms taken into account.
The “black box” attack

If the polynomial expression is not available, for instance because we have to deal with a huge polynomial or because the internal structure of the cipher has been kept secret, we have to find the required maxterms in a more complex way. In this case, in fact, we have to proceed with a random walk tweaking both public and secret variables, which can always be done in the preprocessing phase. This is possible because of the following theorem, in which we do not distinguish between public and secret variables.

**Theorem 4** Let \( t_I \) be a maxterm in a polynomial \( p(x_1, \ldots, x_n) \), so that its superpoly is \( p_S(I)(x_1, \ldots, x_n) = a_0 + a_1 x_1 + \ldots + a_n x_n \), and let \( X \) and \( X_j \) be the sets \( X = \{ x \in \mathbb{F}_2^n : x_i = 0 \text{ for all } i \notin I \} \) and \( X_j = \{ x \in \mathbb{F}_2^n : x_i = 0 \text{ for all } i \notin I \cup \{ j \} \) and \( x_j = 1 \}. \) Then

1. \( a_0 = \sum_X p(x) \);
2. \( a_j = a_0 + \sum_{X_j} p(x) \) for all \( j \notin I \).

**Proof:** We recall that, given the maxterm \( t_I \), the polynomial \( p \) can be written as \( p = t_I p_S(I) + q_I \).

\[
\sum_X p(x) = \sum_X [t_I(x) p_S(I)(x)] + q_I(x) = \sum_X t_I(x) p_S(I)(x) + \sum_X q_I(x) = p_S(I)(0, \ldots, 0) = a_0
\]

since the second sum is zero modulo 2 because, as already mentioned, each term in \( q_I \) is a monomial that lacks at least a variable with index in \( I \). As a consequence, the values obtained when the monomial is evaluated on \( C_I \) are summed an even number of times. On the other hand, in the first sum, the result is that only in one case the terms \( t_I(x)p_S(I)(x) \) take a value different from 0, namely when \( t_I \) is evaluated to 1: in this case, since the superpoly does not contain any variable with index in \( I \), it is evaluated only with its \( n-k \) arguments all equal to zero.

\[
\sum_{X_j} p(x) = \sum_{X_j} [t_I(x) p_S(I)(x)] + q_I(x) = \sum_{X_j} t_I(x) p_S(I)(x) + \sum_{X_j} q_I(x) = a_0 + a_j
\]

since the second sum is zero and in the first one the only term summed is the one for which \( t_I = 1 \), but in this case the top vertex element \( v^* \) has a 1 also in the \( j \)th position, so, with the free term of the superpoly \( p_S(I) \) also the coefficient of the term \( x_j \) is summed.

In Algorithm 4.1, we report the strategy presented in (Dinur & Shamir 2009), which can be applied when the polynomial representation of the cryptosystem can not be assumed to be a \( d \)-random polynomial.

For what concern the testing of the linearity of the superpoly, Dinur and Shamir suggest to use a probabilistic linearity test, as for instance the BLR test, which consists in choosing independently and randomly vectors \( a, b \in \mathbb{F}_2^n \) and verifying the condition

\[
p_S(I)(a) + p_S(I)(b) + p_S(I)(0, \ldots, 0) = p_S(I)(a + b).
\]

**Algorithm 4.1 Random walk for the Cube Attack.**

**Require:** the number \( m \) of public variables

(1) choose a random value \( k \) such that \( 1 \leq k \leq m \) choose a subset \( I \) of indices such that \( |I| = k \)

(2) compute the superpoly with Theorem 4

if \( p_S(I) \) is a constant \( \{ I \text{ is too large} \} \) then

drop a variable from \( I \) and go to (1)

end if

if \( p_S(I) \) is nonlinear \( \{ I \text{ is too small} \} \) then

add a public variable to \( I \) and go to (1)

end if

correct choice for \( I \) are between the two cases above; if such an \( I \) does not exist go to (1)

end when enough suitable maxterms have been collected

For each element \( v \in C_I \) we denote by \( p_{vl} \) the polynomial in \( n-k \) variables \( p(v) \). Moreover, we define the top vertex \( v^* \) of the cube \( C_I \) as

\[
v^*_i := \begin{cases} x_i & \text{if } x_i \text{ does not appear in } t \\ 1 & \text{if } x_i \text{ appears in } t \end{cases}
\]

**5 Cube attack in \( \mathbb{F}_q \)**

We consider a polynomial \( p \in \mathbb{F}_q[x_1, \ldots, x_n] \) of degree \( d = \deg(p) \) (without distinguishing between public and secret variables) and a monomial \( t = x_{i_1}^{r_1} \cdots x_{i_k}^{r_k} \), where \( 0 \leq r_i < q \) for \( 1 \leq i \leq k \). For any monomial \( t \), we can factor \( p \) as before

\[
p = t \cdot p_S(I) + q_I
\]

where \( q_I \in \mathbb{F}_q[x_1, \ldots, x_n] \) is the sub-polynomial of \( p \) which is the sum of all the terms of \( p \) which are not divisible by \( t \). Note that, differently from the case \( \mathbb{F}_2 \), the superpoly \( p_S(I) \) can actually contain some of the variables of \( t \), while both \( p_S(I) \) and \( q_I \) can even have terms containing all the variables \( x_{i_1}, \ldots, x_{i_k} \); we denote this sub-polynomial of \( q_I \) as \( q_I' \).

**Example 2** Let us consider the monomial \( t = x_1^2 x_2^2 \) and the polynomial \( p \in \mathbb{F}_2[x_1, \ldots, x_5] \):

\[
p = x_1^3 x_2^3 x_3 + x_1^2 x_2 x_4 + x_1^2 x_3^3 + x_2^2 x_3 x_3 + + x_1^3 x_2 x_4 + x_1^2 x_2 x_3 + x_1 x_2 x_3 + x_1 x_2 x_3 + x_2 x_4 + x_2 x_5 + x_3 x_4 + x_4 x_5 + x_5 x_6 + + 1 =
\]

In this case

\[
p_{S(t)} = x_1 x_2 + x_2 x_4 + x_2 + x_3 + x_4,
\]

and

\[
q_I = x_1 x_2 + x_1 x_2 x_3 + x_1 x_2 x_3 + x_1 x_2 x_3 + x_2 x_4 + x_2 x_3 + x_2 + 1.
\]

We have a similar result as in (Dinur & Shamir 2009), by considering the \( k \)-dimensional Boolean cube

\[
C_t = \{(x_1, \ldots, x_n) : x_i \in \{0, 1\} \subset \mathbb{F}_q \}
\]

for each variable appearing in \( t \).

For each element \( v \in C_t \) we denote by \( p_{vl} \) the polynomial in \( n-k \) variables \( p(v) \). Moreover, we define the top vertex \( v^* \) of the cube \( C_t \) as

\[
v^*_i := \begin{cases} x_i & \text{if } x_i \text{ does not appear in } t \\ 1 & \text{if } x_i \text{ appears in } t \end{cases}
\]
Let us denote by $W_H(v)$ the number of ones in $v$, i.e., it would be like the Hamming weight if we were in $\mathbb{F}_2$, but here we have vectors with elements which are 1, 0 or variables. Note that the weight of the top vertex is $k$ by definition $W_H(v^*) = k$.

Analogously to Theorem 3 we have the following result:

**Theorem 5** Let us define

$$p_t := \sum_{v \in C_t} (-1)^{w(v)} p_v$$

with $w(v) := W_H(v^*) + W_H(v) = k + W_H(v)$.

Then

$$p_t = (p_{S(t)} + q_t^*)_{|v^*}$$

**Proof:** In a way similar to the proof of Theorem 3, we obtain

$$\sum_{v \in C_t} (-1)^{w(v)} p_v = \sum_{v \in C_t} (-1)^{w(v)} \left[ t \cdot p_{S(t)} + q_t \right]_{|v^*}$$

Therefore:

$$\sum_{v \in C_t} (-1)^{w(v)} \left( t \cdot p_{S(t)} \right)_{|v} = p_{S(t)}(v^*)$$

For what concerns the second sum, any term appearing in $q_t$ contains all variables in $t$, so it is summed only once (in the top-vertex $v^*$). All the other terms of $q_t$ lack at least one of the variables in $t$, so they contribute an even number of times to the sum, half times with a positive sign and the other half with a negative one, thus globally they vanish and therefore

$$\sum_{v \in C_t} (-1)^{w(v)} q_t_{|v} = q_t^*(v^*)$$

$\square$

**Note 2** Note that $w(v)$ has been defined with the “correcting value” $k$ so that the element in the sum which corresponds to the top-vertex $v^*$ has always a positive sign. This correction is also present in the definitions of $w$ in Theorem 6.

**Example 3** Let us continue with Example 2; we already have the superpoly

$$p_{S(t)}(x_1, \ldots, x_5) = x_1 x_3^5 + x_2 x_4 + x_2 + x_3 + x_4,$$

and

$$q_t^*(x_1, \ldots, x_5) = x_2 x_2 + x_1 x_2 x_3.$$

Then we consider the evaluation of $p$ on the cube

$$C_t = \{ (0, 0, x_3, x_4, x_5), (0, 1, x_3, x_4, x_5), (1, 0, x_3, x_4, x_5), (1, 1, x_3, x_4, x_5) \}$$

in order to compute $p_t$. Thus

$$p_t = p(0, 0, x_3, x_4, x_5) - p(0, 1, x_3, x_4, x_5) +$$

$$- p(1, 0, x_3, x_4, x_5) + p(1, 1, x_3, x_4, x_5)$$

Thus

$$p_t = (x_3^5 + 1) - (x_3^2 + 1 + 1) - (x_3^5 + x_2 + x_3 + 1 + x_3) +$$

$$+ (x_3^5 + x_4 + 1 + x_4 + 1 + x_3 + x_3 +$$

$$+ x_2^2 + x_2 + 1 + 1) =$$

$$= (x_3^5 + x_4 + 1 + x_3 + x_4) + 1 + 1 =$$

$$= p_{S(t)}(v^*) + q_t^*(v^*)$$

Hereby, we consider the problem of determining all the coefficients of the sub-polynomial $p_{S(t)} + q_t^*$ when it is linear and $p$ is given as a black box function. With respect to the case in $\mathbb{F}_2$, the main difference is that whatever the monomial is, the polynomial we obtain evaluating $p$ as in Theorem 4 is always the linear part of $p_{S(t)}$, where $t_0$ is the monomial which is product of all the variables in $t$ taken with exponent 1, $t_0 = x_i_1 \cdots x_i_k$. This is due to the fact that, called

$$I = \{ i_1, \ldots, i_k \}$$

the set of the indices of the variable appearing in $t$, evaluating $p$ on the sets $X = \{ x \in \mathbb{F}_2^n : x_i = 0 \text{ if } i \notin I \text{ and } x_i \in \{ 0, 1 \} \text{ if } i \in I \}$ and

$$X_j = \{ x \in \mathbb{F}_2^n : x_i = 0 \text{ if } i \notin I \cup \{ j \}, x_i \in \{ 0, 1 \} \text{ if } i \in I \text{ and } j = 1 \}$$

we can only obtain the coefficients of the linear part of $(p_{S(t)} + q_t^*)_{|v^*}$ (which is a polynomial in $n-k$ variables, all but $x_i_1, \ldots, x_i_k$), i.e., this sum can distinguish if a variable is present in the monomial but cannot determine its degree.

We denote with $x^+$ (and respectively $x^*$) the element of $X$ (respectively of $X_j$) such that $x_i^+ = 1$ (respectively $x_i^* = 1$) for all $i \notin I$. Then, we have:

**Theorem 6** Let $t = x_{i_1}^+ \cdots x_{i_k}^+$ be a monomial, and let $I$ be the subset of indices which appear in $t$ as denoted above. If the polynomial $(p_{S(t)} + q_t^*)_{|v^*}$ is linear,

$$(p_{S(t)} + q_t^*)_{|v^*} = a_0 + a_1 x_1 + \ldots + a_n x_n,$$

then

1. $a_0 = \sum_X (-1)^{w(x)} p(x)$

where $w(x) = W_H(x^*) + W_H(x)$;

2. $a_j = -a_0 + \sum_{X_j} (-1)^{w(x)} p(x)$

where $w(x) = W_H(x^*) + W_H(x)$ and $j \notin I$.

Besides, denoting with $t_0$ the monomial $x_{i_1}^+ \cdots x_{i_k}^+$, then

$$p_{S(t)}(v^*) = (p_{S(t)} + q_t^*)_{|v^*}$$

**Proof:**

1. $\sum_X (-1)^{w(x)} p(x) =$

$$= \sum_X (-1)^{w(x)} \left[ t(x) p_{S(t)}(x) + q_t(x) \right] =$$

$$= \sum_X (-1)^{w(x)} t(x) p_{S(t)}(x) + \sum_X (-1)^{w(x)} q_t(x)$$

in the first sum the only (possibly) nonzero term is the one corresponding to $x^* \in X$, which is summed with a positive sign thus we have

$$\sum_X (-1)^{w(x)} t(x) p_{S(t)}(x) = p_{S(t)}(x^*)$$
in the second sum all the terms containing other variables than \(x_i, \ldots, x_k\) are obviously zero, and all the other terms are summed an even number of times (and vanish since they take opposite signs in pairs), except the ones which contains all (and only) variables \(x_i, \ldots, x_k\), which are summed only once (in \(x^*\)) with a positive sign:

\[
\sum_X (-1)^{w(x)} q_r(x) = q'_t(x^*).
\]

Therefore we obtain the claimed result:

\[
\sum_X (-1)^{w(x)} p(x) = p_{S(t)}(x^*) + q'_t(x^*).
\]

2.

\[
\sum_{X_j} (-1)^{w(x)} p(x) = \sum_{X_j} (-1)^{w(x)} [t(x)p_{S(t)}(x) + q_r(x)] = \sum_{X_j} (-1)^{w(x)} t(x)p_{S(t)}(x) + \sum_{X_j} (-1)^{w(x)} q_r(x) = p_{S(t)}(x^{*j}) + q'_t(x^{*j})
\]

similarly to the previous case, only coefficients of the terms which contain all the variables in \(t\) and possibly \(x_j\) are summed.

Besides, to show the last equality, it is sufficient to notice that

- \(t \cdot p_{S(t)} + q'_t = t_0 \cdot p_{S(t_0)}\)
- the sums above return only the free terms and the coefficients of the linear terms \(x_i, j \notin I\), once that all the variables \(x_i, \ldots, x_k\) are set to 1.

\[\square\]

Example 4 Let us consider the polynomial \(p \in F_3[x_1, x_2, x_3, x_4]\) then we have

\[
p = x_1^2 x_2^2 x_3 + 2x_1^2 x_2 x_4 + x_3 x_2^3 + x_1 x_2^2 + 2x_1 x_3 + 2x_2^2 x_3 + x_1 + 1
\]

and the maxterm \(t = x_1 x_2 x_3^2\), so that

\[
p_{S(t)} = x_3 + 2x_4 + x_1 \quad \text{and} \quad q'_t = x_1 x_2^2.
\]

With the sums above we obtain \(a_0 = 2, a_3 = 1\) and \(a_4 = 2\), which are the free term and the coefficients of the polynomials

\[
(p_{S(t)} + q'_t)|_{x^*} = p_{S(t_0)}|_{x^*} = x_3 + 2x_4 + 2 \in F_3[x_3, x_4].
\]

We conclude by stating the following lemma, which can be used to estimate the probability of success of the attack.

Lemma 2 Given a random matrix \(A \in M_n(F_q)\), then the probability that \(A\) is invertible is

\[
\prod_{i=1}^{n} \left(1 - \frac{1}{q^i}\right).
\]

Note that the proof of this lemma is analogous to the one of Lemma 1, and that the conclusion of Remark 1 remains valid (see Figure 2).

```plaintext
Figure 2: Graphs of the probabilities Pr[^{det(A) \neq 0}] in function of the order of the random matrix \(A\), for finite fields \(F_q\) where \(q = 2, 4, 8, 16\).
```

6 Conclusions

The cube attack has similarities with the AIDA² (Vielhaber 2007), published a year before the one by Dinur and Shamir, but which was presented as a technique directed to the analysis of the Trivium cipher. In this paper, we presented the necessary modifications of the cube attack in order to be applied to a system working on a generic finite field \(F_q\). The applicability of this version of the cube attack has to be further investigated from the point of view of implementation and it will probably share difficulties with the implementation of the original version on \(F_2\). A technique similar to the efficient brute force method recently presented in (Bouillaguet et al. 2010), once applied on cubes seems to provide better results. This technique has to be considered in the higher order setting we showed in this work.

References


URL: http://dx.doi.org/10.1201/9781420070033


² Algebraic IV Differential Attack.
Secure Two-Party Association Rule Mining

Md. Golam Kaosar  Russell Paulet  Xun Yi
School of Engineering and Science
Victoria University, Australia
Email: md.kaosar@live.vu.edu.au, russell.paulet@live.vu.edu.au, xun.yi@vu.edu.au

Abstract

Association rule mining algorithm provides a means for determining rules and patterns from a large collection of data. However, when two sites want to engage in an association rule mining, data privacy concerns are raised. These concerns include loosing a competitive edge in the market place and breaching privacy laws. Techniques that have addressed this problem are data perturbation and homomorphic encryption. Homomorphic encryption based solutions produce more accurate results than data perturbation. Most previous solutions for privacy preserving association rule mining require the disclosure of intermediate mining results such as support counts and database size to determine frequent itemset. To overcome this weakness we propose a secure comparison technique based on state-of-the-art fully homomorphic encryption scheme, by which we build secure two-party association rule mining protocol. Our solution preserves complete privacy of both parties and it is more efficient than other solutions because there is no need for exponentiation of numbers.

1 Introduction

Data mining algorithms discover patterns and interesting trends from large amount of data, which are critical to business success. Usually, the database is distributed across different sites and owned by different organizations. For example, two hospitals, each having patient records, want to discover common disease patterns from their joined data. In this case the privacy issue is raised, because privacy law does not permit the patient data to be disclosed. It is a challenge for them to perform data mining algorithms on their union data without disclosing the patient data.

The first privacy preserving data mining algorithm was introduced by Lindell and Pinkas in 2000 (Lindell & Pinkas 2000). The authors presented a protocol that produced a decision tree using the ID3, proposed by Quinlan (Quinlan 1990), whereby the entropy or information gain is computed privately. This is achieved through the use of garbled circuit of Yao (Yao 1986) and 1-out-of-2 oblivious transfer (Even, Goldreich & Lempel 1985). The main overhead in the protocol is the oblivious transfer, since this can be expensive to compute depending on the public key cryptosystem used. Our work presents a similar data mining solution, but is an association rule mining solution instead of a data classifier as in the case of Lindel and Pinkas.

Association rule mining is a data mining algorithm that provides a way to determine rules from data. The algorithms are based on calculating the support, and confidence. This is trivial when the data is in one site. Efficient algorithms (Agrawal & Srikant Sep 1994) can be used to generate association rules at user specified thresholds. However, when the data is distributed across many sites, privacy concerns are introduced. Hence we need tools to mitigate these privacy concerns.

Generally speaking, the meaning of privacy in data mining algorithms is to prevent data misuse (Clifton, Kantarcioglu & Vaidya 2002). Since there is difficult to assume a fully trusted third party, it is impossible to simply transfer all data to one site and perform the necessary mining process. Since, in most cases, it is not possible to trust people with the data because control of data distribution is very difficult once released. Therefore, tools need to be developed in order to achieve mutual beneficial result without disclosing data.

General tools for the job include either using data perturbation (Kantarcioglu & Clifton 2004), or some kind of homomorphic encryption (Pailler 1999, El Gamal 1985) scheme in order to calculate the total counts from different sites. Adding randomness to the data actually reduces the accuracy of the result, whereas homomorphic encryption allows to operate on ciphertext and those operations represented in the plaintext. Therefore, it is possible to increase the accuracy of the result by using the actual values concealed by the encryption.

Adding randomness has been shown to reduce privacy concerns (Clifton, Kantarcioglu, Vaidya, Lin & Zhu 2003, Kantarcioglu & Clifton 2004). This randomness confuses the actual data values. Hence making the process of determining the actual values difficult because the random distribution would need to be known. Adding randomness to the data actually reduces the accuracy of the outcome of the algorithm. This is true even if both parties use the same distribution, and remove it at the end.

Homomorphic properties of encryption schemes enable to perform calculations in the ciphertext, and have those operations represented in the plaintext (Rivest, Shamir & Adleman 1978, El Gamal 1985, Pailler 1999). A mixture of data perturbation and homomorphic encryption has also been considered (Ouyang & Huang 2006, Zhan, Matwin & Chang 2007). This is to both confuse the result from an adversary, and allow the adversary to still operate on the data. Up until recently, all known homomorphic encryption schemes were only partially ho-
momorphom. Partially homomorphom means that they are only homomorphome under one operation. Dijk2010 et al. (Dijk, Gentry, Halevi & Vaikuntanathan 2010) presented a scheme that was fully homomorphome on binary data. This enables the operations of data, represented as binary circuits, to be executed in its ciphertext form.

Using this result, this paper aims to provide a solution that evaluates counts without any intermediate result. This removes the issue of one party learning the other party’s input as the actual result has been masked. It is shown that this is secure based on the security assumptions of the underlying security scheme with secure parameters.

2 Background

This section encompasses the scope of the research by introducing necessary background knowledge, terms and concepts. This section will provide better understanding about the foundations of the research.

2.1 Association Rule Mining (ARM)

Association rule mining (ARM) is popular and effective way to discover correlations (known as rules) among variables in a large database. Based on interestingness ARM is capable to discover some knowledge from a huge transactional database which is apparently intelligible. As for an example an ARM may discover a rule {sugar, flour}⇒yeast from a supermarket transactional data which implies that if a customer buys sugar and flour then most probably he will buy yeast. If this correlation is revealed to the supermarket authority, they might put all these items in same aisle to increase sales.

Let us consider $I$ as the set of items in a database, where each transaction $T \subseteq I$. Any combination of items is known as itemset. That is, an itemset $I_k = \{I_1 \cup I_2 \ldots \cup I_k\}$, where $I_k \subseteq I$. An itemset with length $k$ is known as $k$-itemset. Typical form of an association rule is $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \Phi$. The support of $X \Rightarrow Y$ is the probability of a transaction in the database contains both $X$ and $Y$. On the other hand confidence of $X \Rightarrow Y$ is the probability of a transaction containing $X$ will contain $Y$ too. Usually it is the interest of the data vendor to find all association rules having support and confidence greater than or equal to a minimum threshold value. Let us illustrate the definitions of support and confidence for another instance of an association rule $AB \Rightarrow C$.

$$\text{Support}_{AB \Rightarrow C} = s = \frac{\sum \text{items} \cdot \text{SupportCount}_{AB \Rightarrow C}}{\text{DatabaseSize}}$$  \hspace{1cm} (1)

$$\text{Support}_{AB} = \frac{\sum \text{items} \cdot \text{SupportCount}_{AB}}{\text{DatabaseSize}}$$  \hspace{1cm} (2)

$$\text{Confidence}_{AB \Rightarrow C} = c = \frac{\text{Support}_{AB \Rightarrow C}}{\text{Support}_{AB}}$$  \hspace{1cm} (3)

More detail on association rule mining process is available in (Han & Kamber 2006, Tan, Steinbach & Kumar 2006).

Association rule mining process consists of two major parts (Han & Kamber 2006). First part is to find frequently large itemsets which have support and confidence values more than a threshold number of times. Second part is to construct association rules from those large itemsets.

1. Generate frequent itemset:

To determine frequent itemsets within a database, it is necessary to compare all possible itemsets. In many research works some solutions are provided to reduce this number of candidate itemsets as well as number of comparisons. Apriori algorithm is one of the leading algorithms, which determines all frequent large itemsets along with their support counts from a database efficiently. This algorithm was proposed by Agrawal (Agrawal & Srikant Sep 1994) which is discussed here in brief: Let us say $L_k$ be the frequent $k$-itemsets. Apriori algorithm finds $L_k$ from $L_{k-1}$ in two stages: joining and pruning.

- Joining: Generates a set of $k$-itemsets $C_k$ known as candidate itemsets by joining $L_{k-1}$ and other possible items in the database.
- Pruning: Any $(k-1)$-itemsets cannot be a subset of a frequent $k$-itemsets which is not frequent. Therefore it should be removed.

Apriori algorithm reduces the number of candidate itemsets and comparisons due to its two distinguishing features: (1) it traverses the itemset lattice one level at a time and (2) at each iteration new candidate itemsets are generated from the frequent itemsets of previous iteration support of which must be greater than or equal to the minimum support value. If $k_{\text{max}}$ is the maximum size of the largest frequent itemset then number of iterations in the Apriori algorithm is $k_{\text{max}} + 1$.

2. Generate association rules from frequent itemsets:

Collection of frequently large itemsets with minimum support value (defined in Equation 1) are the inputs and association rule with minimum confidence value (defined in Equation 3) are the outputs in this process. Following simple steps would generate all possible association rules with minimum support and confidence:

- For each frequently large itemsets $f$, generate subset $s$ such that $f \neq \emptyset$.
- For each $ft$, generate the association rule $ft \Rightarrow (f - ft)$ such that $\text{Confidence}_{ft \Rightarrow (f - ft)} \geq \text{minimum confidence}$

Support of $ft$ is not needed to be considered in computation, since $\text{Support}_{ft} \geq \text{Support}_f$.

2.2 Fully Homomorphic Encryption System

Homomorphic encryption is a special form of encryption where one can perform a specific algebraic operation on the plain-text by applying the same or different operation on the cipher-text. If $X$ and $Y$ are two numbers and $E$ and $D$ denotes encryption and decryption function respectively, then homomorphic encryption holds following condition for an algebraic operation such as $^{1+}$:

$$D[E(X) + E(Y)] = D[E(X + Y)]$$  \hspace{1cm} (4)

Most homomorphic encryption system such as RSA (Rivest et al. 1978), ElGamal (El Gamal 1985), Boneh (Clarkson 1994), Pailler (Paillier 1999) etc are capable to perform only one operation. But fully homomorphic encryption system can be used for many operations (such as, addition, multiplication, division etc.) at the same time.
2.2.1 Fully Homomorphic Encryption for Binary Bits

The work in (Dijk et al. 2010) proposes a new cryptosystem that provides fully homomorphic encryption over integer ciphertext. If a cryptosystem is fully homomorphic, then it has the ability to perform both addition and multiplication over the ciphertext and these operations are represented in plaintext. Hence, an untrusted party is able to operate on private or confidential data, without the ability to know what data the untrusted party is manipulating.

The fully homomorphic scheme (Dijk et al. 2010) is a simplification of an earlier work involving ideal lattices (Gentry 2009). It encrypts a single bit (in the plaintext space) to an integer (in the ciphertext space). When these integers are added and multiplied, the hidden bits are added and multiplied (modulo 2). The symmetric version of the encryption function is given by \( c = pq + 2r + m \), where \( p \) is the private key, \( q \) and \( r \) are chosen randomly, and \( m \) is the message \( m \in \{0, 1\} \). The decryption is simply \( (c \mod p) \mod 2 \), which recovers the bit. Hence, when we add or multiply by the ciphertext, the message is manipulated accordingly.

Using the symmetric version of the cryptosystem, it is possible to construct an asymmetric version. The asymmetric version is far more useful to the association rule application, since another party must be able to encrypt in order to use the homomorphic property of the cryptosystem. The following functions define the asymmetric version of the cryptosystem (Dijk et al. 2010).

**KeyGen(\( \lambda \)):** Choose a random \( n \)-bit odd integer \( p \) as the private key.

Using the private key, generate the public key as \( x_i = pq + 2r \) where \( q \) and \( r \) are chosen randomly, for \( i = 0, 1, ..., \tau \). Relabel so that \( x_0 \) is the largest

**Encrypt(\( pk, m \in \{0, 1\} \)):** Choose a random subset \( S \subseteq \{1, 2, ..., \tau \} \) and a random integer \( r \), and output \( c = (m + 2r + \sum_{i \in S} x_i) \mod x_0 \).

**Decrypt(sk, c):** Output \( m = (c \mod p) \mod 2 \)

This asymmetric version still achieves the same level of correctness as the symmetric version. The addition and multiplication of ciphertexts result in addition and multiplication being acted on the message bit. This produces the correspondence between ciphertext space and the plaintext space, as addition in the ciphertext space reduces to exclusive OR (\( \oplus \)) in the plaintext space and multiplication in the ciphertext space reduces to AND (\( \land \)). This correspondence (homomorphism) between these two operations, addition and multiplication, are shown in Equations 5 and 6, respectively.

\[
E(m_1) + E(m_2) = E(m_1 \oplus m_2) \tag{5}
\]

\[
E(m_1) \cdot E(m_2) = E(m_1 \land m_2) \tag{6}
\]

Hence, from this correspondence, it is possible to construct very complicated binary circuits to evaluate on the data, without exposing the actual data. More details regarding implementation can be found in the original paper (Dijk et al. 2010).

2.2.2 Fully Homomorphic Encryption for Integers

The association rule mining algorithms does not operate on binary data, however. They operates on values in the integer space. Hence we need to extend the underlying cryptosystem to accommodate integer numbers. This is achieved by representing the integer as a binary vector and encrypting each bit.

Using this format it is possible to encrypt two integers and apply binary AND and XOR to each respective encrypted binary value, with the consequence of the homomorphic property of the encryption scheme. Let us consider two integers \( u \) and \( l \), which can be represented as binary numbers \( u = [u_{n-1}, u_{n-2}, \ldots, u_1, \ldots, u_0] \) and \( l = [l_{n-1}, l_{n-2}, \ldots, l_1, \ldots, l_0] \) respectively. This is illustrated in Figure 1, where \( c_i \) refers to the carry bit.

![Figure 1: Displays the carry bit operation](https://via.placeholder.com/150)

Computer architecture implements the carry bit to perform regular addition (Brookshear 2005) and therefore the homomorphic encryption must accommodate this requirement. This is trivial since the carry bit can be calculated with the following expression in Equation 8 using the homomorphic property of the cryptosystem, starting with the least significant bit.

\[
c_i = (l_i \land u_i) \lor (l_i \oplus u_i) \land c_{i-1} \tag{8}
\]

where \( l_i \) and \( u_i \) refer to the lower and upper binary vector respectively, \( \land \) and \( \lor \) represent AND and OR respectively, and \( c_i \) representing the carry bit. The carry bit is initially zero \( c_0 = 0 \). In this expression the binary OR \( \lor \) is represented by \( (p \lor q) \oplus (p \cdot q) \) where \( \lor \) and \( \cdot \) refer to XOR and AND operations respectively. This substitute is needed because binary OR is not directly available as part of the homomorphism of the cryptosystem. The output bit for that position in the binary vector is calculated using Equation 9.

\[
o_i = l_i \lor u_i \lor c_{i-1} \tag{9}
\]

Using Equation 9, it is possible to add any two \( n \)-bit integers represented as binary vectors. This has
demonstrated how to perform addition on integers. For completeness, multiplication must be also implemented. Multiplication can be simply performed in terms of multiple addition, which avoids the need for complicated binary circuits, including bit shifting. This simple scheme can be extended to provide more functionality, such that we are able to discover association rules in a privacy preserving setting.

2.2.3 Homomorphic Function Abstraction

Using this abstraction for integers, it is now possible to define functions in terms of n-bit integers. The encryption and decryption functions for (abstracted) integers are as follows.

- **E<sub>pk</sub>(i):** Encrypts a n-bit integer i using the public key pk, returning an encrypted n-bit integer c as ciphertext.

- **D<sub>sk</sub>(c):** Decrypts a n-bit integer c using the private key sk, returning a plaintext n-bit integer i

The purpose of these functions is to convert an integer between the plaintext and ciphertext. Due to the abstraction of binary bits into integers, it will assist the creation of a higher level protocol. The functions for such a protocol are defined next.

- **Homomorphic Binary AND Operation:**
  \[ \text{HomAND}(x, y) = (x \land y) \]
  Receives two encrypted n-bit integers x and y, and returns a third encrypted n-bit integer z. The output is calculated bit-by-bit using the homomorphic property, that is \( z_i = x_i \land y_i \), where AND is evaluated using Equation 6, for \( i = j \).

- **Homomorphic Binary XOR:**
  \[ \text{HomXOR}(x, y) = (x \lor y) \]
  Receives two encrypted n-bit integers x and y, and returns a third encrypted n-bit integer z. The output is calculated bit-by-bit using the exclusive OR property of the homomorphic encryption, that is \( z_i = x_i \lor y_i \), where XOR is evaluated using Equation 5, for \( i = j \).

- **Homomorphic Addition:**
  \[ \text{HomAdd}(x, y) = (x + y) \]
  Receives two encrypted n-bit integers x and y and returns a third encrypted n-bit integer z. Where \( z_i \) is calculated using Equations 9 and 8 for the current column and carry bit calculation respectively. Figure 1 illustrates this double calculation of carry bit and column bit for each bit of the integer.

3 Proposed Solution

This section describes motivations for our proposed solution along with the model definition, necessary function development and finally, the proposed algorithm.

3.1 Motivation and Model Definition

Let us consider two data sites Alice(A) and Bob(B) possess two horizontally partitioned transactional database DB<sub>1</sub> and DB<sub>2</sub> of size |DB<sub>1</sub>| and |DB<sub>2</sub>| respectively, where, combined database DB = {DB<sub>1</sub> \( \cup \) DB<sub>2</sub>}. Let us, also assume \( I = \{i_1, i_2, \ldots, i_n\} \) is the set of items where each transaction \( T \subseteq I \). Therefore, any items to be frequently large, its support must be greater than or equal to the minimum support threshold denoted by s. Similarly for an association rule to be selected, it’s confidence must be greater than or equal to a minimum confidence threshold denoted by c.

To highlight the significance of our proposed solution; let us consider some simple ARM algorithm steps necessary to determine, whether an itemset with counts \( c_1 \) and \( c_2 \) in A and B respectively are frequent or not:

- **Step 1:** Data site A sends count \( c_1 \) and |DB<sub>1</sub>| to other party B.
- **Step 2:** Data site B sends count \( c_2 \) and |DB<sub>2</sub>| to other party A.
- **Step 3:** Both A and B can compute whether \( |DB_1| + |DB_2| \geq s \). If true then the itemset is frequent.

Most privacy preserving solutions either for two parties or for multiple parties have following weaknesses:

- **Privacy of individual itemset counts \( c_1 \) and \( c_2 \) are preserved, but their summation \( (c_1 + c_2) \) is disclosed. This should not be considered as fully privacy preservation. Sometimes this would help the adversary guess some counts. In fact for two party: the privacy is not preserved at all.
- **Privacy of individual database sizes |DB<sub>1</sub>| and |DB<sub>2</sub>| are preserved, but their summation \(|DB_1| + |DB_2|\) is disclosed. This again is considered as violation of privacy.
- **During the generation of association rules from frequently large itemsets, the total count of itemsets are disclosed too.
- **Mining result provides association rules along with their support and confidence values (e.g. \( AB \Rightarrow C \), \( s = .32 \) and \( c = .25 \)). This discloses the privacy too.

This paper proposes a solution which would preserve the privacy in all these cases. No such intermediate results would be disclosed to any party as opposed to most privacy preserving ARM solutions.

Let us, consider Alice(A) has public and secret keys pk and sk respectively generated according to the key generation process mentioned in Section 2.2. Bob(B) performs cryptographic operations using pk, but the cannot decrypt the result since he does not know sk. Data encryption and decryption is performed according to the fully homomorphic encryption system discussed in Section 2.2. These parties communicate through a private channel which is protected by a standard secret key cryptosystem, such as DES (FIPS-PUB.46 1977) or AES (FIPS-PUB.197 2001). It is also assumed, A and B are semi-honest which implies they follow the protocol but they are allowed to record intermediate computation for future use to break privacy of other entity.

3.2 Secure Comparison of Two Integers

This section proposes a solution to compare two numbers privately. Let us, consider two n bit long integer numbers M and N. The proposed technique compares M and N and determines whether M is equal or less than or greater than N without revealing the value of M or N themselves.

Let us, consider the first version of the algorithm (Algorithm 1) which performs the comparison without preserving the privacy (This basic technique can
be found in many computer architecture books such as (Harris & David 2007)):

Algorithm 1 Comparison of two integers (M and N) without privacy concern

input : integers M, N
output : (One bit output. If output = 0 then M ≥ N otherwise M < N.)

Begin

Y ← M + N + 1 /* Subtraction of M and N gives the clue about their relative size. Two’s complement of a number is equivalent to the negative of the same number. Therefore, Y = M − N.* /
R ← Y AND 2n−1
return MSB(R) /* returns the most significant bit (MSB) of R. This is actually is the sign bit of the subtracted result */
End

With the consideration of fully homomorphic functions -\textit{HomXOR}, \textit{HomAdd} and \textit{HomAND} derived in Section 2.2; this proposed solution would do the same comparison as in Algorithm 1 with preserving privacy. Let us say Alice and Bob have their secret numbers encrypted \( \alpha \leftarrow E_{pk}(M) \) and \( \beta \leftarrow E_{pk}(N) \) respectively. Secure comparison between \( \alpha \) and \( \beta \) is proposed in Algorithm 2.

Algorithm 2 Secure comparison of two encrypted integers (M and N)

input : ciphertexts \( \alpha, \beta \)
output : ciphertext \( R \) (One bit encrypted output. If \( R = D_{sk}(R) = 0 \) then \( M \geq N \) otherwise \( M < N \.)

Begin

\( \overline{\beta} \leftarrow \text{HomXOR}(\beta, E_{pk}((2^{n} - 1)) \) /* Binary negation of \( \beta \)*
\( Y' \leftarrow \text{HomAdd}(\alpha, \overline{\beta}) \) /* Homomorphic addition of \( \alpha \) and \( \overline{\beta} \)*
\( Y' \leftarrow \text{HomAdd}(Y', E_{pk}(1)) \)
\( R' \leftarrow \text{HomAND}(Y', E_{pk}(2^{n} - 1)) \) /* The result is encrypted and only Alice can decrypt that */
return MSB(R') /* returns the sign bit or the Most Significant Bit (MSB) of \( R' \).*/
End

In summary; a function can be defined -\textit{HomComparison}(\( \alpha, \beta, \gamma \)) which determines whether \( \frac{\alpha}{\beta} \geq \gamma \) is true or not. If \( R = D_{sk}(R) = 0 \) then \( \frac{\alpha}{\beta} \geq \gamma \), else \( \frac{\alpha}{\beta} < \gamma \)

3.3 Secure Comparison of Fraction Numbers

In the comparison of support and confidence of an itemset and an association rule respectively, it is necessary to compare two fractional numbers. The general form of both cases is to determine whether \( \frac{\alpha}{\beta} \geq \gamma \) is true, where \( \alpha \) and \( \beta \) are two integers and \( \gamma \) is a fractional number. This section presents a way to perform the comparison without performing homomorphic division operation. \( \frac{\alpha}{\beta} \geq \gamma \) can be simplified as follows:

\[ \frac{\alpha}{\beta} \geq \gamma \equiv \alpha \geq \gamma \times \beta \equiv \alpha \times 10^n \geq \gamma \times \beta \times 10^n \equiv \alpha \times 10^n \geq \varepsilon \times 10^n \]

Where, \( n \) = number of digits in \( \gamma \) after decimal points and \( \varepsilon = \gamma \times 10^n \)

We propose Algorithm 3 to perform above mentioned comparison securely where \( \alpha \) and \( \beta \) are encrypted. Let us say \( \alpha t = E_{pk}(\alpha) \) and \( \beta t = E_{pk}(\beta) \)

Algorithm 3 Secure comparison of two fractional numbers

input : ciphertext \( \alpha t, \beta t \), threshold \( \gamma \)
output : \( Rt \) (One bit encrypted output. If \( R = D_{sk}(R) = 0 \) then \( \frac{\alpha t}{\beta t} \geq \gamma \), otherwise \( \frac{\alpha t}{\beta t} < \gamma \))

Begin

\( \eta \leftarrow \) number of digits in \( \gamma \) after decimal
\( \varepsilon \leftarrow \gamma \times 10^n \)
\( \text{TempAlpha} \leftarrow 0 \)
for \( i = 1 \) to \( 10^n \) do
\( \text{TempAlpha} \leftarrow \text{HomAdd}(\text{TempAlpha}, \alpha t) \) /* iterative addition to avoid multiplication */
end for
\( \text{TempBeta} \leftarrow 0 \)
for \( i = 1 \) to \( \varepsilon \) do
\( \text{TempBeta} \leftarrow \text{HomAdd}(\text{TempBeta}, \beta t) \) /* iterative addition to avoid multiplication */
end for
\( R t \leftarrow \text{HomComparison}(\text{TempAlpha}, \text{TempBeta}) \)
return MSB(Rt) /* returns the sign bit or the most significant bit (MSB) of \( R t \).*/
End

In summary; a function can be defined -\textit{HomFractionComparison}(\( \alpha, \beta, \gamma \)) which determines whether \( \frac{\alpha}{\beta} \geq \gamma \) is true or not. If \( R = D_{sk}(R) = 0 \) then \( \frac{\alpha}{\beta} \geq \gamma \), else \( \frac{\alpha}{\beta} < \gamma \)

3.4 Proposed Two Party ARM

Let us, say among the two parties (Alice(\( A \)) and Bob(\( B \)), \( A \) has public and secret keys \( pk \) and \( sk \) respectively.\( E_{pk} \) and \( E_{sk} \) denotes fully homomorphic encryption and decryption respectively discussed in Section 2.2. Let us consider following operations are performed initially:

\( A : \)
\( D_1 \leftarrow E_{pk}(|DB_1|) \)
\( \text{SendToB}(D_1) \)

\( B : \)
\( D_2 \leftarrow E_{pk}(|DB_2|) \)
\( pt \leftarrow \text{HomAdd}(D_1, D_2) \)

Let us consider for \( k \)th iteration in the Apriori algorithm where, \( L_k \) and \( C_k \) represents large \( k \)-itemset and candidate \( k \)-itemset respectively. Following algorithm illustrates necessary steps in both sides (\( A \) and \( B \)) to generate frequently large itemsets:
Algorithm 4 Large itemset generation between A and B

input of A : \( L_1 \) with counts, \( \pi \)
input of B : \( L_2 \) with counts, \( \pi \), s
output : \( L_{k+1} \)

Begin
Both Alice(A) and Bob(B)

\( C_{k+1} \leftarrow \text{GenerateCandidate}(L_k) \)
for all \( \rho \in C_{k+1} \) do

Alice(A)

\( \omega_1 \leftarrow \text{count}(\rho) \)
\( \omega_1 \leftarrow E_{pk}(\omega_1) \)
SendToB(\( \rho,\omega_1 \)) /* transmits itemset and its encrypted count to B*/

Bob(B)

\( \text{FrequencyDBofA} \leftarrow \{ \text{FrequencyDBofA} \cup (\rho,\omega_1) \} \) /* B stores all encrypted counts of itemsets sent by A.

These information would be necessary in itemset generation in Algorithm 5.*/

\( \omega_2 \leftarrow \text{count}(\rho) \)
\( \omega_2 \leftarrow E_{pk}(\omega_2) \)
\( \omega \leftarrow \text{HomAdd}(\omega_1,\omega_2) \)
\( R_t \leftarrow \text{HomFractionComparison}(\omega,\pi,s) \)
SendToA(\( R_t \))

Alice(A)

\( R \leftarrow D_{sk}(R_t) \) /* result is only decrypted by Alice*/
if \( R = 0 \) then

\( L_{k+1} \leftarrow \{ L_{k+1} \cup \rho \} \) /* itemset satisfies minimum support requirement*/
SendToB(\( \rho \))

end if

Bob(B)

\( L_{k+1} \leftarrow \{ L_{k+1} \cup \rho \} \)
end for
End

Repeated use of above mentioned Algorithm 4 generates all frequently large global itemsets \( L_g = \{ L_1, L_2, \ldots, L_m \} \), where \( m = \) maximum number of items present in any large itemsets.

Figure 2 illustrates the flow diagram of Algorithm 4 with the assumption that counts of an itemset in A is \( c_1 \) and in B is \( c_2 \).

![Algorithmic steps for itemset generation](chart)

Figure 2: Algorithmic steps for itemset generation

Now, all association rules with minimum confidence \( c \) are to be generated from \( L_g \) preserving privacy. If Equations 1, 2 and 3 are combined together, following simplification may be achieved:

\[
\text{Confidence}_{AB} = c = \frac{\text{Support}_{AB} \cdot c}{\text{Support}_{A}}
\]

Following algorithm generates association rules from all frequently large itemsets in \( L_g \).

Algorithm 5 Association rule generation

input of B : \( L_{\text{all}},c \), \( \text{FrequencyDBofA} \)
output : \( AR \) (Set of all association rules)

Begin
Both Alice(A) and Bob(B)

for all \( \rho \in L_{k+1} \) do

Bob(B)

Split \( L_t \) into all possible \( t_1 \) and \( t_2 \) such that, \( L_t = \{ t_1 \cup t_2 \} \) and \( \{ t_1 \cup t_2 \} = \phi \) /* to generate all possible combinations of association rules from \( L_t \)*/

\( \alpha_1 \leftarrow E_{pk}(\text{count}(t_1)) \) /* Local count in B*/
Encrypted count of \( t_1 \) from \( \text{FrequencyDBofA} \)
Encrypted counts of \( A \) stored in \( B \) during itemset generation stage/*

\( \alpha \leftarrow \text{HomAdd}(\alpha_1, \alpha_2) \)

\( \beta_1 \leftarrow E_{pk}(\text{count}(t_2)) \) /* Local count in B*/
Encrypted count of \( t_2 \) from \( \text{FrequencyDBofA} \)

\( \beta \leftarrow \text{HomAdd}(\beta_1, \beta_2) \)

\( R_t \leftarrow \text{HomFractionComparison}(\alpha, \beta, c) \) /* Tests whether satisfy minimum confidence requirement*/
SendToA(\( R_t \))

Alice(A)

\( R = D_{sk}(R_t) \)
if \( R = 0 \) then

\( AR \leftarrow \{ AR \cup t_1 \rightarrow t_2 \} \) /* The association rule satisfies conditions and added to the final output*/
SendToB(\( t_1 \rightarrow t_2 \))

end if

Bob(B)

\( AR \leftarrow \{ AR \cup t_1 \rightarrow t_2 \} \)
end for
End

4 Analysis

Since the protocol is created under a semi-honest model, it is assumed that both parties cannot deviate from protocol. This means the only way to circumvent the security of the protocol is if one party has the ability to decrypt the ciphertext. It is also assumed that the parameters of the cryptographic scheme are chosen from secure ranges (Dijk et al. 2010). The analysis will consider the security from both Alice’s and Bob’s point of view.

The security of Alice’s data is satisfied if Bob can determine her value. Since her value is encrypted as ciphertext using the fully homomorphic encryption scheme, Bob is unable to determine her value unless he can decrypt the ciphertext. This would mean that Bob would have to break the approximate-gcd problem (Dijk et al. 2010). That is, given ciphertext determine the private key. Since the encryption scheme
is a probabilistic cipher, it provides semantic security. Fundamentally, this means there are many ciphertexts to the same plaintext, this makes the encryption of the actual number indistinguishable from the encryption of a random number.

Conversely, the security of Bob's data is satisfied if Alice cannot determine his value. Since Bob is calculating the sign bit of the difference homomorphically and transmitting that bit, Alice does not have access to any number besides the encrypted sign bit, which she can decrypt. Therefore, Bob's security is satisfied.

Taken together, both Alice and Bob cannot determine each other's value, while still being able to perform meaningful calculations to produce accurate ARM rules.

5 Conclusion

This paper has presented a 2-party association rule mining algorithm using fully homomorphic encryption (Dijk et al. 2010). The protocol was shown to be secure under the semi-honest model of multi-party computation. This security is based on the approximate-gcd problem associated with the fully homomorphic encryption system. The main contribution was to calculate the support and confidence of association rules homomorphically, and returning a single bit.

This greatly improves the security over previous homomorphic public key cryptosystems. Classical cryptosystems like the RSA and El Gamal are only partially homomorphic, and after a calculation has been performed homomorphically, one party has to decrypt the ciphertext to obtain a meaningful result. In the proposed solution, however, the actual data value is reduced to a single bit.

Future work regarding privacy preserving data mining would include; improving the efficiency of by removing unnecessary communication; expanding on the number of parties to a multi-party computation interaction; and applying the fully homomorphic encryption system to other data mining algorithms. Further work is also required to improve both the efficiency and security of the underlying cryptosystem.

References


Tan, P. N., Steinbach, M. & Kumar, V. (2006), Introduction to Data Mining, Pearson Education, Inc.

*Journal of Network and Computer Applications* 30(3), 1216 – 1227.
Detection of Anomalies from User Profiles Generated from System Logs

Malcolm Corney  George Mohay  Andrew Clark
Information Security Institute
Queensland University of Technology
PO Box 2434, Brisbane QLD 4001, Australia
m.corney@qut.edu.au, g.mohay@qut.edu.au, a.clark@qut.edu.au

Abstract
We describe research into the identification of anomalous events and event patterns as manifested in computer system logs. Prototype software has been developed with a capability that identifies anomalous events based on usage patterns or user profiles, and alerts administrators when such events are identified. To reduce the number of false positive alerts we have investigated the use of different user profile training techniques and introduce the use of abstractions to group together applications which are related. Our results suggest that the number of false alerts that are generated is significantly reduced when a growing time window is used for user profile training and when abstraction into groups of applications is used.

Keywords: User profiling, insider misuse, abstraction.

1 Introduction
Computer crime continues to be problematic for both public and private sectors not only in Australia but at an international level. Over 50% of respondents to the 2006 Computer Security Institute/FBI Computer Crime and Security Survey (Gordon, Loeb et al. 2006) reported unauthorized use of computer systems. In an equivalent Australian survey, the 2006 Australian Computer Crime and Security Survey (AusCERT 2006), 22% of respondents reported experiencing one or more electronic attacks.

The field of computer forensics has been rapidly expanding in the past twenty years in an effort to combat the continuing increase in the incidence of criminal activity involving computers. This field is normally defined around the identification, securing and analysis of evidence for eventual presentation in a court of law. Few cases result in a criminal prosecution and a broader definition of computer forensics can be made that simply attempts to detect, secure and analyse evidence from computer systems. This may be done by an organization, for instance, in response to a security incident, internal or external.

The surveys highlight that the most common types of criminal activity are the results of virus, worm or Trojan infections. Insider abuse of Internet access, email or computer system resources, however, is the third most common type of misuse in the United States of America (Gordon, Loeb et al. 2006) and the second most common type of misuse in Australia (AusCERT 2006).

Insider misuse can be defined as the performance of activities where computers and networks in an organization are deliberately misused by those who are authorized to use them. Some activities which can be categorized as insider misuse include:
- unauthorized access to information which is an abuse of privileges
- unauthorized use of software or applications for purposes other than carrying out one’s duties
- theft or breach of proprietary or confidential information
- theft or unauthorized use of staff or customer’s access credentials
- computer facilitated financial fraud

This paper reports on work aimed at detecting anomalous events that may be indicators of insider misuse. More specifically we attempt to detect unauthorized use of software applications by users from within an organization. Our general approach has been to build user profiles from computer security audit logs which record the applications used. In particular, we have used the security audit log from computers running the Windows XP operating system for this work.

While we have based this research on a specific operating system, the approach may be generalised to any operating system or computer system, such as an ERP system, which records user’s activities as events.

We have created user profiles from data recorded in the Windows Security log by identifying the processes or applications run by a computer user. The events recorded in the Windows security audit log can be correlated to determine when a process was started and terminated by the user or the system. Data from the correlated events can be stored and queried for post hoc investigation of a user’s activities on the computer. User profiles include information on which times of the day or week the various applications were used by an individual and also record the first time an application was used by the user.

Users in an organization may have specific duties which they carry out on a routine basis at various times of the working day or week. A sudden departure from routine may be an indicator that the user is not carrying...
out their routine duties. Use of applications outside of regular working hours may also be an indicator of misuse of the computer system.

The first use of an application may be an indicator of a user installing software which is outside of an organization’s Standard Operating Environment (SOE) if they have the privileges to do so. It may also simply be the user using an application from the SOE for the first time.

We define an alert for this work as the result of detecting a user instigated event which is atypical for that user’s profile. Alerts are based on the situations mentioned above, i.e. first use of an application and usage of an application at times outside the norm for the user.

When the use of new applications or use of applications outside of a user’s typical profile is detected, alerts can be generated but many of these are likely to be false positives. Any approach which aims to detect anomalous usage must also concentrate on reduction in the number of alerts to have any benefit to the organization.

It is likely that a user’s profile will not remain static but will vary as the user’s duties change or the organization’s SOE changes. It is therefore a requirement of such a system which detects anomalous usage that it be flexible or dynamic in its generation of user profiles. In this paper we present two options for the generation of dynamic user profiles.

The first use of an application may be an indicator of a user installing software which is outside of an organization’s SOE if they have the privileges to do so. We worked with data from the Windows security audit log from computers running the Windows XP operating system. We defined a user instigated event which is atypical for that user’s profile as any activity out of the ordinary for a user’s normal or habitual use of applications. The software and have used it to do so. We worked with data from the Windows security audit log from computers running the Windows XP operating system. When various audit controls are enabled, these logs record information about user log on sessions and the applications or processes invoked by the users of the computers and by the computer system itself.

2 System Design – User Profiling and Event Abstraction

Our aim has been to develop prototype software that implements a capability to identify events that are anomalous and may be indicative of computer misuse within an organization. We have in addition collected data for evaluating the effectiveness and performance of the software and have used it to do so. We worked with data from the Windows security audit log from computers running the Windows XP operating system. When various audit controls are enabled, these logs record information about user log on sessions and the applications or processes invoked by the users of the computers and by the computer system itself.

2.1 Summary of the Design

Our approach consists of six main stages which are summarized below. Further detail of these steps is provided in Section 3 of this paper.

- Data Collection and Preparation – Windows Security log data is collected using a VB script run daily to collect, compress and clear the log file. Further conversion steps are applied to the logs before any data reduction steps are undertaken as the Windows Security log is stored in a proprietary binary format.

- Data Reduction and Correlation – As the amount of data collected from the log sources is voluminous, only events which are recorded as a direct result of user action, such as logging in, logging out and starting and stopping applications, are used for further investigations. The approach taken for correlation is similar to that used by Abbott, Bell et al. (2006) where event abstraction is used to recognise logical events from the raw events recorded in the logs.

- Data Storage – For further processing, including preparation of user profiles and alert generation, the data is persistently stored in a relational database.

- Profiling – User profiles are generated so that users’ normal or habitual use of applications can be determined.

- Alert generation – Simple alert types are defined based on the data recorded in the user profiles, and usage data for users from ensuing time periods are used for comparison. When abnormal events are detected, alerts are generated.

- Alert checking – When alerts are generated it is necessary for them to be checked to determine if they are benign or if there is some real threat behind the cause of the event.

2.2 Profile Generation and Training

Before a profile can be generated for a particular user we must have that person’s usage data for a specific continuous period of time. The training period should be selected so that most of the routine activities a user
performs are included. This will likely be different for different users. For this paper we determine this time period empirically.

We suggest three possible approaches for generating the user profile. The first approach is to use a static or constant window user profile. With this approach detection of alerts is carried out on a weekly basis after the training period and the user profile remains the same for each week of testing. Alerts are always generated based on that initial user profile. This approach is likely to generate many alerts as the person’s usage changes due to their role changing within their organization or when software updates are applied. We concede that this is unlikely to be a successful approach but it provides a baseline for comparison with other approaches.

A second approach is to use a growing window, where the profile training time period is continually extended by adding events to the user profile from the testing period after alerts have been generated for the week under test. It is necessary for feedback to be given about whether or not the event causing an alert is benign before it can be added to the training data. With this approach, the user profile becomes dynamic and captures changes in a user’s behaviour or in the user’s environment due to software updates. A possible problem with this approach is that the user profile may retain too much stale history especially if a user’s role changes over time.

A third approach is to use a sliding time window for the user profile where the width of the time window remains constant. After training the user profile and generating alerts based on the data from the testing week, the user profile is recalculated by removing the oldest week of profile data and adding the new week of data that has had any events causing false alerts. This approach is dynamic in nature and does not allow the user profile to become cluttered with too much historical data.

2.3 Event Abstraction

The previous section discussed how we created user profiles. The creation of the profiles is based on the usage of specific processes, that is, specific versions of applications, each of which has its own path and executable name. A limitation with this approach is the high number of alerts that may be generated for a user. A user’s activities are likely to change over time as that person’s duties or roles change within their organization and the software tools and applications which a person uses is not a stable set. In addition, applications in a user’s profile will change as the applications in use are upgraded or changed by the organization.

Some questions which need to be considered then include:

1) Are executables and applications in an organization’s SOE similar enough in nature to be grouped together in some way?

2) Should different versions of the same application be grouped together in a user’s profile?

For example, a person in a clerical role in an organization would be likely to use tools from an office suite like Microsoft Office, including Word, Excel, PowerPoint and Outlook, and a software developer might use an Integrated Development Environment like Eclipse, and may require the Java Standard Development Kit, an SQL Database server, and a repository tool like Subversion. Each of these applications is likely to be upgraded to the latest versions by the organization as they become available. Alternatively, the organization may change its software procurement policies, meaning that completely different suites of applications may be used.

We propose two types of groupings or abstractions for applications, which we have termed process families and process groups. We compare the use of process families and process groups with individual processes for profile and alert generation.

The first grouping type, process families, is based on families of applications or processes that are used for similar purposes, as discussed above, e.g. software development or office administration. All applications and by extension, the process names and their paths that are recorded in the security audit logs, can be assigned to a process family. Further details on how this was implemented are given in Section 4.

The second type of grouping, we have named process groups. Process groups are constructed from clusters of applications or processes which are grouped on their path and their executable name. Using this approach processes with the same value for their path or processes with the same value for their executable name will be grouped together. This allows all applications that are run from the same directory along with all version updates to belong to the same process group.

2.4 Alert Checking

Alerts that are generated in a user profiling system would most likely require human processing, although if an organization’s security policy specified a Standard Operating Environment, alerts could first be automatically checked against that.

It is desirable in any system that is checking employee’s activities that there be as few false alerts as possible, so that system administrators are not wasting time or becoming complacent because they are checking numerous alerts.

In organizations which have a SOE there will be some users who are granted higher level privileges, including the right to install software which is not included in that SOE. In these situations, the user profile is important for reducing the number of alerts that may be generated. If a user profile is recorded for a person, software which is not part of the SOE would only have to be checked the first time an alert is generated. It could be marked as normal or acceptable use and become part of that person’s user profile. Once it is part of the user profile, further usage of that software would not cause any alerts to be generated. If the organization did not perform user profiling, and relied on checking all application usage against the SOE, every usage of software not in the SOE would cause an alert to be generated.

3 Implementation and Experimental Methodology

The following sections describe how we implemented the system and the experiments we undertook to find the best set of parameters.
3.1 Data Collection and Preparation

The Windows Security logs collected and examined during the course of the project were from desktop computers running Windows XP Professional Edition with Service Pack 3 installed. Local event logging was enabled and all available auditing options were set. Further to this, auditing of file accesses was enabled where possible at the root level of the logical disks on the computers being logged.

Under normal usage during office hours, with all auditing options enabled, over 1 million events were generated daily. Even though the Security log is meant to be able to be configured to grow in size to 4.5 GB, the maximum log size achieved on the systems under study was approximately 10% of this at 450 MB. The auditing of Object Access with all accesses to all files on the workstation being audited lead to the rapid growth of log files and was turned off after one month of data collection to conserve processing time and storage space.

A VB script was prepared which saved and emptied the log and compressed it to preserve local disk storage. ELDump (Lauritsen 1998) was used to convert the binary event file to a text version for further processing and analysis. Data was collected for a period of nine consecutive months on one of the Windows computers and eighteen months on the other. Table 1 displays the number of login sessions and the number of applications started for each user on the computer systems being surveyed.

<table>
<thead>
<tr>
<th>Computer Name</th>
<th>Days of Data Collection</th>
<th>Number of Login Sessions</th>
<th>Number of Processes Started</th>
</tr>
</thead>
<tbody>
<tr>
<td>GANDALF</td>
<td>297</td>
<td>348</td>
<td>27,821</td>
</tr>
<tr>
<td>ARAGORN</td>
<td>531</td>
<td>124</td>
<td>953,429</td>
</tr>
</tbody>
</table>

Table 1: Information about logged data for the computers studied

3.2 Data Reduction and Correlation

A great deal of information useful for extraction of a user’s activities is recorded in the Windows Security log. All events in the Windows Security log contain some common data, such as the date and time stamp, computer name, domain name, user name, event type and identifier numbers, and further information specific to each type of event.

For user profiling, the most useful event types include log on, log off, process start and process exited events. These provide details of a user’s log-in sessions and interactions with the applications and services installed on the computer.

It is necessary to correlate log on and log off events to determine the duration of a user’s log in session. This is possible in the Windows Security log by matching the value in the Session Identifier field of the event. Similarly, the process start and process exited events from a specific user and log in session can be matched to determine the duration of a user’s application usage. This can be done by matching the Session Identifier, Process Identifier and Process Name for the relevant events.

When the matching process exited event occurs, the duration for which the process ran can be recorded.

When full auditing is enabled in the Windows Security log, Object (file handles, network resources etc.) accesses are recorded in the log and these are recorded with the specific process name and process identifier which accessed the object. Many hundreds of object access events are recorded while an application is being used. For this work, the object access events were neglected due to the complexity of data recorded in them.

3.3 Data Storage

Relational database tables were used to store information relating to users and their activities on the computers studied. Information from the Windows Security logs was stored for each user of each computer. This entailed an entry for each log-in session including the start and end times of the session and the total number of applications invoked by the user during that session. Entries were also stored for each application including the name and full path of the application, start and end times of the application, and details of the parent process responsible for invoking the application. In many cases, the parent process is Windows\explorer.exe as this is the desktop application that the user interacts with when using the Windows XP operating system. For example, when the Eclipse IDE application is started from the desktop, Windows\explorer.exe is recorded as the parent process of the Eclipse\eclipse.exe process. Many processes, however, are spawned by an application a user may have started. For example, when a user starts a Java application from within the Eclipse IDE, the Java\bin\javaw.exe process is invoked to run the Java application but it is invoked by the Eclipse\eclipse.exe process. This information could allow complex hierarchies of process usage to be determined. The hierarchical nature of the processes was not analysed for inclusion in this paper.

3.4 User Profiles

To generate user profiles we collected data on the usage of which applications a person started. In particular, we chose to record the following attributes which were used for the generation of alerts:

1. The hour of the day an application was started;
2. The day of the week an application was started; and
3. Whether or not the application had been run by the user previously

A time period for user profile construction also had to be considered. Figure 1 shows the cumulative number of new processes used by different users for the data we collected. There are a large number of processes recorded for the first time, for a user in the first eight weeks or so of the logs and then the rate of new processes recorded tapers off slightly.

We tested each of the user profile types introduced in Section 2.2, namely constant time window, growing time window and sliding time window. For these experiments, we tested each combination of profile type and process
abstraction type with 14, 28 and 56 day initial time periods for the user profile.

To ensure that all experiments could be compared with each other, the end date for the training period of the initial time window was set as a constant. All alerts generated in successive weeks were based on the same set of usage data. This means that the starting date of the initial time window for the 56 day training period precedes the starting date of the 14 day training period by 42 days.

![Figure 1: Cumulative Count of New Processes Used by a User](image)

### 3.5 Alert Generation

Once the different profiles were created, alerts were calculated for a seven day testing period directly following the profile training period. The total number of alerts for the week for each of the attributes measured, i.e. hour of day that a process was started, day of week that a process was started, and if a process was started for the first time, were recorded.

For the purposes of comparison of different approaches in our system, all events causing alerts have been considered to be benign or in other words, false positives. After the testing was carried out on each week’s data, the dynamic user profile types were regenerated to include all events from the week’s data that was just tested. The results reported in Section 4 for each experiment are the cumulative number of alerts for a period of thirty weeks of alerting for each of the two users for which we had data.

### 3.6 Abstraction

As discussed in Section 2.3, we have proposed two abstractions for collections of processes for reduction of the number of alerts generated by our system. Our experiments have also recorded the results when the abstractions were not used.

The first of the abstractions we proposed was process families. For the security audit logs collected from the computers under study, nine process families were created from 2,204 differently named processes. The nine process families were: operating system tools, office applications, games, security tools, browsers, development tools, servers, utility applications and installers. This was a laborious and manual task, although some of the processing could have been automated, e.g. all applications in a particular directory hive could have been labelled as part of the same family. In a large organization, such an approach without automation would be prohibitively expensive to deploy.

The second abstraction proposed was process groups. These groups were generated programmatically using a clustering approach. For each distinct process name recorded in the security audit logs, the full process path was extracted and split into a path name and a process name. For example, the process `C:\Program Files\Java\jdk1.6.0_17\ bin\java.exe` has a path value of `C:\Program Files\Java\jdk1.6.0_17\bin` and a process name value of `java.exe`. All applications with the same path as the `java.exe` executable are considered part of the process group, e.g. `javaw.exe` and `javac.exe`. Continuing with the construction of this process group, applications from different versions of the Java SDK `bin` directory are grouped together:

- `C:\Program Files\Java\jdk1.6.0_17\bin\java.exe`
- `C:\Program Files\Java\jdk1.6.0_17\bin\javaw.exe`
- `C:\Program Files\Java\jdk1.6.0_16\bin\java.exe`
- `C:\Program Files\Java\jdk1.6.0_16\bin\javaw.exe`
- `C:\devel\jdk1.6.0_10\bin\java.exe`
- `C:\devel\jdk1.6.0_10\bin\javaw.exe`
- ...

The process groupings were constructed for the data sets collected from each computer. The number of process groups formed and the original number of distinct processes are recorded in Table 2. The distinct process names are based on the full path and executable name as recorded in the computer logs. This table also records the number of distinct executable names and the number of distinct directories in which those executables were located.

<table>
<thead>
<tr>
<th>Computer</th>
<th>GANDALF</th>
<th>ARAGORN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distinct Processes</td>
<td>561</td>
<td>1099</td>
</tr>
<tr>
<td>Distinct Executables</td>
<td>424</td>
<td>747</td>
</tr>
<tr>
<td>Distinct Directories</td>
<td>255</td>
<td>576</td>
</tr>
<tr>
<td>Process Groups</td>
<td>153</td>
<td>271</td>
</tr>
</tbody>
</table>

### Table 2: Number of Process Groups Formed

For both process families and process groups, the alerting process using the three different profile types for user profile generation were tested and compared with the results where no abstraction of processes was attempted for each of the three initial time window period.

### 4 Experimental Results

The number of alerts for the three alert types (new process, new hour for process and new day for process) were collected for each of the three profile type training schemes (constant window, moving window and growing window) for initial training periods of 14, 28 and 56 days. The alerts were generated for the data sets collected from both computers under study for a period of 30 weeks with regeneration of the dynamic profile types after each week of testing.
These results were collected for processes as the baseline case, and for process families and process groups to determine the effect of the different abstraction mechanisms.

In all tests, the starting date for alert generation for a particular computer began on the same date, so that the effect of training period could be compared. This means that the training periods extended back to different starting dates for the different training periods. The total number of alerts for the thirty week period for each of the tests conducted is displayed in Tables 3, 4 and 5. The results are discussed in further detail in the following paragraphs.

4.1 Baseline Case – All Processes

Table 3 displays the number of alerts generated for the different training approaches for user profile generation for both computers, using three different initial training time windows.

It can be seen for the three different training approaches when longer initial training periods were used, fewer alerts were generated. This is an expected result as the trained profile contains more information when a longer training period is used.

The results indicate that the training approach which generates the fewest alerts is the Growing Window approach. This approach is dynamic in nature and retains all past history for a user. While the Moving Window approach is also dynamic, it would appear from the results that when past history is removed from the user profile, higher numbers of alerts are generated. The Moving Window approach generated fewer alerts than the Constant Window approach for one computer but the results were reversed for the other.

### Table 3: Number of Alerts Generated over 30 Weeks for Process Families

<table>
<thead>
<tr>
<th>Training Period (days)</th>
<th>GANDALF New</th>
<th>Day</th>
<th>Hour</th>
<th>ARAGORN New</th>
<th>Day</th>
<th>Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 Constant Window</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>28 Constant Window</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>56 Constant Window</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>14 Moving Window</td>
<td>3</td>
<td>9</td>
<td>9</td>
<td>3</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>28 Moving Window</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>56 Moving Window</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>14 Growing Window</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>28 Growing Window</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>56 Growing Window</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

4.3 Process Groups

Table 5 displays the results of the number of alerts generated when processes were clustered together into process groups using the approach outlined in Section 2.3. It can be seen from the results that the number of alerts from the 30 week testing period is significantly lower than when individual processes were tested. The number of alerts generated for the growing window profiles with eight weeks of training data produce on average three or four alerts per user per week. For a large organization this could still be a significantly high number of alerts in total but it is a significant improvement on the average of five to twenty alerts per user per week when no aggregation is used.

### Table 5: Number of Alerts Generated over 30 Weeks for Process Groups

<table>
<thead>
<tr>
<th>Training Period (days)</th>
<th>GANDALF New</th>
<th>Day</th>
<th>Hour</th>
<th>ARAGORN New</th>
<th>Day</th>
<th>Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 Constant Window</td>
<td>96</td>
<td>129</td>
<td>120</td>
<td>126</td>
<td>159</td>
<td>154</td>
</tr>
<tr>
<td>28 Constant Window</td>
<td>88</td>
<td>123</td>
<td>118</td>
<td>115</td>
<td>143</td>
<td>139</td>
</tr>
<tr>
<td>56 Constant Window</td>
<td>85</td>
<td>112</td>
<td>112</td>
<td>98</td>
<td>134</td>
<td>130</td>
</tr>
<tr>
<td>14 Moving Window</td>
<td>122</td>
<td>142</td>
<td>141</td>
<td>155</td>
<td>164</td>
<td>165</td>
</tr>
<tr>
<td>28 Moving Window</td>
<td>117</td>
<td>141</td>
<td>133</td>
<td>133</td>
<td>149</td>
<td>153</td>
</tr>
<tr>
<td>56 Moving Window</td>
<td>97</td>
<td>120</td>
<td>118</td>
<td>126</td>
<td>145</td>
<td>148</td>
</tr>
<tr>
<td>14 Growing Window</td>
<td>112</td>
<td>141</td>
<td>131</td>
<td>98</td>
<td>131</td>
<td>129</td>
</tr>
<tr>
<td>28 Growing Window</td>
<td>105</td>
<td>134</td>
<td>123</td>
<td>98</td>
<td>128</td>
<td>127</td>
</tr>
<tr>
<td>56 Growing Window</td>
<td>85</td>
<td>109</td>
<td>112</td>
<td>89</td>
<td>119</td>
<td>115</td>
</tr>
</tbody>
</table>

5 Related Work

There are quite comprehensive event monitoring and event correlation products on the market. However they are typically platform specific and focus generally on network event correlation and/or centralized event monitoring and log management rather than post hoc correlation of events for forensic purposes.

We note also a considerable body of research in the area of security event correlation, ranging from alert correlation in intrusion detection systems (Ning, Cui et al. 2004; Morin, Mé et al. 2009) through to the standardization and formatting of audit or log records (Bishop 1995; Kent and Souppaya 2006) and audit reduction (Pfleeger and Pfleeger 2003).
Specific related work in insider misuse detection quite commonly has implemented systems aimed at specific operating systems without mention of the system’s applicability for other operating systems (Christoph, Jackson et al. 1995), or have developed approaches not aimed at operating systems at all, e.g. for database systems (Chung, Gertz et al. 1999). It is also quite common for researchers to use simulated data (Maybury 2006; Anderson, Selby et al. 2007).

Security personnel at the Los Alamos National Laboratory (Christoph, Jackson et al. 1995) implemented an approach to detect security policy violations on computer systems. This was capable of detecting activities by insiders abusing system privileges and outsiders attempting to gain clandestine access. They produced Network Anomaly Detection and Intrusion Reporter (NADIR) and UNICOS Real-time NADIR (UNICOS) to summarize user and system activity profiles. This system was aimed specifically at the UNICOS operating system and no extensibility was considered.

Chung, Gertz et al (1999) created a misuse detection system for relational databases. They computed user profiles from audit log data in an attempt to detect insider misuse of a financial database system in use in a bank. They present scenarios based on a bank teller misusing their privileges to gain customer credit card information or to transfer customer funds to their own account.

Another work by Shavlik, Shavlik et al. (2001) focused on profiling and identifying Windows 2000 users via keystroke dynamics. This work was intended to complement insider misuse detection rather than to detect insider misuse.

Maybury (2006) reported on a collaborative, six month workshop to characterize and create analysis methods to counter sophisticated malicious insiders in the United States Intelligence Community. His paper discusses a generic model of malicious insider behaviours, distinguishing motives, (cyber and physical) actions, and associated observables. The paper outlines several prototype techniques developed to provide early warning of insider activity, including novel algorithms for structured analysis and data fusion and reports performance assessment in an operational network against simulated insiders (an analyst, application administrator, and system administrator).

Anderson, Selby et al (2007) report on their behaviour profiling and misuse detection system, IRIS, which involves an intricate architecture of components to achieve real-time anomaly detection based upon a variety of inputs including operating system logs. IRIS employs the proprietary MQ Telemetry Transport protocol (MQTT) implemented by the IBM MQ MicroBroker. The system has been deployed to date only with simulated data and the authors note that their system requires a comprehensive set of user data, suggesting periods of time of the order of years.

Cathey, Ma et al (2003) concentrate on the detection of insider misuse in information retrieval systems. Their rules based approach is based on the creation of user profiles and relies on each user’s profile recording the types of documents that the user is allowed to retrieve from the information retrieval system. Again, this work is quite different to the research we have undertaken.

Ma and Goharian (2004) built user profiles to detect misuse in search systems based on activities learnt through clustering and relevance feedback. Goharian and Ma (2005) showed that they could achieve equivalent results to Cathey, Ma et al (2003) in detecting off-topic accesses to files in an information retrieval system by using a subset of the features that were originally proposed. The research presented in these approaches has focused on the detection of insider abuse of privileges by detecting anomalous behaviour from access control lists either prescribed by system administrators or from definitions generated from “learnt normal” behaviour. We note that these approaches have not targeted an operating system’s Security Audit event logs and the applications used for generation of user profiles but have concentrated on file and document accesses.

Magklaras and Furnell (2006) have produced a threat prediction specification language for modelling insider threat and intrusion incidents. This approach is quite different to the anomaly detection approach which we have presented in this paper.

Other researchers in the field of insider misuse detection have defined the threat of insider misuse (Bishop and Gates 2008; Pfleeger and Stolfo 2009) but have not moved to implementation. There have also been some frameworks for insider misuse proposed (Baek, Kim et al. 2008; Zhang, Ma et al. 2009) but again, these have not been fully implemented.

None of these papers have discussed the likelihood of high levels of false positives or ways to address this problem. There has been no mention in any of these papers of the different approaches we have presented for training user profiles and there has been no discussion of abstractions for collections of user applications.

6 Conclusions

We have shown that it is possible to build user profiles from data recorded in the Security Audit logs of computers running the Windows XP Operating System. The main information used from the events logged has been the names of the applications invoked by the users of the computers. It should be possible to use the same approach for any computers whose operating system records similar data.

We outlined three different approaches to generating user profiles, where a constant window or a growing window or a moving window can be used to specify the training time period used in the user profile. A user’s profile may change due to a change in their role or may be due to software upgrades or changes imposed by the user’s organization. Ideally such changes should not cause a major increase in the number of alerts generated by the system.

Our experimental results indicate that the growing window approach generates fewer alerts than the moving window approach and its dynamic nature captures changes in usage of applications without generating an excess number of alerts. This is thought to be due to the growing window approach not losing any past history from the user profile.
Our results also indicate that a longer training period will result in a richer user profile which has the effect of generating fewer alerts.

We also proposed and implemented two groupings of processes in an effort to reduce the number of alerts generated by the system. One of these, Process Families where all applications used on the computers were placed in a small number of groups, was too coarse in its groupings to be useful.

The second approach we named Process Groups and created by clustering together applications with the same path and/or the same executable name. This approach resulted in a significant reduction in the number of alerts when compared to individual process names, three to four alerts per user per week. We conclude that the use of this abstraction provides a positive benefit to the overall system.

It is of course necessary to investigate any alerts that are generated by a system such as the one described in this paper. If an organization employs a Standard Operating Environment, alerts generated from the first invocation of a process or process group could be automatically checked against the list of software in the SOE. Such a check should result in a further reduction in the number of alerts that need to be manually checked.

All user profiles have been generated for individual users. Using the abstraction approach it should be possible to make comparisons between users based on the collection of processes in their profile or alternatively to create profiles for an organization. Based on the role of similar people in the organization, a profile could be assigned based on the expected usage of applications. The profile could then be dynamically updated as the user settles in to their work routine. If this were done, there would be no need to wait for a training period for a new employee in an organization before their usage is routinely monitored. Care would have to be taken with such an approach that the user profile is properly regenerated once the employee has been working for the training window time period.

We believe that the approach outlined here for users of PCs could also be applied to other sources of data. Any source of data where user's actions are recorded could be used, e.g. web proxy logs where people's web browsing habits are recorded or ERP systems where user transactions are recorded. A possible application for this approach with ERP systems is the detection of financial fraud.

7 Future Work

In future work we intend to improve the system by changing and improving the way process groups are formed. When the process group clusters were formed, it was noticed that some of the clusters contained a single process name. A further reduction in alerts may be achieved by aggregating single processes by using their process family rather than their process group, i.e. by using a mixed model of aggregation.

Alternatively we could use some other means to determine if sufficient parts of a process’ path are the same for the process to belong to an existing process group. This could be implemented as a set proportion of the path or it could use an edit distance measure with a set threshold determining whether two processes belong in the same process group.

It should be noted that the work carried out to date is based on the presence or absence of a particular application in a user’s profile. In future work we will utilise the frequency of usage of applications, allowing us to build probabilistic models of usage for each person.

In our current work we have recorded but not made use of the amount of time that a user uses particular applications or groups of applications in their routine work. This is calculated from the start and stop times recorded for each process in the Security Audit log. By measuring the amount of time an application is used, better models of typical usage for a user can be built and this will be incorporated in our future work.

We intend to investigate the proportion of time a process is actively used while it is open by making use of a library which allows the amount of CPU time an application uses to be monitored to provide a more suitable measure of application usage. Keystroke logging tools may also provide a solution here as they can record which application is receiving key strokes and mouse clicks. Further investigation into monitoring application usage is therefore warranted.

References


Identification of Potential Malicious Web Pages

Van Lam Le, Ian Welch, Xiaoying Gao, Peter Komisarczuk
School of Engineering and Computer Science, Victoria University of Wellington
P.O. Box 600, Wellington 6140, New Zealand
{van.lam.le, ian.welch, peter.komisarczuk, xiaoying.gao}@ecs.vuw.ac.nz

Abstract
Malicious web pages are an emerging security concern on the Internet due to their popularity and their potential serious impact. Detecting and analysing them are very costly because of their qualities and complexities. In this paper, we present a lightweight scoring mechanism that uses static features to identify potential malicious pages. This mechanism is intended as a filter that allows us to reduce the number suspicious web pages requiring more expensive analysis by other mechanisms that require loading and interpretation of the web pages to determine whether they are malicious or benign. Given its role as a filter, our main aim is to reduce false positives while minimising false negatives. The scoring mechanism has been developed by identifying candidate static features of malicious web pages that are evaluate using a feature selection algorithm. This identifies the most appropriate set of features that can be used to efficiently distinguish between benign and malicious web pages. These features are used to construct a scoring algorithm that allows us to calculate a score for a web page’s potential maliciousness. The main advantage of this scoring mechanism compared to a binary classifier is the ability to make a trade-off between accuracy and performance. This allows us to adjust the number of web pages passed to the more expensive analysis mechanism in order to tune overall performance.

Keywords: Internet Security, Drive-by-download, malicious web page.

1 Introduction
A “malicious web page” refers to a web page that contains malicious content that can exploit a client-side computer system. This attack is delivered to client’s web browser when a malicious web page is requested. This type of attack is termed web-based client-side attack. The attack is delivered as part of the web page itself and is designed to exploit client-side vulnerabilities such as flaws in the implementation of browser functionality, interpreters of active content within webpages or scriptable client-side components such as ActiveX components. The result of an attack is often the installation of malware in the client system without the user’s consent and disclosure of user’s information. The user’s computer is often “owned” by attacker and can take part in generating SPAM and Distributed Denial of Service (DDOS) attacks.

Detection and blacklisting of malicious web pages has been the subject of several research projects. One effective approach is to build virtualised environments like high interaction client honeypots (Seifert 2007a) where suspicious web pages are loaded, executed and monitored to track potential malicious activities or behaviour. The virtualised environment allows this to be done without allowing any malware to be propagated to production systems. While this method shows very efficient results in term of detecting unknown attacks, it is expensive in terms of the resources required to provide a virtualised environment containing a complete operating system and is relatively slow with each visit taking up to 10 seconds. To attempt to reduce the required resources and increase the speed of the detection method, previous work (Seifert 2007a) has proposed using a hybrid approach where web pages are first filtered using a lightweight mechanism before being passed to the more expensive high-interaction mechanism. Our work focuses on improving the efficiency and effectiveness of that lightweight mechanism.

There are three main issues that we have explored in the design of our lightweight mechanism. Firstly, we want our mechanism to be lightweight in terms of its resource requirements. Therefore our mechanism is a data-mining algorithm that uses features derived from the static web page rather than runtime features gathered through the expensive process of loading the web page into a web browser within a virtual environment. This paper proposes a set of features that have been arrived at through analysis of known malicious web pages. These features are then evaluated by feature selection methods in order to find out the most suitable feature set to identify potential malicious web pages. Secondly, we want our lightweight mechanism to be tuneable to allow us to control the number of pages passed through to the more expensive mechanisms such as high interaction honeypots. This allows us to manage overall system performance. This has led us to develop a lightweight mechanism that computes a score rather than a simple binary malicious/benign classifier (Seifert, Welch and Komisarczuk 2008). By choosing the threshold that must be reached before passing on the web page, the overall performance can be tuned to reflect overall performance constraints. Thirdly, we believe that it is worse to miss a potential malicious web page (a false negative) than incorrectly class a web page as malicious (a false positive) and pass it onto the second stage for further analysis. Therefore, our aim has been to design a mechanism that minimises the number of false negatives whilst keeping the false positives at an acceptable level. Note that when taking resource usage into account that
there will most likely be a relationship between our choice of threshold value and the false negative rate and part of our interest is in understanding this relationship.

2 Background and Related Work

2.1 Web-based Client-side Attacks

As the number of Internet users has increased significantly, web-based attacks that use malicious web pages to exploit users’ system have become a primary concern in the Internet security. A web-based client-side attack happens when an Internet user visits malicious web pages which attempt to exploit the user’s browser vulnerabilities, plug-in application vulnerabilities or user’s operating system vulnerabilities in order to compromise the user’s system.

A web application is defined as an network application which is typically interacting with the web browser over the Internet (Mehdi 2007). Information service providers use web applications to deliver their services to users. To do that, they implement their business logic through web applications at a web server with an advertised URL (Gollmann 2008). To enrich their services, the providers can use more than one web server and backend servers and applications which work in cooperation in order to deliver services to the customers. In the client-side, there is the main application – web browser which users use to access information services from the providers. In order to expand their functionalities, almost all web browsers support adding third-party plug-in components such as Adobe Acrobat, Adobe Flash, Apple QuickTime, and Microsoft ActiveX.

To deliver malicious content to the client-side, an adversary first needs to publish malicious contents on the Internet. Compromising a web server is one of the common ways to deliver malicious contents. Various methods are reported to be used to increase attack effectiveness (Websense 2008, Sophos 2009, ScanSafe 2009, Symantec April 2009, ScienceDirect 2008, Websense 2009). Intruders can compromise a website by exploiting some vulnerabilities in the web server, exploiting a vulnerable web application (Symantic April 2009), vulnerable database applications such as SQL injection (Niels, Moheeb Abu and Panayiotis 2009, ScanSafe 2009, Microsoft 2009). The results from this compromising are inserting malicious contents which can be delivered to the client-side system (Niels, Moheeb Abu and Panayiotis 2009, Microsoft 2009). Some vulnerabilities in web server and web applications are reported as a very common issue (Provos, Mavrommatis, Abu and Monrose , Symantic April 2009). Web 2.0 technology, in addition, has become a common environment for attackers to spread their malicious contents (Websense 2008, Adam and Meledath 2008). Visitors are allowed to put arbitrary HTML and they can insert malicious codes into websites, insert links to malicious sites or even upload malicious files (Provos, McNamee, Mavrommatis, Wang and Modadugu 2007, Adam and Meledath 2008, Patsakis, Ashtenidis and Chatzidimitriou 2009, Lawton 2007).

After publishing their malicious contents on the Web, attackers must get users to visit the malicious web pages in order to make exploitation (Niels, Moheeb Abu and Panayiotis 2009). Spam is a common technique which intruders use to lure user to their malicious web pages. For instance, spam emails can contain a links to a malicious web page. Web blogs and social networking sites are also abused to get users to visit malicious sites (Garrett, Travis, Micheal, Atul and Kevin 2008). In addition, some legitimate sites have third-party contents like access counters, advertisements which refer to malicious sites (Alme 2008, Provos, McNamee, Mavrommatis, Wang and Modadugu 2007, Websense 2008, Barth, Jackson and Mitchell 2009). Moreover, search engine are also abused by attackers in order to get users to visit their malicious sites. Popular search terms are used to make malicious web pages be displayed in the search results (Keats and Koshy 2009, Alme 2008, Barth, Jackson and Mitchell 2009, Gyongyi and Garcia-Molina 2004, Websense 2009) so there is a very high chance for their malicious sites to be visited.

When a user visits a malicious site, malicious contents are delivered to exploit the user’s system. Malicious code is usually used to target a specific vulnerability of the web browser itself or plug-in applications (Jose, Ralf, Helen and Yi-Min 2007, Charles, John, Helen, Opher and Saher 2007). To discover available vulnerabilities in the user’s system, adversaries abuse scripting support via JavaScript, Visual Basic or Flash to collect information about the user’s computing environment (Provos, McNamee, Mavrommatis, Wang and Modadugu 2007). Moreover, obfuscation is used to hide exploit code in order to make malicious pages hard to be detected (Seifert, Welch and Komisarczuk 2008, Seifert 2007b, Seifert, Steenson, Holz, Yuan and Davis 2007).

In addition, Seifert’s study about malicious web servers shows that there are some available web exploitation kits (Seifert 2007b). These web exploitation kits are very powerful in term of compromising web servers and delivering malicious contents. The result from this kind of attacks is usually to redirect users’ requests to malware distribution networks. In addition, other related researches also show that malicious web pages are delivered by malware distribution networks (Provos, Mavrommatis, Abu and Monrose, Wang, Beck, Jiang and Roussev 2006, Jianwei, Yonglin, Jinpeng, Minghua, Xulu, Weimin and Yuejin 2007).

2.2 Related Work

In this section, we preview some current analysis methods which are used to detect malicious web pages. They are classified into three main approaches: Signature approach, state-change approach and machine learning approach.

2.2.1 Signature technique

In the signature approach, detection systems use known signature to detect malicious web pages. Signatures can be from some well-known Intrusion Detection Systems (IDS) or anti-virus applications. This approach is commonly used in the detecting system using low interaction client honeypot. Snort signature is used to detect malicious web pages in their HoneyC system (Seifert, Welch and Komisarczuk 2007). The HTTP responses from web servers are constructed under XML format, and then analysed against Signature signatures. In
Monkey-Spider system, Ikinci, Holz and Freiling also used signature approach to detect malicious websites. The contents of websites are crawled and stored in files. The crawled contents are then scanned by ClamAV – an antivirus application (Ikinci, Holz and Freiling 2008).

2.2.2 State-change technique (rule-based technique)

In addition, state-change approach is commonly used in the detecting systems using high interaction client honeypot – one of the efficient instruments to detect malicious web pages. The main idea of this approach is monitoring the state change in the client system during visiting an URL time. If there is any unauthorized state change during visitation, the visit URL is classified as malicious. In the Strider HoneyMonkeys system, a monkey program loads a browser, instruct it to visit each URL and wait for a few minutes for downloading process. The state changes in the system is then detected against unauthorized creating executable files or registry entries in the system (Wang, Beck, Jiang and Roussev 2006). Moreover, to detect drive-by-download attack, Moshchuk, Bragin, Gribble and Levy use event triggers. They create some trigger conditions to track unauthorized activities in process creation, file system and registry system. The trigger conditions also include any event that makes browser or the system crash. During visitation, if an URL make a trigger fire, it is classified as unsafe (Moshchuk, Bragin, Gribble and Levy 2006). The state change approach is also used by Xiaoyan, Yang, Jie, Yuefei and Shengli in their client honeypot system to collect Internet-based malware. A behaviour monitoring module is conducted to track malicious behaviour. It hooks native API, DLL functions and TDI in order to monitor all activities causing buffer overflow, accessing system resources such as process, network, file, and registry (Xiaoyan, Yang, Jie, Yuefei and Shengli 2008).

2.2.3 Machine Learning Approaches

Seifert et al. (Seifert, Welte and Komisarczuk 2008) proposed a novel classification mechanism to detect malicious web pages. This method is based on HTTP responses from potential malicious web servers which are then analysed to extract potential malicious characteristics. The method was used in a hybrid system in which all URLs are classified by static heuristic method and sent to high interaction client honeypot for verification. To classifying URLs by static heuristics method, some common attributes are chosen based on three proposed main elements in malicious web pages: exploit, exploit delivery mechanism and obfuscation. The first step in this method is collecting malicious and benign web pages and then extracting potential attributes from these web pages. In learning step, all attributes extracted from 5,678 instances of malicious and 16,006 instances of benign web pages were fed into Weka with 34.8 decision tree learning algorithm implementation. The outcome classifier from learning step was used to classify 61,000 URLs. This classifier had very good false positive rate (5.88%) but very high false negative rate (46.15%).

Hou et al proposed a machine learning approach to detect malicious web content (Hou, Chang, Chen, Laih and Chen 2009). The key point in this research is the method used to choose features according to the usages of DHML knowledge. The chosen features have to meet the requirement for abilities against obfuscation vs. accuracy. Three groups with 171 features were chosen. There are 154 features used to count the use of native Java functions. Nine features are also used to measure some elements inside a HTML documents. There is 8 advanced features are used to count the use of ActiveX object. In the first step, 965 benign and 176 malicious web pages were collected, analysed and labelled manually. The malicious web pages were then categorized into nine predefined types based on the skill used by attackers. In order to study about choosing type of features, the authors took some experiments with different chosen features. Decision tree algorithm is used in these experiments. While using all features cannot get high true positive and low false positive result, the combination of three features can get very good result. The authors also compared the results of different classification algorithms with the use of all the features. Four classification algorithms used in this comparison are decision tree, Naïve Bayes, SVM and boosted decision tree. The result showed that the boosted decision tree got the best performance with high true positive rate and low false positive rate.

To detect malicious web pages, Liang (Bin, Jianjun, Fang, Dawei, Daxiang and Zhaohui 2009) proposed the concept of abnormal visibilities. According to their studies, malicious web pages are usually changed in their display modes in order to be invisible or almost invisible. The authors showed three main forms of abnormal visibility. The first one is changing the width and height attributes of iframe in order to make embedded malicious codes invisible or almost invisible. Setting the display style of iframe ‘display: none’ is the second form of abnormal visibility. The last form is generating iframe tag dynamically in order to make obfuscation. Abnormal visibility fingerprints are created and used to detect malicious web pages. Each web page is scanned to detect any form of abnormal visibility. The detected value in any kind of abnormal visibility is compared with a threshold value. If the detected value is less than the threshold value, the web page has an abnormal visibility and is considered as a possible malicious page. To carry out the experiment, the authors detect 60 websites reported malicious by StopBadWare.org. They scanned 66882 pages from these websites and found 30561 malicious one. They also figured out that their system has low false positive (1.99%) and false negative rates (2.63%).

Ma et al. (Ma, Saul, Savage and Voelker 2009a) pinpointed a new approach to detect malicious web pages named lightweight URL classification. In this approach, they classify web pages based on relationship between URLs, their lexical and host-based features. It does not use contents of web pages in detection. Lexical features include any features which make the page ‘look different’. They can be the length of the host-name, length of the entire URL, number of dot in URL and so on. Hosted-base feature include IP address properties, WHOIS properties, Domain name properties and geographic properties. Naïve Bayes, SVM and Logistic Regression are used for classification. The authors used two experiments in their study. The first experiment is for
comparing between feature sets. The features were divided into nine feature sets and these sets were fed into the 1-regularized logistic regression (LR) classifiers. The results showed that using more features got better classification accuracy. In addition, their another experiment (Ma, Saul, Savage and Voelker 2009b) was conducted to build online learning algorithm to detect malicious web pages. They used the same feature as the experiment (Ma, Saul, Savage and Voelker 2009a). There were three online algorithms implemented: Perception, Logistic Regression with Stochastic Gradient Descent and Confidence-Weight. They compared their online learning algorithm with Support Vector Machine (SVM). The results showed that SVM needed more training data set in order to get better accuracy but their algorithms did not.

To build an inductive learning model to detect malicious web pages, Liu et al. (Liu and Wang 2009) extracted features from HTTP responses such as iframe, javascript, body redirect, css redirect etc. The inductive learning model consisted of behaviour signatures based on extracted features and the relationship of features. The results from their experiment showed that the inductive learning model missed many malicious web pages (46.15%).

Chia-Mei et al (Chia-Mei, Wan-Yi and Hsiao-Chung 2009) proposed a model to detect malicious web pages based on unusual behaviour features such as encoding, sensitive key word splitting and encoding and some dangerous JavaScript functions. To classify web pages, they created a scoring mechanism which cored based on 9 predictor variable. Moreover, weights for each predictor variable were decided by training phrase. The results from their experiment showed that their model worked very well. However, their dataset was very small with 460 benign and 513 malicious web pages.

Shih-Fen et al. (Shih-Fen, Yung-Tsung, Chia-Mei, Bingchiang and Chi-Sung 2008) proposed a novel semantics-aware reasoning detection algorithm to detect malicious web pages (SeAR) which was based on structures of HTML codes. Firstly, they defined templates for HTML codes. For each tested HTML code, the distance between the tested HTML code and templates were calculated. Secondly, the best match was chosen based on the distance and weight of the template. Finally, threshold was used to make decision whether web pages were classified as malicious or benign. The outcome from this research is very good but their dataset had only 147 malicious instances (no benign one).

Cova et al. (Cova, Kruegel and Vigna 2010) presented a novel approach which used anomaly detection and emulation to identify malicious JavaScript Code. The features were chosen based on sequence of carrying out an attack: redirection and cloaking, de-obfuscation, environment preparation, and exploitation. They argued that not all of the features were necessary for an attack happening and classified the features into two groups: useful features and necessary features. To extract features, they used emulated HTML browser HtmlUnit (Gargoyole). They carried experiments on over 115K web pages and their approach achieves very good outcome in comparison to other approaches such as ClamAV, PhoneyC and Capture-HPC.

While there is a few of works focusing on identifying malicious web pages, this paper presents a mechanism to detect potential malicious one in order to reduce number of suspicious web pages which need to be investigated further by detection instruments or experts.

3 Scoring Mechanism

This work focuses on how to reduce number of suspicious web pages but minimize missing attacks. A scoring mechanism is proposed to work as a filter which classifies suspicious web pages into classes: benign web pages and potential malicious web pages. Only potential malicious web pages are forwarded to detection devices or experts for further investigations (Fig. 1).

![Figure 1: Scoring Mechanism](image)

3.1 Feature Selection

The first step on feature selection is to identify potential malicious features which can distinguish between benign web pages and malicious one. By analysing the selected common malicious web pages, we find that there are three main groups of malicious contents of web pages as follows:

- Foreign contents are malicious contents which are loaded from outside along with suspicious web pages. These contents can be loaded with suspicious web pages by some of malicious HTML tags such as frame, iframe, image source... Iframe is especially known as very common method to load outside malicious web pages along with suspicious one (Provos, Mavrommatis, Abu and Monrose). In almost all of cases, foreign malicious contents are resulted from compromises or uncontrolled third-party contents such as advertising and site hit counters.

- Script contents are known as the most common malicious contents of malicious web pages. In almost all of cases, script codes are used for two main purposes: delivering and hiding malicious codes by obfuscations. We identify some of potential malicious features from scripts which could distinguish between benign web pages and malicious web pages, such as script size, string
size, word size, argument size, character distribution...
- Exploit code contents are the core contents of malicious web pages. They are target specific vulnerabilities in web browsers, plug-ins or operating systems. Some of HTML tags known as delivery of potential malicious codes are applet, object, embed... However, there are rarely malicious codes found in this direct form. In almost cases, exploit codes are encoded in scripts with obfuscations to hide from detection devices.

Information gain is a measurement of how much information a feature provides about the class. The greater the information gain of an attribute a, the more valuable is a of attribute a. The greater score an instance x has in each group, the more likely it is classified as potential malicious class.

A group score of instance x is calculated as follows:

$$GS_{g}(x) = \sum_{a \in g} \frac{|x_a - \mu_a|}{\delta_a}$$

Where g is an attribute group which can be foreign content group, script content group or exploit content group; a is an attribute of g; x_a is value of attribute a of instance x; \(\delta_a\) is a standard deviation of attribute a which is estimated during training a set of benign instances; \(\mu_a\) is mean of attribute a which is estimated during training a set of benign instances.

The greater score an instance x has in each group, the more likely it is classified as potential malicious class. If \(T_g\) is chosen as a threshold for content group g in order to identify potential malicious instances, the rule of classification is as follows:

$$x = \begin{cases} \text{potentially malicious if } \exists g \in G: GS_g(x) > T_g \\ \text{otherwise, } x \text{ is benign} \end{cases}$$

Any page will be classified as potential malicious that has a group score greater than the threshold value for that group.

### Data Collection

To get dataset for our experiments, we firstly collect candidate web pages which include both malicious and benign one. To collect benign web pages, we collect hot search terms from Google Search Engine (Google 2010) and then feed these search terms to Yahoo API websearch (Yahoo 2010) to get top 10 URLs from the search results. In addition, we collect malicious web pages from some of common public announced malware and exploit websites like Blade-defender.org, Clean-mx.de, ParetoLogic.com, Malwaredomainlist.com. These selected web pages are verified by our Capture-HPC, a high interaction client honeypot (Seifert and Steensom 2009).

Secondly, we create a low interaction client honeypot which interacts with web servers to request for the selected web pages. The HTTP responses from web servers are extracted based on the attributes and their potential values described on Table 1. We totally collect 33646 instances of web pages, including 33422 instances of benign web pages and 224 instances of malicious one.

### Experiments

To evaluate our scoring mechanism, we divide dataset into two subsets as follows:
- Training dataset consists of 20,000 benign instances and it is used for training scoring algorithm to calculate mean and standard deviation for each attribute.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Group 1: Foreign Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Number of redirection</td>
</tr>
<tr>
<td>2</td>
<td>Number of iframe and frame tag</td>
</tr>
<tr>
<td>3</td>
<td>Number of external link in iframe and frame tag</td>
</tr>
<tr>
<td>4</td>
<td>Iframe and frame link length: Median</td>
</tr>
<tr>
<td>5</td>
<td>Ratio of vowel character in iframe and frame link: Minimum</td>
</tr>
<tr>
<td>6</td>
<td>Ratio of special character in iframe and frame link: Minimum</td>
</tr>
<tr>
<td>7</td>
<td>Number of external links (except iframe and frame)</td>
</tr>
<tr>
<td>8</td>
<td>Other link length: Minimum</td>
</tr>
<tr>
<td>9</td>
<td>Number of scripts</td>
</tr>
<tr>
<td>10</td>
<td>Number of script lines</td>
</tr>
<tr>
<td>11</td>
<td>Number of script word</td>
</tr>
<tr>
<td>12</td>
<td>Ratio of special character in scripts</td>
</tr>
<tr>
<td>13</td>
<td>Script length: Minimum</td>
</tr>
<tr>
<td>14</td>
<td>Script line length: Minimum</td>
</tr>
<tr>
<td>15</td>
<td>Script string length: Maximum</td>
</tr>
<tr>
<td>16</td>
<td>Script word length: Minimum</td>
</tr>
<tr>
<td>17</td>
<td>Script function argument length: Minimum</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature</th>
<th>Group 2: Script Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>Number of objects</td>
</tr>
<tr>
<td>19</td>
<td>Number of applets</td>
</tr>
<tr>
<td>20</td>
<td>Object link length: Maximum</td>
</tr>
<tr>
<td>21</td>
<td>Ratio of special character in object links</td>
</tr>
<tr>
<td>22</td>
<td>Ratio of vowel character in object links</td>
</tr>
<tr>
<td>23</td>
<td>Number of object attributes: Median</td>
</tr>
<tr>
<td>24</td>
<td>Applet link length: Minimum</td>
</tr>
<tr>
<td>25</td>
<td>Ratio of special character in applet link</td>
</tr>
<tr>
<td>26</td>
<td>Ratio of vowel character in applet link</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature</th>
<th>Group 3: Exploit Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>Number of scripts</td>
</tr>
<tr>
<td>28</td>
<td>Number of script lines</td>
</tr>
<tr>
<td>29</td>
<td>Number of script word</td>
</tr>
<tr>
<td>30</td>
<td>Ratio of special character in scripts</td>
</tr>
<tr>
<td>31</td>
<td>Script length: Minimum</td>
</tr>
<tr>
<td>32</td>
<td>Script line length: Minimum</td>
</tr>
<tr>
<td>33</td>
<td>Script string length: Maximum</td>
</tr>
<tr>
<td>34</td>
<td>Script word length: Minimum</td>
</tr>
<tr>
<td>35</td>
<td>Script function argument length: Minimum</td>
</tr>
</tbody>
</table>

Table 1: Appropriate Features for Identifying Potential Malicious Web Pages

According to our analysis, we select 52 potential features from these main malicious contents. If a feature appears more than once, we use four values to measure it at the first sight: minimum, maximum, mean and median. However, only one measured value for each feature is chosen for scoring algorithm.

Secondly, we use information gain as a measurement method to choose high valuable features only. Information gain for an attribute a is defined as follows:

$$IG(S, a) = \text{Entropy}(S) - \sum_{v \in S_a} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v)$$

Where S is collection of instances, S_a is a subset of S with relevant value v of attribute a. The greater information gain an observed attribute obtains, the higher value it contributes to the process to identify malicious web pages. The training dataset which is used to calculate information gain must have both malicious and benign instances. There are 26 potential features selected based on information gain (Table 1).

### 3.2 Scoring Mechanism

Our scoring algorithm works based on the concept of standard score which measure how many standard deviations a value of observed attribute is far from the mean (Carroll and Carroll 2002). Each instance has three types of scores based on three groups of contents of web pages: Foreign content score, script content score and exploit content score.

A group score of instance x is calculated as follows:

$$GS_{g}(x) = \sum_{a \in g} \frac{|x_a - \mu_a|}{\delta_a}$$

Where g is an attribute group which can be foreign content group, script content group or exploit content group; a is an attribute of g; x_a is value of attribute a of instance x; \(\delta_a\) is a standard deviation of attribute a which is estimated during training a set of benign instances; \(\mu_a\) is mean of attribute a which is estimated during training a set of benign instances.

The greater score an instance x has in each group, the more likely it is classified as potential malicious class. If \(T_g\) is chosen as a threshold for content group g in order to identify potential malicious instances, the rule of classification is as follows:

$$x = \begin{cases} \text{potentially malicious if } \exists g \in G: GS_g(x) > T_g \\ \text{otherwise, } x \text{ is benign} \end{cases}$$

Any page will be classified as potential malicious that has a group score greater than the threshold value for that group.
- Testing dataset contains 13,646 instances with 13,422 benign instances and 224 malicious one. This dataset is used to test the scoring mechanism.

The experiment is carried in three steps. Firstly, training dataset is fed into our scoring mechanism in order to calculate some statistic values such as mean, standard deviation. Secondly, we calculate group scores for each instance in the testing dataset. Each instance has three types of scores: foreign content score, script content score and exploit content score. Finally, we adjust threshold score values in each group in order to find the relationship between false negative rate and the number of identified potential web pages.

6 Results

![Figure 2: The relationship between false negative rate and number of potential malicious web pages.](image)

We use 20,000 instances of benign web pages to train our scoring algorithm and 13646 instances of malicious and benign web pages for testing. To find out the relationship between false negative rate and the number of identified potential malicious web pages, we adjust the value of score threshold in each group and calculate number of negative. The threshold start from the maximum value of each group score, and then reduce to the value corresponded to the percentage of potential malicious web pages. Figure 1 shows the relationship between the number of identified potential malicious web pages and false negative rate. When number of potential malicious increases, false negative rate decreases. Our aim is to minimize the false negative rate, in Figure 1 this is achieved when number of potential malicious web pages reaches 14% of the total number of instances in the testing dataset. In the other word, we can reduce 86% number of suspicious web pages without missing attacks.

7 Conclusion

This paper presents a scoring mechanism to estimate maliciousness of web pages in order to reduce the number of suspicious web pages which need to be analysed by a secondary mechanism such as high-interaction honeypot. The advantages of this scoring mechanism are discussed as using lightweight static features, capability to make trade-off between number of potential malicious web pages and false negative rate (that is, missing an attack).

Three main groups of malicious contents are identified in this paper. Based on these contents groups, we extracted 52 potential features from both malicious and benign web pages. Information gain is used in order to identify 26 potential features. Each web page has three scores corresponded to three contents groups. Thresholds are chosen for each content group. A web page is classified as potential malicious web pages if it has at least one group score higher than threshold.

The proposed scoring mechanism is initially tested on 13,646 instances with 224 malicious web pages. The result shows that there is capability to make trade-off between number of potential malicious web pages and missing attacks.

This work however has some limitations, which are identified and required for future works. Firstly, a limited number of malicious samples (224 instances) may not present all statistical characteristics of malicious web pages. Secondly, only information gain feature selection method is used in the feature selection process. Other feature selection methods could be investigated in order to have a good comparison. Thirdly, there are three contents groups with three thresholds but the relationship between them in order to form the overall score with only one overall threshold has not identified yet.

References


IEEE 802.11 Chipset Fingerprinting by the Measurement of Timing Characteristics

Guenther Lackner$^1$ Peter Teufl$^1$

$^1$ Institute of Applied Information Processing and Communications (IAIK)
University of Technology Graz
Inffeldgasse 16a, 8010 Graz, AUSTRIA
Email: guenther.lackner@iaik.tugraz.at, peter.teufl@iaik.tugraz.at

Abstract

In this paper we present a technique to create WLAN device fingerprints by measuring timing properties without the use of special-purpose hardware. Our proposed process is absolutely passive and cannot be detected by the targeted device. The timing measurement is based on a delay caused by the hardware implementation of the CRC checksum algorithm at the network interface card (NIC) of the client. This delay turned out to be significant for a large number of different chipset implementations. The ability of identifying connected devices could improve the security of a wireless network significantly. It could help to enhance access control mechanisms and would deliver valuable real time information about the connected clients. As a proof of our concept we present a prototype implementation called WiFinger to evaluate our approach.

Keywords: IEEE 802.11, MAC Address Spoofing, passive Chip-set Fingerprinting, Significant Histograms

1 Introduction

During the last years wireless networking spread into countless fields of application like mobile telephony, wireless computer networks, mobile sensor networks, and many more. This wireless revolution daily pervades new areas of our lives providing an increase in the grade of mobility, usability and comfort. But there seems to be a price to pay. Due to their open-air propagation nature, wireless networks raise a new variety of potential security and privacy risks for attackers. During the development and definition of related industrial standards there was obviously not enough focus on security issues. Some of the most popular and widest spread standards in wireless computer communications like WEP (Wireless Equivalent Privacy) are full of security breaches which open up attackers. During the development and definition of related industrial standards there was obviously not enough focus on security issues. Some of the most popular and widest spread standards in wireless computer communications like WEP (Wireless Equivalent Privacy) are full of security breaches which open up all gates to attackers (Fluhrer et al. 2001). Further on, even state-of-the-art standards like WPA2 (Wireless Protected Access based on AES) begin to crumble (Airtight-Networks 2010).

The aim of our work is to bring more safety into the wireless world by identifying network participants via timing measurements. Our approach focuses on the widely spread standards of the IEEE 802.11 family. We do not intend to improve or alter encryption mechanisms. With our tool called WiFinger, one could be able to detect, and in succession prevent layer 2 MAC address spoofing attacks. If unauthorized participation of attackers in a wireless network can be detected many possible attacks could be prevented. Our work is based on a technique of passive fingerprint creation by observing the timing behavior of IEEE 802.11 compliant devices without the necessity of special purpose hardware like frequency spectrum analyzers.

This paper is organized as follows. Section 2 introduces related work. Section 3 describes our method of creating fingerprints of IEEE 802.11 device chipsets. Section 4 introduces the fundamentals of the classification method developed us, based on Self Organizing Maps. Section 5 illustrates the design and implementation details of the WiFinger software. The real-world applicability and performance analysis is placed in section 6. Section 7 provides a short outlook on future extensions and improvements and concludes the article.

2 Related Work

A straightforward approach for device identification is to utilize the device addresses such as the MAC (Media Access Control) address (layer 2) or the assigned IP address (layer 3). This can easily be achieved by analyzing relevant ARP (Address Resolution Protocol) traffic (Plummer 1982). Unfortunately, this approach has a major drawback. Most devices allow to modify their assigned MAC address with easy to use, free software tools. This problem might be tackled by creating fingerprints of network hardware. This would allow the identification of any device by observing its external characteristics.

Remote Physical Device Fingerprinting: One of the most significant papers in the field of device fingerprinting has been published by Tadayoshi Kohno and his team at UC San Diego (Kohno et al. 2005). Kohno developed a method to identify remote devices by exploiting small, microscopic deviations in the hardware: clock skews. By analyzing the deviation of TCP or ICMP timestamps over a certain period of time it is feasible to distinguish different hardware clocks and thus different devices. The main difference to our approach is, that Kohno et al. is not applicable in an encrypted wireless environment as it needs plaintext TCP or ICMP payloads for analysis.

Radio Frequency Fingerprinting: This fingerprinting technique is based on the signal characteristics of turn-on transients of wireless transceivers. These transients are specific to each different transceiver and thus are perfectly suited as data source for fingerprint generation. Transient capturing and analysis requires a special infrastructure for signal capturing which is expensive and has
to be operated by experts. Hall et al. evaluated the performance of the fingerprinting method with 30 transceivers. For each transceiver 120 signals were captured and used for the performance evaluation. The results indicate that the method is capable of achieving a very low false positive rate (0% during the evaluation) and a high detection accuracy (95% during the evaluation). The biggest disadvantage of this method is the special hardware needed for signal capturing which limits the broad deployment. (Hall et al. 2006)

Passive Data Link Layer Fingerprinting: Franklin et al. (McCoy et al. 2006) identified an imprecision in the IEEE 802.11 Media Access Control specifications that has been interpreted differently by wireless NIC firmware developers. The time between sending two so called beacon frames used for network detection is not strictly defined. This method is able to classify different firmware versions instead of the underlying hardware. For creating a meaningful fingerprint a large number of probe-requests need to be captured. Due to the fact that a NIC willing to join a network, usually just needs a hand-full of these requests it could take a rather long time to obtain a suitable amount of data. Another significant drawback is that fingerprinting may easily be avoided by using passive-scanning or altering the device firmware. (McCoy et al. 2006) Some improvements to this approach have been developed by Loh et al. (Desmond et al. 2008).

Active Fingerprinting by Timing Analysis: Bartolomiej Sieka work on device fingerprinting (Sieka 2006) is probably the one closest related to our approach. It uses the time that elapses between the first acknowledgement is sent and the moment the authentication response is sent. For classification purpose, support vector machines are used. The drawback of this approach is its limitation to the authentication phase for measurements. As this phase only occurs during the initialization of the connection, Sieka actively needs to provoke the repetition of it by sending specifically crafted 802.11 frames. This could be detected by an intrusion detection system or the device to fingerprint, allowing it to counteract. As the next section describes, our approach is immune against such countermeasures as it is absolutely passive.

3 Fingerprinting on Layer 2

Creating a fingerprint of a device is the process of identifying it by the observation of its external characteristics. We developed one possibility of creating fingerprints of IEEE 802.11 devices by observing their timing behavior. This section provides a compact overview of the basic principles.

Our approach examines the timing behavior of IEEE 802.11 devices generating so called acknowledgement packets (ACK). Due to the fact that IEEE 802.11 standards follow the principle of half-duplex communication, a collision avoidance technique is generally needed to be deployed. If a participant A (client) has sent a data frame to participant B (access point), A is not able to observe if its message was transmitted correctly or collided with a data frame sent from another participant at the same time. IEEE 802.11 standards are based on the so called Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism (Brenner 1997). In this paper we just shortly describe the IEEE 802.11 media access system. For further details consult (IEE 1999) and (Kerry 2007).

To inform A that its data frame was transmitted correctly, B generates and transmits an ACK packet after having correctly evaluated the CRC checksum of A’s data frame. If the CRC check fails, no ACK will be sent and A retransmits the data frame after a certain time (IEE 1999).

The computing and evaluation of the CRC checksum plus the generation of the ACK packet takes a certain amount of time. This amount depends on the hardware implementation of the CRC algorithm, the firmware and some other components of the used wireless network device. We call this delay The Acknowledge Delay. If one regards the distribution of a certain number of ACK delay values the outcome represents a significant property of the used wireless device. This outcome is called Significant Histogram.

Based on these Significant Histograms it is possi-
4 Classification

This section describes the classification algorithm based on multiple Self Organizing Maps (SOM) arranged in a tree and implements an extension which improves the quality of SOMs when used for supervised learning.

4.1 SOM based algorithm

Self Organizing Maps (SOM) belong to the broader category of neural networks (Kohonen 2001). They are mainly used for unsupervised learning and the visualization of high-dimensional data. In this paper we employ SOMs for a supervised classifier. Although, other supervised algorithms like neural networks or support vector machines are better suited for classification tasks, we still focus on the SOM due to one main reason: The visual representation of the data in a 2D map allows us to quickly gain insight on the main reason: The visual representation of the data classification tasks, we still focus on the SOM due to one main reason: The visual representation of the data in a 2D map allows us to quickly gain insight on the visual representation of the data in a 2D map allows us to quickly gain insight on the.

By labeling the SOM units during the training process according to the class labels of the data they represent, the SOM can also be employed for supervised learning. However, due to the unsupervised nature of the SOM, the class information is not taken into account during the training process. Therefore, the accuracy of the trained model might be inadequate for the separation of data belonging to different classes. This data is mapped by the same units and leads to classification errors that decrease the accuracy of the SOM. In order to cope with this issue, our classifier utilizes multiple SOMs arranged in a tree.

Whenever the model of a trained SOM is not precise enough to separate data of different classes accurately, we extract this data, train a new SOM on this data and link the units of the old SOM covering this data to the new SOM. Therefore, we do not need to deal with SOM model complexity manually. If the model of a trained SOM is not accurate enough, the algorithm simply trains a new SOM that is only trained on the data which requires more complex modeling (indicated by a higher misclassification rate).

The multiple SOMs are trained and arranged in a tree according to this algorithm:

1. Train a SOM on the input data
2. Label the units according to the classes they represent
3. Calculate misclassification rates for all classes
4. Extract the data of classes that cannot be separated with an error rate lower than a given threshold
5. Mark the units that cover the extracted data to indicate that the actual classification will be made in the next SOM
6. Go to step one and train a SOM for new extracted data. Repeat these steps until the error conditions are met or only two classes remain in one SOM.

A simple example with five classes is shown in Figure 4. The first SOM is trained on the complete data set and the misclassification rates are determined. The example shows that classes A/B/C and D/E cannot be separated accurately by the first SOM. Therefore, two data sets for A/B/C and D/E are extracted. For both data sets, new SOMs are trained and the units corresponding to these classes in the first SOM are linked to the newly trained SOMs. In case of A/B/C, the second SOM is able to separate the class C from A/B but the misclassification rates for A/B are still too high. Therefore, another SOM is trained that increases the classification performance. The picture indicates that the SOM for A/B still has some misclassification errors, which cannot be removed without losing generalization (and thereby overfitting the data).

A simple example with five classes is shown in Figure 4. The first SOM is trained on the complete data set and the misclassification rates are determined. The example shows that classes A/B/C and D/E cannot be separated accurately by the first SOM. Therefore, two data sets for A/B/C and D/E are extracted. For both data sets, new SOMs are trained and the units corresponding to these classes in the first SOM are linked to the newly trained SOMs. In case of A/B/C, the second SOM is able to separate the class C from A/B but the misclassification rates for A/B are still too high. Therefore, another SOM is trained that increases the classification performance. The picture indicates that the SOM for A/B still has some misclassification errors, which cannot be removed without losing generalization (and thereby overfitting the data).

The trained SOM hierarchy of SOM tree is used for the classification of unknown data in this way:

- Present the data to the root SOM of the tree and determine the best matching unit (BMU)
- If the unit is linked to another SOM further down in the hierarchy, load this SOM and go to the previous step. If the unit is not linked to another SOM, return the class label of the unit.

This procedure is indicated in the example by the two classification paths for data vectors from class B end E.

The described strategy is employed for the classification of the WLAN chipsets. The same technique was already successfully applied to other classification problems, especially for network traffic classification (Payer et al. 2005). For SOM training, the SOM toolbox (Vesanto et al. n.d.) which is available for Matlab@((MATLAB - The Language of Technical Computing: Mathworks, http://www.mathworks.com n.d.) was used. The classification algorithm was also implemented in the tool WiFinger.
Classification of data belonging to class B

Classification of data belonging to class E

Figure 4: SOM Tree example: The classes A/B/C and D/E cannot be separated within the accepted error threshold, therefore two new SOMs are trained for A/B/C data and D/E data. Another SOM is attached due to a high number of misclassifications for A/B. The two arrows indicate the paths that are used for the classification of unknown data.
4.2 Features

An initial evaluation of WLAN traffic showed us, that the ACK delays of different packets vary from WLAN chipset to chipset and therefore could be used to identify such chipsets. By analyzing the spectrum of the ACK delays of the same chipset we can derive a histogram that represents the number of packets over the various observed delay times. In addition we capture the packet size in order to find out whether the ACK delay also depends on the packet size. The packets of a session – the time frame, where packets of a given chipset are captured – are arranged in the histograms in the following way:

1. Collect the ACK delays for each session of traffic generated by different WLAN chipsets.

2. For each 50 packets, create a 3D histogram which stores the frequency of the packets with a specific ACK delay and packet size. Each histogram is converted into a feature vector used for SOM training and classification.

3. Train a SOM tree with the histograms of the different WLAN chipsets.

4. The trained SOM tree is used for the classification of new traffic.

The length of the feature vectors depend on the number of analyzed ACK delay values (indicated as n) and packet size values (indicated as m). By storing the number of packets for given delay values and packet sizes we gain a 3D histogram that can be converted into a feature vector with \( f = n \times m \) entries. In order to keep the feature vectors at a feasible length, we need to map delay and packet size ranges into single values. E.g. by considering ACK delay values from 1 ms to 300 ms (\( n = 300 \)) and packet size values from 1 byte to 1600 bytes (\( m = 1600 \)) we would get feature vectors with \( f = 300 \times 1600 = 480000 \) entries, which is not feasible. However, this resolution is not needed and on the contrary would decrease the accuracy of the classifier. Therefore, we reduce the number of features by mapping several ACK delay values and packet size values into bins representing value ranges. E.g. if we use a bin size of 10 ms for the ACK delay (then \( n = 30 \)) and a bin size of 40 bytes for the packet size (then \( m = 40 \)) the feature vector length is reduced to \( f = 30 \times 40 = 1200 \).

In Figures 6 and 7 two histograms based on delay information only (packet size is ignored) are shown. We observe that there is a significant difference between the analyzed chipsets. The role of the packet size combined with the delay values is visualized in Figures 8 and 9. Here we observer that the ACK delay values also depend on the packet size – at least for certain chipsets. In Figure 8 the captured data of the Agere chipset clearly shows that there is such a dependence. In contrast this dependence cannot be observed when analyzing the Edimax chipset (Figure 9).

By integrating both features into the classification process we are able to increase the accuracy compared to classifiers based on the delay information only. For further details and evaluation results we refer to the results section.

5 WiFinger

As proof of concept of our approach, a small linux command line utility called WiFinger was developed and implemented in C/C++. The implementation was kept small in anticipation of possible use on handheld PCs as passive scanning devices. Additionally to
PACKET SIZE (%) | ACK DELAY (MS) | MEAN VALUE | STANDARD DEVIATION
--- | --- | --- | ---
1 | 100 | 200 | 300
2 | 200 | 300 | 400
3 | 300 | 400 | 500

Figure 9: EdimaxTech (Chipset 6): Here, we cannot observe a dependency between the packet size and the ACK delay values.

5.2 Feature Measurement

Figure 11 illustrates the process of feature measurement. Depending on the type of the captured frame, one of three states is entered: IDLE, CAPTURED DATA FRAME and CAPTURED ACK OF DATA. Recognized types are data frames and frames containing an ACK. During contention free periods, DATA frames can contain contention free acknowledgments, hence referred to as CF-Ack. During these periods DATA frames with embedded CF-Acks may appear in direct succession of each other. Thus as a special case, the states LAST FRAME WAS DATA and DATA/ACK PAIR CAPTURED can be entered during the same pass. MAC addresses are read from the frame header’s address field which always contains the wireless destination station and the address 2 field which always contains the sending wireless station (IEE 1999). Both fields are 6 bytes long and start at byte 4 and 10 respectively.

The destination station’s MAC address, payload size and the time of reception of the latest data frame are saved in temporary variables. If the frame acknowledges the immediately previously sent data packet, the acknowledge delay is measured as time between the reception of the last data frame and the reception of the acknowledging frame (see Figure 1). Note that interval is longer than the Shortest Inter-Frame Space, since the delay of receiving the data frame is added.

Measurements on broadcast addresses are discarded. Since an ACK frame carries only a destination address (see figure 2), it is possible that on-air data is missed and a later ACK frame is mistaken for an expected acknowledgment. To minimize such mistakes, the sending stations address of the previous data frame is checked against the destination address of the following ACK frame. This only works outside of contention free periods since the destination address of frames containing CF-Ack does not need to match the address of the previously transmitting station. The following frame types are relevant during contention free periods (IEE 1999):

- CF-End + CF-Ack
- Data + CF-Ack
- Data + CF-Ack + CF-Poll
- NoData + CF-Ack
- NoData + CF-Ack + CF-Poll

5.3 Usage of Matlab® for SOM training

Figure 10 depicts the interaction between WiFinger and Matlab®. For each host WiFinger writes a simple tab and newline delimited text file. Measurements exported for use in Matlab® are not preprocessed by WiFinger. During the experimentation process this provided more flexibility in finding the best parameters for classification. Like in (Payer et al. 2005) the used scripts produce a SOM-tree which is then automatically loaded by WiFinger.

5.4 Usage of libSom for SOM classification

libSom (Payer et al. 2005) provides loading of SOM-trees, datatypes for SOMs and vectors as well as classification functionality. Classification works as described above in section 4. For each host a SOM-Vector is used to save a histogram of ACK delays. The file somconfig generated by the scripts was extended to save the parameters used in the training of the SOM-tree. These are:

$\text{http://www.tcpdump.org/}$
Figure 10: WiFinger Dataflow

- minimum
- maximum
- number of subdivisions

for each of the two features, acknowledge delay and data frame size.

Captured values above the given limits (500ms) are discarded. If the number of values between minimum and maximum differ from the number of subdivisions, values are scaled to fit the chosen resolution. After a variable number of measured ACK delays, a copy of the SOM-Vector is normalized and the resulting Significant Histogram is classified. Optionally, a number of measurements can be defined after which the histogram is reset.

6 Results

For performance evaluation, the packet size and ACK delay time was taken from real traffic data with WiFinger. To facilitate this, an Aironet 350 access point was set up with Internet access. Traffic was measured using a 802.11b wireless NIC with Orinoco chipset, capable of capturing all layer 2 data in rfmon mode. In turn, six different wireless NICs with known chipsets were used to generate the traffic. The access point shows up in the classification results as chipset 3.

60% of this data was used for the SOM training process. The remaining test data was used to create simulated sessions with 500 respectively 1000 packets. This session based classification should give a hint on how much data from a chipset is needed to get an accurate classification result. Table 1 gives details on the evaluated chipsets and the training respectively test sets.

<table>
<thead>
<tr>
<th>Chipset</th>
<th>training/test data in packets</th>
<th>test data sessions 500/1000 pkts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Atheros</td>
<td>35105/23404</td>
<td>47/24</td>
</tr>
<tr>
<td>2: Agere</td>
<td>53006/35337</td>
<td>71/36</td>
</tr>
<tr>
<td>3: Aironet</td>
<td>292858/195239</td>
<td>391/196</td>
</tr>
<tr>
<td>6: RaLink</td>
<td>19266/12843</td>
<td>26/13</td>
</tr>
<tr>
<td>7: BCM4306</td>
<td>149864/99910</td>
<td>200/100</td>
</tr>
<tr>
<td>8: Inteli2100</td>
<td>41390/27594</td>
<td>56/28</td>
</tr>
<tr>
<td>9: PRISM</td>
<td>16076/10714</td>
<td>22/11</td>
</tr>
</tbody>
</table>

Table 1: training/test data sets

For performance evaluation several parameters were evaluated:

- **Number of used features**: By using the packet size information in addition to the ACK delay time information, the feature space can easily exceed reasonable boundaries. Thus, the ACK delay and packet size information needs to be grouped. This is done by specifying the number of delay partitions and the number of size partitions. A short example explains this grouping: If the packet size from 1 byte to 1600 bytes and the delay time from 70 ms to 500 ms are taken into consideration, the size of the feature space would be $1600 \times 430 = 688000$. By using 40 partitions for the delay time and 40 partitions for the packet size, this size can be lowered to $40 \times 40 = 1600$ features. The grouping is done by mapping 1 byte to 40 bytes to the first feature, 41 bytes to 80 bytes to the second feature and so on. The same procedure is applied to the ACK delay information.

- **Time/packet size range**: These parameters set the range of ACK time delay and packet size which is used for feature generation. For the evaluation of WiFinger we used a range of 70 ms to 500 ms for the delay information and 1 byte to 1600 bytes for the size information.

- **Histogram size**: This parameter is used to set the number of packets which are used to create a histogram. For our tests we evaluated a setting of 50 packets per histogram.

- **Session size**: This parameter is used to create sessions from the test data sets. The evaluation of different sessions sizes gives information about how many packets need to be analyzed before an accurate classification can be made.

- **Training factor**: This factor is used to separate the whole data set into training and test data. We used a setting of 0.6 for all tests.

The classification results were obtained in this way:

- Data was collected with WiFinger for seven different chipsets.
- The parameters described above were tuned to evaluate the impact of delay and packet size features.
- The overall classification results were obtained by getting the number of correctly classified time/size histograms for the whole test dataset.
The dataset was split into sessions with 500 and 1000 packets. This should give an indication on how many packets are needed in order to get an accurate classification result.

The combination of delay and packet size features which gave the best overall results was evaluated with sessions 500 and 1000 packets.

The first row of Table 2 shows the results when using delay information only. Tests from row 2 - 5 evaluate the performance of different feature sets and indicate that adding packet size information can significantly increase the classification performance. In case of 40 time slots and 40 packet size slots 80% of all histograms were classified correctly, which is an increase in classification accuracy of 64% over the first version, which only classifies 51.4% of all histograms correctly. These parameters result in a feature vector with 1600 entries, which is quite large. However, row 2 shows, that the classification performance only drops slightly when using just 10 features for packet size. The feature vector size is also reduced to 25% (400 instead of 1600), which is even lower than the feature vector size used in row 1, where only delay information is taken into consideration.

It is necessary to be careful with the number of features used for the packet size. As real data is used for SOM training, it is not guaranteed that this data has an equal distribution of packet sizes over all chipsets. Thus, by using a too fine resolution for the packet size (meaning a large feature space), the algorithm learns to classify the chipsets according to the packet size.

The parameters which gave the best classification accuracy (row 5) were used to rerun the experiment with 500/1000 packets per session. These sessions were created by using data from the test sets. The results of this evaluation can be seen in Table 3 and 4.

The following conclusions can be drawn from the results:

- There is only a very small performance increase if 1000 packets instead of 500 are used per session. The session size needed for an accurate classification result depends on the type of analyzed data. Generally, increasing the session size increases classification accuracy as noise is reduced.
- Most of the classification errors are made, when chipsets are classified as chipset 7 (PRISM 3). It seems that this chipset is quite similar to the other ones tested. Furthermore, the training set for chipset 7 was rather small. As the classification is based on the hits on the SOM, noise plays a larger role when smaller training sets are used.

Table 3: Confusion matrix for 500 packet sessions. E.g. 89.4 % of chipset 1 (in row 1) sessions are classified correctly as chipset 1, 0 % as chipset 2, 2.1 % as chipset 3, etc.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>89.4</td>
<td>0</td>
<td>2.1</td>
<td>8.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>9.9</td>
<td>16.9</td>
<td>0.0</td>
<td>1.4</td>
<td>5.6</td>
<td>28.2</td>
<td>38.0</td>
</tr>
<tr>
<td>3</td>
<td>0.0</td>
<td>0.0</td>
<td>93.6</td>
<td>0.0</td>
<td>6.4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>34.6</td>
<td>0</td>
<td>0</td>
<td>65.4</td>
</tr>
<tr>
<td>5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
<td>100.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0.0</td>
<td>3.6</td>
<td>0.0</td>
<td>0</td>
<td>0</td>
<td>82.1</td>
<td>14.3</td>
</tr>
<tr>
<td>7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

Table 4: Confusion matrix for 1000 packet sessions. E.g. 95.8 % of chipset 1 (in row 1) sessions are classified correctly as chipset 1, 0 % as chipset 2, 4.1 % as chipset 3, etc.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>95.8</td>
<td>0</td>
<td>4.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>11.1</td>
<td>13.9</td>
<td>0</td>
<td>0</td>
<td>5.6</td>
<td>22.2</td>
<td>47.2</td>
</tr>
<tr>
<td>3</td>
<td>0.0</td>
<td>0.0</td>
<td>95.4</td>
<td>0</td>
<td>6.6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>30.8</td>
<td>0</td>
<td>0</td>
<td>69.2</td>
</tr>
<tr>
<td>5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
<td>85.7</td>
<td>14.3</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

7 Conclusion and Future Work

This article describes the possibility of chipset fingerprinting based on timing characteristics in wireless networks in IEEE 802.11 standards compliance. The presented approach is absolutely passive and thus not detectable by the fingerprintee or any other party. All measurements are carried out by off-the-shelf low-cost hardware. The approach is resistant against any kind of standardized cryptographic routines like WEP and WPA due to the fact that management frames are not encrypted by these standards. As a proof of concept we implemented a tool called WiFinger to validate our approach in real world scenarios and obtained promising results in identifying single devices and chipsets. This paper further on describes the basics and application of the applied classification method based on Self Organizing Maps (SOM).

Some improvements are still possible. Beside the optimization of the feature selection for the ACK delay classification, we intend to add an implementation of the fingerprinting approach proposed by Jason Franklin et al. (McCoy et al. 2006) (see section 2 for further details) to our software. We plan to integrate the SOM training algorithm into the WiFinger tool which would eliminate the time consuming process of exporting/importing the data from/into Matlab®. This integration allows for a better evaluation by using larger training/test sets with a larger number of different chipsets and will help to explain why several of the chipsets cannot be classified with high accuracy. The next step will then be to use openWRT based IEEE 802.11 compatible access points for data collection and a centralized analysis and classification server for network wide WLAN MAC spoofing detection. This approach would allow the usage of existing WLAN infrastructure to apply our method.

We further on need to validate if it is possible to classify different devices with the same chipset. Hardware properties like clock skew could render a timing offset that affects the Significant Histograms and al-
Table 2: Comparing the impact of different features. The results show that the delay information is more important than the packet size. However, adding packet size information increases the classification accuracy.

Plummer, D. C. (1982), ‘Rfc 862 - an ethernet address resolution protocol or converting network protocol addresses to 48.bit ethernet address for transmission on ethernet hardware’.  
**URL:** http://tools.ietf.org/html/rfc826


A Modal Logic For Information System Security

Yun Bai

1 School of Computing and Mathematics
University of Western Sydney
Locked Bag 1797, Penrith South DC
NSW 1797, Australia
Email: ybai@scm.uws.edu.au

Khaled M. Khan

2 Department of Computer Science and Engineering
Qatar University
Qatar
Email: k.khan@qu.edu.qa

Abstract

As a security mechanism, authorization or access control ensures that all accesses to the system resources occur exclusively according to the access polices and rules specified by the system security agent. Authorization specification has been extensively studied and a variety of approaches have been investigated. In this paper, we propose a knowledge oriented formal language to specify the system security policies and their reasoning in response to system resource access request. The semantics of our language is provided by translating our language into epistemic logic program in which knowledge related modal operators are employed to represent agents' knowledge in reasoning. We demonstrate how our authorization language handles the situation where the security agent's knowledge on access decision is incomplete.

Keywords: Access Control, Intelligent Systems, Formal Language, Authorization

1 Introduction

One important mechanism to protect information system is to control the access to the system resources. Authorization or access control is such a mechanism. It is to ensure that all accesses to the system resources occur exclusively according to the access polices and rules specified by the security agent of the information system. Authorizations or access control has been extensively studied in (Athuri et al. 2002), (Chomicki et al. 2000), (Fernandez et al. 1995), (Zhou et al. 2008) etc. and a variety of authorization specification approaches such as access matrix (Dacier et al. 1994), (Denning 1976), role-based access control (Crampton et al. 2008), access control in database systems (Bertino et al. 1996), authorization delegation (Murray et al. 2008), procedural and logical specifications (Bai et al. 2003), (Bertino et al. 2003) have been investigated. Since logic based approaches provide a powerful expressiveness (Fagin et al. 1995) as well as flexibility for capturing a variety of system security requirements, increasing work has been focusing on this aspect. Jajodia et al (Jajodia et al. 2001) proposed a logic language for expressing authorizations. They used predicates and rules to specify the authorizations; their work mainly emphasizes the representation and evaluation of authorizations. The work of Bertino et al (Bertino et al. 2000) describes an authorization mechanism based on a logic formalism. It mainly investigates the access control rules and their derivations. In their recent work (Bertino et al. 2003), a formal approach based on C-Datalog language is presented for reasoning about access control models.

Nevertheless, there were some limitations so far in these approaches. For instance, when the security agent does not have complete, specific information about the security domain, how to reason and answer access queries under such a scenario? For example, the agent currently does not know clearly who can access the classified file between Alice and Bob, but knows only one of them can. This can be specified by a disjunctive logic program (Baral 2003) as follows:

\[
\text{AliceCanAccessFile} \lor \text{BobCanAccessFile} \leftarrow , \]
\[
\text{AliceCanAccessFile} \leftarrow \neg \text{BobCanAccessFile}, \]
\[
\text{BobCanAccessFile} \leftarrow \neg \text{AliceCanAccessFile}. \]

If a query asks if Alice can read the classified file, the agent will not be able to make the decision, because this program has two different answer sets: \{AliceCanAccessFile\} and \{BobCanAccessFile\}. In fact, under many circumstances, using disjunctive logic programming to specify security policies is not sufficient to precisely handle incomplete information.

In this paper, we propose a knowledge based formal languages \(L^k\) to specify authorization domain with incomplete information in secure computer systems. We introduce modal logic to specify and reason about a security domain then translate the domain into an epistemic logic program. We show that our approach has an expressive power to describe a variety of complex security scenarios.

In our presentation, we assume the existence of a single, local system security officer or security agent administering the authorizations. This assumption enables us to concentrate on a single administering agent system and hence avoids the problem of coordination among multi agents. Access control in a multi-security system or administered by multi-security agents is out of the scope of this paper.

The rest of the paper is organized as follows. Sec-
The syntax of a high level language $L^k$ by outlining its syntax and gives some authorization policy examples specified by the language. Section 3 explains the semantics of language $L^k$. We start by introducing a general overview of epistemic logic program, then map the domain description specified by $L^k$ into the logic program, and we give some examples to show the process and the domain description and its corresponding logic program. In section 4, we present a case study to demonstrate the reasoning of our system. Section 5 outlines the implementation issue. Section 6 concludes the paper with some remarks.

2 The syntax of a high level language $L^k$

In this section we define the basic syntax of a high level language $L^k$ which embeds a modal operator $K$ to represent an agent’s knowledge about access control policies.

Language $L^k$ includes the following disjoint sorts for subject $S$, group-subject $(G, G_1, G_2, \cdots)$ for the constants and $g, g_1, g_2, \cdots$ for the variables), group-access-right $(A, A_1, A_2, \cdots)$ for the constants and $a, a_1, a_2, \cdots$ for the variables), group-access-right $(GA, GA_1, GA_2, \cdots)$ for the constants and $ga, ga_1, ga_2, \cdots$ for the variables), group-access-right $(O, O_1, O_2, \cdots)$ for the constants and $o, o_1, o_2, \cdots$ for the variables), object $(GO, GO_1, GO_2, \cdots)$ for the constants and $go, go_1, go_2, \cdots$ for the variables) together with predicate symbols holds which takes arguments as subject or group-subject, access-right or group-access-right and object or group-object respectively; $\&$ which takes arguments as subject and group-subject or access-right and group-access-right or object and group-object respectively; $\subseteq$ which both arguments are group-subjects, group-access-rights or group-objects and logic connectives $\land$ and $\neg$. We also introduce a modal operator $K$ to represent what an agent knows to be true. We use $K$-hold$(S, R, O, F)$ to represent that “it’s believed that Sue can read File”. In $L^k$, a subject $S$ has read right to an object $F$ if $\text{read}(S, F)$. It defines an atomic formula of the language. We define a fact $f$ to be an atomic formula or its negation. A ground fact is a fact without variable occurrence. We view $\neg f$ as $f$. A fact expression $\phi$ of $L^k$ is defined as follows: (i) each fact $\phi$ is a fact expression; (ii) if $\phi$ and $\psi$ are fact expressions, then $\phi \land \psi$ and $\phi \lor \psi$ are also fact expressions. A ground fact expression is a fact expression without variable occurrence. A ground fact expression is called a ground instance of a fact expression if this ground fact expression is obtained from the fact expression by replacing each of its variable occurrence with the same sort constant. A fact expression $\phi$ is called conjunctive (or disjunctive) if it is of the form $\phi_1 \land \cdots \land \phi_n$ (or $\phi_1 \lor \cdots \lor \phi_n$ respectively), where each $\phi_i$ is a fact.

Now we are ready to formally define the propositions in $L^k$. An initial proposition in $L^k$ is defined as

\[
\text{initially } \phi \tag{1}
\]

where $\phi$ is either a conjunctive or disjunctive fact expression. That is, $\phi$ is of the form $\phi_1 \land \cdots \land \phi_n$ or $\phi_1 \lor \cdots \lor \phi_n$, where each $\phi_i$ is a fact.

An objective proposition is an expression of the form

\[
\phi \text{ if } \psi \text{ with absence } \gamma \tag{2}
\]

where $\phi$ is either a conjunctive or disjunctive fact expression, $\psi$ and $\gamma$ are two conjunctive fact expressions.

A subjective proposition is an expression of the form

\[
\phi \text{ if } \psi \text{ with absence } \gamma \text{ knowing } \beta, \tag{3}
\]

or

\[
\phi \text{ if } \psi \text{ with absence } \gamma \text{ not knowing } \beta. \tag{4}
\]

where $\phi$ is a conjunctive or disjunctive fact expression, and $\psi$, $\gamma$ and $\beta$ are conjunctive fact expressions.

A proposition is called a ground proposition if it does not contain variables. A policy domain description $D$ in $L^k$ is a finite set of initial propositions, objective propositions and subjective propositions.

In the following, we describe a few complex security scenarios using language $L^k$, and demonstrate that $L^k$ is an expressive language to represent incomplete information, default information, and agents’ knowledge in relation to various access control situations.

Example 1 The example mentioned in the introduction can be represented by a domain description:

\[
\begin{align*}
\text{initially} & \quad \text{holds}(Alice, Access, File) \lor \\
& \text{holds}(Bob, Access, File), \\
& \text{holds}(Alice, Access, File) \text{ if } \\
& \neg\text{holds}(Bob, Access, File), \\
& \neg\text{holds}(Alice, Access, File)
\end{align*}
\]

Here the initial fact holds$(Alice, Access, File) \lor$ holds$(Bob, Access, File)$ represents an incomplete information about Alice and Bob’s access right to the file.

Example 2 Consider a domain description $D$ consists of the following propositions:

\[
\begin{align*}
\text{initially} & \quad \text{holds}(S, Own, O), \\
& \text{holds}(S, Write, O) \text{ if } \text{holds}(S, Own, O), \\
& \text{with absence } \neg\text{holds}(S, Write, O), \\
& \neg\text{holds}(S, Own, O) \text{ if } \neg\text{holds}(S, Read, O).
\end{align*}
\]

This domain description expresses the following policies: initially subject $S$ owns object $O$. If there is no evidence that $S$ cannot write on $O$ is absent from the domain, then $S$ has write right on $O$, and $S$ will no longer owns $O$ if somehow $S$ cannot read $O$ anymore. Here with absence $\neg\text{holds}(S, Write, O)$ represents a default information. As long as there is no clear information indicating $\neg\text{holds}(S, Write, O)$, it would be assumed that $S$ can write $O$.

Example 3 Let us look at another example. A policy says that if a subject group $G$ can read file $F$, then a member $S_1$ of $G$ will be assumed to be able to read $F$, as well if we don’t know that $S_1$ cannot read $F$. This can be specified by the following propositions:

\[
\begin{align*}
\text{initially} & \quad \text{holds}(G, Read, F), \\
\text{initially} & \quad S_1 \in G, \\
& \text{holds}(S_1, Read, F) \text{ if } \\
& \text{holds}(G, Read, F), S_1 \in G, \\
& \text{not knowing } \neg\text{holds}(S_1, Read, F)
\end{align*}
\]

This example represents a policy involving agent (subject)’s knowledge for making decision. As we will show next, the semantics of knowledge in $L^k$ will be defined based on epistemic logic programming.
3 Semantics of $\mathcal{L}^k$

Given a domain description $D$, we will translate it into an epistemic logic program $\Pi(D)$, then the semantics of $D$ will be defined based on the world view semantics of program $\Pi(D)$.

In the following, we first introduce epistemic logic programs, and then define the semantics of $\mathcal{L}^k$.

3.1 Epistemic logic programs

In this section, we present a general overview on epistemic logic programs. Gelfond extended the syntax and semantics of disjunctive logic programs to allow the correct representation of incomplete information in the presence of multiple extensions (Gelfond 1994).

In epistemic logic programs, the language of (disjunctive) extended logic programs is expanded with two modal operators $K$ and $M$. $KF$ is read as ”$F$ is known to be true” and $MF$ is read as ”$F$ may be believed to be true”. In this paper we will consider propositional epistemic logic programs where rules containing variables are viewed as the set of all ground rules by replacing these variables with all constants occurring in the language. The semantics for epistemic logic programs is defined by the pair $(\mathcal{A}, W)$, where $\mathcal{A}$ is a collection of sets of ground literals which is also simply called is a collection of belief sets, and $W$ is a set in $\mathcal{A}$ called the agent’s working set of beliefs. The truth of a formula $F$ in $(\mathcal{A}, W)$ is noted by $(\mathcal{A}, W) \models F$ and the falsity is denoted by $(\mathcal{A}, W) \not\models F$. They are defined as follows.

$$(\mathcal{A}, W) \models p \text{ if } p \in W \text{ where } p \text{ is a propositional atom.}$$

$$(\mathcal{A}, W) \models KF \text{ if } (\mathcal{A}, W_i) \models F \text{ for all } W_i \in \mathcal{A}.$$  

$$(\mathcal{A}, W) \models MF \text{ if } (\mathcal{A}, W_i) \models F \text{ for some } W_i \in \mathcal{A}.$$  

$$(\mathcal{A}, W) \models F \land G \text{ iff } (\mathcal{A}, W) \models F \text{ and } (\mathcal{A}, W) \models G. $$  

$$(\mathcal{A}, W) \models F \lor G \text{ iff } (\mathcal{A}, W) \models F \text{ or } (\mathcal{A}, W) \models G.$$  

$$(\mathcal{A}, W) \models \neg F \text{ iff } (\mathcal{A}, W) \not\models F.$$  

$$(\mathcal{A}, W) \models F \rightarrow G \text{ iff } (\mathcal{A}, W) \models G \text{ or } (\mathcal{A}, W) \not\models F.$$  

$$(\mathcal{A}, W) \models G \text{ or } (\mathcal{A}, W) \not\models F.$$  

It is worth mentioning that since belief set $W$ allows both positive and negative propositional atoms, in Gelfond’s semantics, $(\mathcal{A}, W) \models \varphi$ is not equivalent to $(\mathcal{A}, W) \not\models \varphi$ in general. For instance, $\{\{a,b\}\}, \{a,b\} \not\models c$, but we do not have $\{\{a,b\}\}, \{a,b\} \models c$ (i.e. $\{\{a,b\}\}, \{a,b\} \not\models \neg c$).

Consequently, here $K$ and $M$ are not dual modal operators here$^2$. Consider $\mathcal{A} = \{\{a,b\}, \{a,b,\neg c\}\}$. Clearly we have $\mathcal{A} \models \neg Kc$. But having $\mathcal{A} \models Mc$ seems to be wrong.

If a formula $G$ is of the form $KF$, $\neg KF$, $MF$, or $\neg MF$ (where $F$ is a propositional formula), then its truth value in $(\mathcal{A}, W)$ will not depend on $W$. In this case we call $G$ a subjective formula. If $F$ is a propositional literal, then we call $KF$, $\neg KF$, $MF$, and $\neg MF$ subjective literals. On the other hand, if $G$ does not contain $K$ or $M$, then its truth value in $(\mathcal{A}, W)$ will only depend on $W$ and we call $G$ an objective formula or objective literal if $G$ is a propositional literal. In the case that $G$ is subjective, we simply write $A \models G$ instead of $(\mathcal{A}, W) \models G$, and $W \models G$ instead of $(\mathcal{A}, W) \models G$ in the case that $G$ is objective.

In general, we simply write $A \models G$ if for each $W \in A$, we have $(\mathcal{A}, W) \models G$. Each $\mathcal{A}$

An epistemic logic program $\Pi$ is a finite set of rules of the form:

$$F \leftarrow G_1, \ldots, G_m, not \text{ } G_{m+1}, \ldots, not \text{ } G_n. \ (5)$$

In (5), $m, n \geq 0$, $F$ is of the form $F_1 \text{ or } \cdots \text{ or } F_k$ ($k \geq 1$) and $F_1, \ldots, F_k$ are objective literals, $G_1, \ldots, G_m$ are objective or subjective literals, and $G_{m+1}, \ldots, G_n$ are objective literals. For an epistemic logic program $F$, its semantics is given by its world view which is defined in the following steps:

Step 1. Let $\Pi$ be an epistemic logic program not containing modal operators $K$ and $M$ and negation as failure not. A set $W$ of ground literals is called a belief set of $\Pi$ iff $W$ is a minimal set of satisfying conditions: (i) for each rule $F \leftarrow G_1, \ldots, G_m$ from $\Pi$ such that $W \models G_1 \land \cdots \land G_m$, we have $W \models F$; (ii) if $W$ contains a pair of complementary literals then $W = \text{Lit}$, i.e. $W$ is an inconsistent belief set$^3$.

Step 2. Let $\Pi$ be an epistemic logic program not containing modal operators $K$ and $M$ and $W$ be a set of ground literals in the language of $\Pi$. By $\Pi_W$ we denote the result of (i) removing from $\Pi$ all the rules containing formulas of the form not $G$ such that $W \models G$ and (ii) removing from the rules in $\Pi$ all other occurrences of formulas of the form not $G$.

Step 3. Finally, let $\Pi$ be an arbitrary epistemic logic program and $A$ a collection of sets of ground literals in its language. By $\Pi_A$ we denote the epistemic logic program obtained from $\Pi$ by (i) removing from $\Pi$ all rules containing formulas of the form $\neg G$ such that $G$ is subjective and $A \models \neg G$, and (ii) removing from rules in $\Pi$ all other occurrences of subjective formulas. Now we define that a collection $A$ of sets of ground literals is a world view of $\Pi$ if $A$ is the collection of all belief sets of $\Pi_A$.

Example 4 Consider a simple epistemic logic program $\Pi$ consisting of the following rules:

$$a \lor b \leftarrow \neg Mc,$$

$$d \leftarrow Ka,$$

$$e \leftarrow b, not \ e.$$  

Let $A = \{(a,d)\}$, then from the above definition, we have its belief sets $\Pi_A$:

$$a \lor b \leftarrow \neg Mc,$$

$$d \leftarrow \neg Ka,$$

$$e \leftarrow b, not \ e.$$  

Then it is easy to see that $\{a,d\}$ is the only answer set of $\Pi_A$. So $A$ is a world view of $\Pi$. It can be also verify that $A$ is the unique world view of $\Pi$.

3.2 Translating a domain description into an epistemic logic program

Now we define the semantics of $\mathcal{L}^k$ based on the world view semantics of epistemic logic programs. Let $D$ be

---

$^1$We denote $(\mathcal{A}, W) \not\models \varphi$ if $(\mathcal{A}, W) \not\models \varphi$ does not hold.

$^2$K and M are called dual if $\neg K \varphi$ is logically equivalent to $M \varphi$.

$^3$Note that in our context, a belief set is simply a set of ground literals. Here a belief set of a program is a belief set that satisfies the conditions (i) and (ii).
a given domain description of $\mathcal{L}^k$, i.e. $D$ is a finite set of propositions as illustrated in section 2.1. We specify an epistemic logic program $\Pi(D)$ translated from $D$ as follows:

1. For an initial policy proposition (1): initially $\phi$, if $\phi$ is a conjunctive fact expression $\phi_1 \land \ldots \land \phi_n$, then it is translated to a set of rules$^4$:

   $\phi_1 \leftarrow$, 
   $\ldots$, 
   $\phi_n \leftarrow$, 

   if $\phi$ is a disjunctive fact expression $\phi_1 \lor \ldots \lor \phi_n$, then it is translated to one rule:

   $\phi_1 \lor \ldots \lor \phi_n \leftarrow$.

2. For each objective access proposition (2): $\psi$ with absence $\gamma$, here $\psi = \psi_1 \land \ldots \land \psi_k$ and $\gamma = \gamma_1 \land \ldots \land \gamma_l$, if $\phi$ is a conjunctive fact expression $\phi_1 \land \ldots \land \phi_n$, then it is translated to a set of rules:

   $\phi_1 \leftarrow \psi_1, \ldots, \psi_k, \neg \gamma_1, \ldots, \neg \gamma_l$, 
   $\ldots$, 
   $\phi_n \leftarrow \psi_1, \ldots, \psi_k, \neg \gamma_1, \ldots, \neg \gamma_l$,

   if $\phi$ is a conjunctive fact expression $\phi_1 \land \ldots \land \phi_n$, then it is translated to one rule:

   $\phi_1 \land \ldots \land \phi_n \leftarrow \psi_1, \ldots, \psi_k$, 
   $\neg \gamma_1, \ldots, \neg \gamma_l$.

3. For each subjective access proposition (3): $\phi$ with absence $\gamma$ knowing $\beta$, where $\psi = \psi_1 \land \ldots \land \psi_k$, $\gamma = \gamma_1 \land \ldots \land \gamma_l$, and $\beta = \beta_1 \land \ldots \land \beta_r$, if $\phi$ is a conjunctive fact expression $\phi_1 \land \ldots \land \phi_n$, then translate it to a set of rules:

   $\phi_1 \leftarrow \psi_1, \ldots, \psi_k$, 
   $K\beta_1, \ldots, K\beta_r, \neg \gamma_1, \ldots, \neg \gamma_l$, 
   $\ldots$, 
   $\phi_n \leftarrow \psi_1, \ldots, \psi_k$, 
   $K\beta_1, \ldots, K\beta_r, \neg \gamma_1, \ldots, \neg \gamma_l$,

   if $\phi$ is a disjunctive fact expression $\phi_1 \lor \ldots \lor \phi_n$, then translate it to one rule:

   $\phi_1 \lor \ldots \lor \phi_n \leftarrow \psi_1, \ldots, \psi_k$, 
   $K\beta_1, \ldots, K\beta_r, \neg \gamma_1, \ldots, \neg \gamma_l$.

4. For each subjective access proposition (4): $\phi$ if $\psi$ with absence $\gamma$ not knowing $\beta$, where $\psi = \psi_1 \land \ldots \land \psi_k$, $\gamma = \gamma_1 \land \ldots \land \gamma_l$, and $\beta = \beta_1 \land \ldots \land \beta_r$, if $\phi$ is a conjunctive fact expression $\phi_1 \land \ldots \land \phi_n$, then translate it to a set of rules:

   $\phi_1 \leftarrow \psi_1, \ldots, \psi_k$, 
   $\neg K\beta_1, \ldots, \neg K\beta_r, \neg \gamma_1, \ldots$, 
   $\not \beta_r$, 
   $\ldots$, 
   $\phi_n \leftarrow \psi_1, \ldots, \psi_k$, 
   $\neg K\beta_1, \ldots, \neg K\beta_r, \neg \gamma_1, \ldots$, 
   $\not \beta_r$,

   if $\phi$ is a disjunctive fact expression $\phi_1 \lor \ldots \lor \phi_n$, then translate it to one rule:

   $\phi_1 \lor \ldots \lor \phi_n \leftarrow \psi_1, \ldots, \psi_k$, 
   $\neg K\beta_1, \ldots, \neg K\beta_r, \neg \gamma_1, \ldots, \not \beta_r$.

Now we specify $\Pi(D)$ to be the collection of all rules translated from $D$ by the above procedure. It is noted that $\Pi(D)$ is an epistemic logic program without modal operator $M$.

Since positions in $D$ may contain variables, program $\Pi(D)$ may also contain variables. In this case, a ground epistemic logic program generated from $\Pi(D)$ by replacing each variable with all possible corresponding sort constants occurring in $\Pi(D)$. Without much confusion, we may still use notion $D(\Pi)$ to denote this corresponding ground program.

**Definition 1** Let $D$ be a domain description of $\mathcal{L}^k$, $\Pi(D)$ the epistemic logic program translated from $D$ as described above, and $f$ a ground fact. We say that $D$ entails $f$, denoted as $D \models f$, if $\Pi(D)$ has a world-view, and for each world-view $A$ of $\Pi(D)$, $A \models f$.

**Example 5** Consider Example 3 presented in section 2. According to the above procedure, we can translate the domain description $D$ as the following program $\Pi(D)$:

$$\text{holds}(G, \text{Read}, F) \leftarrow,$$

$$\text{s1} \in G \leftarrow,$$

$$\text{holds}(\text{s1}, \text{Read}, F) \leftarrow \text{holds}(G, \text{Read}, F),$$

$$\text{s1} \in G, \neg K\neg \text{holds}(\text{s1}, \text{Read}, F).$$

Now suppose we need to answer a query whether $\text{s1}$ can read file $F$, i.e. whether $D \models \text{holds}(\text{s1}, \text{Read}, F)$. It is not difficult to see that program $\Pi(D)$ has a unique world-view $A = \{\text{holds}(G, \text{Read}, F), \text{s1} \in G, \text{holds}(\text{s1}, \text{Read}, F)\}$, and $A \models \text{holds}(\text{s1}, \text{Read}, F)$. So we conclude that $D \models \text{holds}(\text{s1}, \text{Read}, F)$.

4 **A case study: Reasoning about knowledge in access control**

In this section, we demonstrate a case study from which we show that our approach can overcome some difficulties in the reasoning about access control when incomplete information is involved.

We consider a typical hospital scenario that doctor assistants take responsibility to manage patients' files and access relevant files and data from other departments. In order to ensure the confidentiality of all patients’ medical records, a number of authorization policies must be implemented in all departments in a hospital.

Suppose that Hobson is a heart specialist in a hospital. He is planning a by-pass surgery for his patient John, for that purpose, he needs to review John’s all other recent medical records before the surgery. Alice and Sue are the personal assistants of Hobson. Each of them can access doctor Hobson’s all patients’ records, while Sue also takes responsibility to request patients’ medical records from other departments in the hospital.

By using our language $\mathcal{L}^k$, we first formalize the general authorization policies across the hospital as follows:

$$\text{holds}(x, \text{Read, All\_heart\_records}) \text{ if knowing assistant}(x, \text{Hobson}),$$

$$\text{holds}(x, \text{Read, y\_heart\_record}) \text{ if }$$

$$\text{holds}(x, \text{Read, All\_heart\_records}) \land \text{patient}(y, \text{Hobson}),$$

$$\text{holds}(\text{Hobson, Read, y}) \text{ if } \text{holds}(x, \text{Read, y}) \land$$

$$\text{assistant}(x, \text{Hobson}) \land \text{patient}(y, \text{Hobson}).$$
sendRequest(Hobson, y) if request(Hobson, y) with absence

sendRequest(Alice, Read, y) if request(Hobson, y) with absence sendRequest(Sue, Read, y). (10)

waitingApproval(x, Read, y) if sendRequest(x, Read, y) ∧

not knowing approved(x, Read, y), (11)

approved(x, Read, y) if sendRequest(x, Read, y) ∧

assistant(x, d) ∧ specialist(d), (12)

holds(x, Read, y) if approved(x, Read, y). (13)

Let us take a closer look at these rules. Basically, rules (6) and (7) say that if it is known that x is a personal assistant of Doctor Hobson, then x can access (read) Doctor’s all patients’ heart records, and if someone is already permitted to read all patients’ heart records, and y is a patient of Doctor, then this person can also read y’s heart record. Note that rule (7) plays a role of inheritance for access control. Also, rule (8) implies the fact that once Doctor Hobson’s assistant x obtains the access read for some patient record from other department, then Doctor Hobson should have the access right on this record obviously.

Rule (9) indicates that if Doctor Hobson has a request of accessing patient y’s record from other departments, then usually Sue should send this request for approval. Note that this rule is defeasible due to with absence. For instance, if Sue is on leave, then ¬ sendRequest(Sue, Read, y) will be presented and hence this rule will not be initiated any more. Rule (10) describes the case that Alice will do Sue’s duty when she is not available. On the other hand, rule (11) means that once a request is sent out, it is on the waiting status if no approval from that department is explicitly informed. Rule (12) states that the corresponding department will approve the request sent by x about y’s record if x is a personal assistant of some doctor d who is a registered specialist of the hospital. Finally, rule (13) is quite straightforward that if x receives the approval of the department that holds patient record, x can then access y’s record in that department.

Now suppose we have the following facts:

initially assistant(Alice, Hobson), (14)

initially assistant(Sue, Hobson), (15)

initially patient(John, Hobson), (16)

initially specialist(Hobson), (17)

initially request(Hobson, John_generalHealth_record), (18)

initially ¬ sendRequest(Sue, Read, John_generalHealth_record), (19)

we would like to know how the access right “Read” for patient John’s general health record can be obtained by Doctor Hobson. Let D be the domain description consisting of propositions (6) - (19). Then applying our translation procedure described in section 3.2, we can obtain the following epistemic logic program II(D):

holds(x, Read, All_heart_records) ←

Kassistant(x, Hobson),

holds(x, Read, y, heart_record) ←

holds(x, Read, All_heart_records),

patient(y, Hobson),

holds(Hobson, Read, y) ←

holds(x, Read, y, assistant(x, Hobson),

sendRequest(Sue, Read, y) ←

request(Hobson, y),

not ¬sendRequest(Sue, Read, y),

sendRequest(Alice, Read, y) ←

request(Hobson, y),

not sendRequest(Sue, Read, y),

waitingApproval(x, Read, y) ←

sendRequest(x, Read, y),

¬approved(x, Read, y),

approved(x, Read, y) ←

sendRequest(x, Read, y), assistant(x, d),

specialist(d),

holds(x, Read, y) ← approved(x, Read, y),

assistant(Alice, Hobson) ←,

assistant(Sue, Hobson) ←,

patient(John, Hobson) ←,

specialist(Hobson, H) ←,

request(Hobson, John_generalHealth_record) ←,

sendRequest(Sue, Read, John_generalHealth_record) ←.

It is easy to see that II(D) has a unique world view A:

{ assistant(Alice, Hobson), assistant(Sue, Hobson), patient(John, Hobson), specialist(Hobson), request(Hobson, John_generalHealth_record), ¬sendRequest(Sue, Read, John_generalHealth_record) sendRequest(Alice, Read, John_generalHealth_record),

holds(Alice, Read, All_heart_records),

holds(Sue, Read, All_heart_records),

holds(Alice, Read, John_heart_records),

holds(Sue, Read, John_heart_records),

approved(Alice, Read, John_heart_records),

holds(Alice, Read, John_generalHealth_record) holds(Alice, Read, John_generalHealth_record) }.

From A we can finally derive that the following results:

D ⊨ sendRequest(Alice, Read, John_generalHealth_record),

D ⊨ approved(Alice, Read, John_generalHealth_record),

D ⊨ holds(Alice, Read, John_generalHealth_record),

D ⊨ holds(Hobson, Read, John_generalHealth_record).

5 The implementation issues

A system for epistemic logic programming has been implemented. In this section we briefly outlined our implementation of our epistemic logic programming system and explain how our formal language $\mathcal{L}^k$ developed in this paper is fulfilled by the system.

The system we implemented is called World Views Solver, simply denoted as Wviews. The essential function of Wviews is to compute one or all world views (models) of an input epistemic logic program. To
compute the world views of an epistemic logic program II, Wviews first performs a reduction to transform II into a traditional disjunctive logic program (DLP), then call dlv to compute the answer sets of II^A. The system structure is outlined as follows\(^5\).

\[\text{Wviews system structure.}\]

As we mentioned in section 3, the semantics of \(L^k\) is defined in terms of the world view semantics of epistemic logic programs. Having system Wviews, we can easily implement our policy language \(L^k\) in the following way: taking the domain description \(D\) as the input, which is a finite set of \(L^k\) propositions (see section 2), we implement a transformation procedure as illustrated in section 3, to translate \(D\) into an epistemic logic program II\((D)\), then by calling system Wviews, we will be able to compute one or all world views (models) of II\((D)\).

6 Conclusion

In this paper, we proposed a formal language \(L^k\) to specify security policies by an authorization domain with incomplete information. Different from previous policy specification languages, our formal language \(L^k\) has knowledge as its key feature to deal with incomplete domains. We specified the semantics of such knowledge oriented authorization specification language based on the well known the world view semantics of epistemic logic programs. The examples showed demonstrated that our approach has a rich expressive power to describe a variety of complex security requirements. Related semantic and computational properties of epistemic logic programs have been studied in (Zhang 2007), which will be help us to fully using the expressive power of epistemic logic programming to represent and reason about knowledge based authorization policies. This is our current research focus.

References


Baral, C. (2003), Knowledge Representation, Reasoning, and Declarative Problem Solving, MIT Press.


---

\(^5\)The system details can be accessed from http://www.scm.uws.edu.au/ yan/Wviews.html.
Detection of Fast Flux Service Networks

Scott Campbell  Stephen Chan  Jason R. Lee

Lawrence Berkeley National Laboratory
National Energy Research Scientific Computing Center
One Cyclotron Rd
Berkeley, CA 94720
Email: {scampbell, sychan, jrlee}@lbl.gov

Abstract

Fast Flux Service Networks (FFSN) apply high availability server techniques to the business of malware distribution. FFSNs are similar to commercial content distribution networks (CDN), such as Akamai, in terms of size, scope, and business model, serving as an outsourced content delivery service for clients. Using an analysis of DNS traffic, we derive a sequential hypothesis-testing algorithm based entirely on traffic characteristics and dynamic white listing to provide real time detection of FFSNs in live traffic. We improve on existing work, providing faster and more accurate detection of FFSNs. We also investigate a category of hosts not fully explored in previous detection algorithms, this problem is compounded by constant and significant differentiator between our work and others. The final result is a real time FFSN detector using an analysis of DNS traffic, we derive a sequential hypothesis-testing algorithm based entirely on traffic characteristics and dynamic white listing to provide real time detection of FFSNs in live traffic. We improve on existing work, providing faster and more accurate detection of FFSNs. We also investigate a category of hosts not fully explored in previous detectors - Open Content Distribution Networks (OCDN) that share many of the characteristics of FFSNs.

Keywords: Fast Flux, DNS, CDN

1 Introduction

Maintaining high availability and security is a problem faced by all web professionals. For web criminals, this problem is compounded by constant and determined adversarial pressure by law enforcement. While sites containing legitimate content can rely on well-known techniques such as Round Robin DNS or outsourcing their content to Content Distribution Networks (CDNs), for criminal purposes, these techniques are too easy to trace and remove. However, Internet criminals have an advantage that legitimate web sites do not, a constantly replenished supply of compromised hosts.

Fast Flux Service Networks (FFSN) take advantage of these hosts and the nature of DNS resolution to rotate domain name resolution across a large pool of compromised hosts. Honeypot (2007) has shown that the time to live on resource records tends to be short; often as short as 3 minutes, making it possible for the same client to visit multiple FFSN hosts during a single session. Since the hosts in a FFSN are targets of security countermeasures, availability is maintained by frequently rotating in fresh compromised hosts. The simplest type of Fast Flux involves rotating DNS A records across a pool of addresses, and is known as a Single Flux. A more complex version, called the Double Flux by Honeypot (2007), rotates both NS and A records.

There is tremendous utility in near real time identification of FFSN membership. Since content hosted by these networks tends toward illegal scams and malware, any local connections to such networks is worthy of note. Knowledge of sites that are involved in FFSN membership may be used to identify compromised systems.

Our approach to this problem uses data gathered from our border to drive the selection of an algorithm that differentiates between CDN type traffic and FFSN. In Section 3 we observe differences between normal DNS, CDN and FFSN traffic, and as CDN and FFSN dynamic behavior in order to characterize what we expect a solution should look like. In Section 4 we apply hypothesis testing in the form of TRW for new subnet prefixes for a detection algorithm. We find that TRW alone is insufficient to cleanly differentiate CDN and FFSN (as explained in Section 3) but show that dynamic whitelisting should address this problem. In Section 5 a series of implementations are run in parallel to identify the best granularity for determining network locality among IP addresses and to test the results against real traffic. The final result is a real time FFSN detector using /12 subnets as discrete units with a low false positive rate can operate on a significant volume of traffic in real-time.

2 Related Work

In researching Fast Flux networks, we found a number of related works identifying Fast Flux activity for specific types of content, but none based on systematically analyzing the characteristics of live DNS traffic without relying on outside information sources such as domain registration authorities, except for Caglayan (2009) whom is taking the same approach but a different detection algorithm. This is the most significant differentiator between our work and others along with the actual detection algorithm being designed for use on live traffic. The use of the Top 500 web site list significantly reduces the variation from CDN traffic since only the main site name is being signed for use on live traffic. The use of the Top 500 web site list significantly reduces the variation from CDN traffic since only the main site name is being signed for use on live traffic. The use of the Top 500 web site list significantly reduces the variation from CDN traffic since only the main site name is being signed for use on live traffic.

One of the closest matches to our work on FFSN host identification was by Holz, et al. (2008). In this paper a first order approximation is presented as well as CDN and FFSN dynamic behavior in order to characterize what we expect a solution should look like. In Section 4 we apply hypothesis testing in the form of TRW for new subnet prefixes for a detection algorithm. We find that TRW alone is insufficient to cleanly differentiate CDN and FFSN (as explained in Section 3) but show that dynamic whitelisting should address this problem. In Section 5 a series of implementations are run in parallel to identify the best granularity for determining network locality among IP addresses and to test the results against real traffic. The final result is a real time FFSN detector using /12 subnets as discrete units with a low false positive rate can operate on a significant volume of traffic in real-time.

One of the closest matches to our work on FFSN host identification was by Holz, et al. (2008). In this paper a first order approximation is presented as well as CDN and FFSN dynamic behavior in order to characterize what we expect a solution should look like. In Section 4 we apply hypothesis testing in the form of TRW for new subnet prefixes for a detection algorithm. We find that TRW alone is insufficient to cleanly differentiate CDN and FFSN (as explained in Section 3) but show that dynamic whitelisting should address this problem. In Section 5 a series of implementations are run in parallel to identify the best granularity for determining network locality among IP addresses and to test the results against real traffic. The final result is a real time FFSN detector using /12 subnets as discrete units with a low false positive rate can operate on a significant volume of traffic in real-time.
flux and the use of lists of known good and bad static web pages to generate behavioral traffic.

The work presented in J. Nazario, T. Holz (2008) is strongly derived from Holz, et al. (2008), sharing a number of tests and fundamental assumptions such as short TTL, ASN spread and large IP return volume. They look at uncategorized hosts, so it becomes necessary to introduce a heuristic to distinguish between legitimate hosts and FFSNs. An interesting addition to this detection heuristic is the use of IP geometry (large distance between returned values) as an indicator of possible fluxiness. Significant differences include the use of domain names as atomic units, using spam trap messages as a significant data source and having an ad-hoc detection heuristic.

In the work done by Emanuele et. al (2008) for the FluXOR paper a similar mechanism to Holz, et al. (2008) is implemented with the addition of Domain Registrar information. Many of the same geometric tests are done on individual hostnames (vs. domains in Holz, et al. (2008)). Data for suspect hosts is provided by the analysis of hostile email content, severely limiting the exposure of CDN traffic. It is worth noting that the distinct class of OCDN networks is identified as an interesting source of false positives. The most significant difference between this work and ours is that our system uses raw network traffic as a data source, is designed to have a much shorter detection window, and we do not rely on Domain Registrar supplied information.

The final analytical flux analysis looked at is by Konte et al. (2000), and uses email spam as the source of potential flux domains. Other similarities with previous works include the use of an aggressive active resolver and the use of the Alexa Top 500 web site list for defining non-hostile sites. There seemed to be a greater use of rate of change versus a static geometric interpretation and the implementation of sharing infrastructure. Many of the differences found in previous papers are also here using suspicious sources like spam for flux candidates, then using active scanning to determine their fluxing characteristics.

The description of Fast Flux networking as described by the Honeypot (2007) is probably the single most cited technical description of Fast Flux techniques. While the descriptive nature of the work is still outstanding, it is mostly qualitative rather than quantitative. This is by no means a criticism, but is a differentiation between it and our work.

In SSAC (2008) a nice overview of Fast Flux techniques is provided along with descriptions of miscellaneous motivation and the relation of Fast Flux to Phishing and ‘Domain Tasting’ (domains that are registered, but are canceled during the Add Grace Period). The paper concludes that there is no connection. A significant part of the paper looks at possible mitigation techniques as well as how domain name registration mechanics might be used against Fast Flux network operators.

There is a plethora of ongoing work in this field by Caglayan (2009), Kevin (2009), Caglayan (2010) and Pawan (2010) that is incorporating many of the ideas presented in this paper in other hybrid ways. We are not saying that the work presented here is the final authority, but just our perspective on the status of this area of work at the time we started our research.

3 Data Analysis

Our initial motivation for working on this problem was to identify the existence and membership of FFSNs, and to be able to identify communication with any member host. To do this we started with a large corpus of raw DNS traffic and attempted to observe and characterize the behavior of both hostile and non-hostile transactions. Measurements of aggregate address and A-records characteristics in terms of domains, subnets and ASNs lead to a solution, which seems to maximize detection while minimizing the effects of false positives.

For raw data we examine traffic at a large National laboratory over a period of 48 hours and categorize it into three major groupings: normal lookups, CDN lookups and FFSN lookups. CDN traffic was identified using publicly available domain lists, FFSN domains were manually extracted and confirmed based on the total number of IP addresses returned over time and normal traffic was everything else.

3.1 Aggregate Characteristics of A-record Types

The first measurement used is the total number of IP addresses returned over the measurement period for a given A-record. The first time an A-record request is responded to, a record is made of the IP addresses in the response. This record the entire set of IP addresses seen during this request. Additional responses to other requests for the same A-record are compared to this set and if any new addresses exist, the new IP event counter is incremented by one for that A-record entry. In addition, all new IP addresses in the response are added to the A-records base set. For example, if a single DNS request results in 1 new IP address or if it results in 5 new IP addresses, the A-records new IP event counter is only incremented by one. This process is repeated for the duration of the analysis and is similar to what is done in Caglayan (2009).

Table 1: Frequency distribution of new IP address events in A record responses over 48 hours

<table>
<thead>
<tr>
<th>New IP</th>
<th>Normal Net</th>
<th>CDN Net</th>
<th>FFDFN Net</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#</td>
<td>%</td>
<td>#</td>
</tr>
<tr>
<td>0</td>
<td>262130</td>
<td>99.59%</td>
<td>1096</td>
</tr>
<tr>
<td>1</td>
<td>881</td>
<td>0.35%</td>
<td>316</td>
</tr>
<tr>
<td>2</td>
<td>124</td>
<td>0.04%</td>
<td>90</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>0.007%</td>
<td>63</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>0.003%</td>
<td>81</td>
</tr>
<tr>
<td>5-9</td>
<td>22</td>
<td>0.008%</td>
<td>81</td>
</tr>
<tr>
<td>&gt; 10</td>
<td>5</td>
<td>0.002%</td>
<td>3</td>
</tr>
<tr>
<td>Total</td>
<td>263189</td>
<td>100%</td>
<td></td>
</tr>
</tbody>
</table>

Once a new hostname is seen, we begin active lookups, periodically the sensor polls the hostname and the results rolled into the observations for that host. This is not a necessary part of the experiment, but does increase the likelihood of CDN and FFSN detection particularly with slower traffic flows and networks which only return a single IP address for each query. Active lookup is described in some detail in Section 3.5.

Table 1 shows the frequency distribution for new IP event counts. We loosely group this data by the type of network, the number of new address events, and the distribution of counts for new IP events.

From Table 1 we see that the vast majority of traffic does not exhibit any increment of new IP event and can effectively be ignored from a detection standpoint. Conversely, the CDN and FFDFN types represent a tremendous degree of IP diversity. This is the
initial indication that the key to a working algorithm will be identifying FFDNs and creating a fast and accurate method for distinguishing them from CDNs.

For the next step we began decreasing the granularity of the measurements to see if it is possible to take advantage of organizational structures like ASN numbers or subnet masks to increase differences between FFDN and CDN sources. We use the term locality to embody the granularity of these bunched together IP addresses two IP addresses from two different organizations is quite different than two addresses from the same ISP. For results in this section there was little change in quantitative results for different locality values. Because of this, we chose to use the graphs tied to the /12 subnet prefix since it seemed to provide optimal results in flux detection.

Figure 1 shows the same data set used for Table 1 and measures the total count of unique subnets for CDN and FFDN traffic. Both CDN and FFDN graphs grow rapidly over a short (<20,000 sec) time interval, then exhibit different asymptotic behaviors. The difference in the curves is especially striking given that there is an order of magnitude more CDN than FFDN A-records. We interpret this as indicating that the pool of subnets representing CDNs has been significantly explored, while subnets from the FFDN pool remains incompletely explored, because it is much larger, or because new hosts are being added constantly.

Two approaches immediately come to mind for the algorithm. The direct approach would be to count the number of new subnets and identify as FFDN when the count passes a reasonable threshold - this is analogous to the method described in the Holz paper. Another method would be to treat the arrival of each subnet as a decision point in a random walk. For the two sets, you are more likely to see that a returned subnet value is new for a FFDN than a CDN making for a natural pair of hypothesis to test. To see which will provide better results, we need to look at detailed behavior for specific A-records.

3.2 Behavior of Individual A-Records

Since the aggregate behavior in Figure 1 obscures the variation in individual records, we selected representative FFDN and CDNs sample records to get a clearer idea about their dynamics. The FFDNs were chosen as representatives of the different types of subnet distribution counts that we have seen. The CDNs were chosen because they have large subnet ranges and most closely mimic FFDNs (but are quite commonly seen in day to day live traffic).

The grouping seen in the sets of FFDN records is typical and has been observed by Holz, et al. (2008), Emanuele et. al (2008) and (Konte et al. 2000). The most interesting thing from an algorithm design perspective is that there are cases where the FFDN are less aggressive at receiving new subnets in their A-record responses then the CDNs. Here FFDN1 is almost indistinguishable from CDN1, and has a smaller subnet count than any of the other identified FFDN members. CDN2 and CDN3 have larger subnet counts than FFDN1, the key thing here is that any subnet counting algorithm that properly matches FFDN1, will misclassify CDN2 and CDN3 as fast flux. FFDN1 was manually confirmed as a Fast Flux domain selling performance enhancing pharmaceuticals. FFDN1 is not an edge case, since from Table 1, we know that 25 other FFDN operate within a range of 10 or fewer IP addresses, which is over 20% of the total FFDNs examined.

For our mental model of CDN behavior we have a provider with a fairly static set of systems spread across multiple locations. These locations are spread out in terms of geography as well as network topology, to reduce network latency to the customer and to minimize the problems associated with a network or data center outage. However, because the servers are managed systems with costs associated with operating them, there will only be a fixed number of them, and we would expect some clustering of IPs around the networks at datacenters. If the provider selects from all possible servers to avoid network congestion and increase fault tolerance, the responses to DNS requests should be well balanced across this static pool of possibilities. We would expect an initial rapid increase in the number of subnets, and then the rate of new subnets would decrease once most of the subnets have been encountered.

Our model of FFDN behavior is quite different. The set of compromised systems from which a botnet is created is huge, distributed, and constantly gaining and losing members through infection and attrition. The network expands via infection, and may spread into residential broadband networks, business networks or onto vulnerable hosted servers. The botnet operator has little control over when (and which) systems are added and removed from membership. At any given time only a small number of members are exposed to a users query. This window of exposed systems changes rapidly over time as part of behavior to avoid sending requests to systems at risk of being "cleaned up" or blocked by various security countermeasures as noted by Honeypot (2007). From this model, we would expect a similar graph to what we see here - large swaths of near linear activity fol-
owed by a steady reduction in slope. When the set of systems assigned to this activity within the botnet changes, there will be a new spike in new subnets.

From the data set, it is clear that any algorithm based on counting distinct prefixes from A-record lookups will have a tradeoff between a low threshold value ensuring faster detection of FFDNs which will result in a high false positives for popular CDN and a high threshold value providing for lower false positives but resulting in high false negatives and longer wait, perhaps on the order of tens of thousands of seconds, for the algorithm to converge on an answer. Because of this we will be focusing on the random walk option instead.

3.3 Overlap within and between CDN and FFDN address pools

Observing subnet overlap within and between the CDN and FFDN subnets provides another way to distinguish between CDN and FFDN A-records. We expect to see some overlap within each of the data sets. For the CDN traffic dataset we would expect local infrastructure to share some common subnets, while grouping of individual A-records in FFDN analysis has been a common indication of shared infrastructure for some time.

From the dataset described in Table 1, there were 109 discrete subnet prefixes for CDNs, 233 for FFDNs, and 9 shared between them. Bearing in mind there were 2297 CDN and 119 FFDN hostnames, this indicates a tremendous amount of overlap of subnets among the CDN hosts. Graphing the distribution of subnet prefix overlap we see the following:

```
Figure 3: Subnet counts versus time for individual hostnames
```

The horizontal axis here represents the overlap distribution - for example, the first entry for CDNs indicates that for prefixes that were only seen in a single hostname (unique prefix not shared with any other hostnames) there were about 32 entries. At the same time, the long tail on the CDN indicates that there are subnets shared by hundreds, up to just over a thousand A-records. Conversely, a significant proportion of FFDN subnets are unique, with a far smaller tail of shared infrastructure.

The observed distribution hints that using some form of whitelisting for CDN and blacklisting for FFDN address prefixes might be practical. Since we have to assume that both types of distribution networks have host churn i.e. that systems get reallocated according to operational needs implementing these lists in a static way would not be practical.

3.4 Dynamic Black and White Lists

To take advantage of what appears to be architectural differences between subnet overlap in CDN and FFDN networks, we propose using dynamic versions of white and black listing. The behavior of each of these will be quite different and is based on behavior observed in Figure 3.

1. Whitelist: When an A-record is tagged as non-fluxing, all of the subnets associated with it are placed in the whitelist. Any new prefix member associated with it after identification are added to the whitelist as well. Once a prefix is placed on the whitelist, it can't be used as a new prefix in flux calculations, which has shown a significant effect on the TRW calculation. If the prefix is on the blacklist, it is removed.

2. Blacklist: When an A-record is identified as fluxing, all of the subnets associated with it are placed on the blacklist. Any new prefix member associated with it is added to the blacklist as well. The effect of being on the blacklist is that your contribution to the probability ratio calculation is 25% greater than if not.

For both black and white lists, the effect on the lambda (probability ratio) calculation only happens once per A-record per list member.

For the whitelist we exploit two factors. First a small number of prefixes seem to be present in a significant percent of CDN domains. If one of these prefixes gets placed on the whitelist (which will happen given the behavior of the majority of CDN members), it will poison the ability for other CDN related A-records to see enough new prefixes to be marked as fluxing. The second factor is the limitation provided by the shrinking number of new prefixes over time as shown in Figure 1. As the number of new prefixes decreases over time, it will be increasingly difficult to observe new prefixes. If a FFDN member gets placed on the whitelist the effect is not pathological since it will just be ignored. Further new subnets should drive the calculation of lambda toward the fluxing threshold.

Being on the blacklist marks a prefix as being more likely to be participating in a fluxing A-record. Since the random walk test is driven off the identification of new subnets for each A-record lookup, this will influence the individual lambda calculation farther toward fluxing in a shorter time. Our rationalization for this is due principally to shared infrastructure and common domain members. On the other hand having a CDN member accidentally identified on the black list need not be pathological in that it will only nudge the hypothesis test for a single round of calculation.

Note that it is quite difficult for a FFDN holder to play the system such that they can operate in a protected mode. It is the same predictable behavior that creates whitelist membership, which is antithetical to how FFDN operate. Also if a FFDN operator gets one or more prefix members in the whitelist, standard flux behavior will still come into play and the A-record will be tagged as fluxing. Gaming our implementation of the algorithm will be covered in significant detail in Section 4.4 .

3.5 Traffic Multiplier

As described in Emanuele et. al (2008)] one of the significant issues detecting FFDN in live traffic is the
Proceedings of the Ninth Australasian Information Security Conference (AISC 2011), Perth, Australia

presence of enough hostile traffic for a fluxing host-name to stand out against all the other traffic being observed. The small number of lookups that a network might see over a small time window exacerbates this. In order to address this issue, we created a mechanism which re-issues A-record queries observed over the monitored link. The purpose of this is to amplify the effects of fluxing, which might otherwise fall under the radar. It is important to stress that this is not a mandatory part of the implementation or algorithm. To address the natural concern of an unintentional denial of service attack either on internal or external systems, two features are built in. First, during the lifetime of the multiplier lookup session, the initiator will ignore any additional A-records of the same name. Second, multiple lookups are done over a set of four increasing time windows. These windows are configurable with values of 5, 10, 15, 30 and 60 minutes being typical.

IP addresses observed by the monitor are recorded in a log file. New addresses incidentally observed but not associated with a multiplier lookup (i.e. they are passively observed) are also recorded. As expected, new addresses for a lookup are recorded as well.

4 Algorithm

The heuristic described up till now indicated that some sort of Random Walk test might provide better detection capability than a simple counting mechanism. In this section we put that heuristic into the detection capability than a simple counting mechanism. In this section we put that heuristic into the

denial introduced by dynamic whitelisting, we believe there are problems with the idea of independence for CDN addresses because the addresses returned are influenced by the DNS load-balancing algorithm used. We will discuss non-independent data elements in conditional probability ratio tests in the Future Work section.

4.1 Sequential Hypothesis Testing

From observations outlined in section three, we suggest that it is possible to differentiate between FFDN and CDN traffic by observing the occurrence of new subnets in returned A-record queries. Here we have random variables are determined by the presence or absence of that subnet in the records history. This technique was originally developed by Wald (1947), and further implemented in scan and worm detection in Jung et al. (2004) and Stuart et al. (2004) whose work we draw heavily on to develop this argument.

When a local host requests resolution of an A-record, one or more IP addresses are returned. These addresses are converted into subnet prefixes which are then tested to see if they have been previously associated with this A-record. As in scan detection, we use a random indicator variable $Y_i$ to represent the outcome of the $i$th subnet test:

$$Y_i = \begin{cases} 0 & \text{if } n \text{ is even} \\ 1 & \text{if } n \text{ is odd} \end{cases}$$

For testing we assume two hypotheses. $H_1$ is the hypothesis that the A-record under examination is not fluxing, and $H_0$ is the null hypothesis that it is. As discussed above, it is assumed that conditional on hypothesis $H_j$, the random variables $Y_i | H_j$, $i = 1, 2, 3$ (and hence the returned IP addresses) are independent and identically distributed. The distribution of the random variable $Y_i$ can be described as:

$$P_{Y_i = 0 | H_0} = \theta_0, \quad P_{Y_i = 1 | H_0} = 1 - \theta_0$$

$$P_{Y_i = 0 | H_1} = \theta_1, \quad P_{Y_i = 1 | H_1} = 1 - \theta_1$$

Since the likelihood of a new subnet is greater for a fluxing host than not, we see $\theta_0 < \theta_1$. To choose between two competing hypotheses, we calculate the likelihood that the model would generate the observed sequence of events $Y_n = (Y_1, \ldots, Y_n)$ under each hypothesis. This may be accomplished by maintaining the ratio $\Lambda(Y_n)$, which is defined by:

$$\Lambda(Y_n) = \frac{P_{Y_n | H_1}}{P_{Y_n | H_0}}$$

From the i.i.d. assumptions we can express the ratio in terms of the likelihoods of individual events, simplifying the calculation tremendously.

$$\Lambda(Y_n) = \prod_{i=1}^{n} \frac{P_{Y_i | H_1}}{P_{Y_i | H_0}}$$

The change to $\Lambda(Y_n)$ as a result of the $i^{th}$ observation can be written as $\phi(Y_i)$:

$$\phi(Y_i) = \frac{P_{Y_i | H_1}}{P_{Y_i | H_0}} = \begin{cases} \frac{\theta_1}{\theta_0} & \text{if } Y_i = 0 \text{ (new)} \\ \frac{1 - \theta_1}{1 - \theta_0} & \text{if } Y_i = 1 \text{ (new)} \end{cases}$$

The change to $\Lambda(Y_n)$ as a result of the $i^{th}$ observation can be written as $\phi(Y_i)$:

$$\Lambda(Y_n) = \prod_{i=1}^{n} \phi(Y_i) = \Lambda(Y_{n-1}) \phi(Y_n)$$

starting with $\Lambda(Y_0) = 1$. During this iterative series, the value of $\Lambda(Y_n)$ is compared to an upper threshold $\eta_1$ above which the $H_1$ hypothesis is accepted. If the value falls below a second value $\eta_0$ the $H_0$ hypothesis is accepted. For values between these two thresholds, the results are indeterminate. Our implementation provides the ability to transition between the hypotheses by watching $\Lambda(Y_n)$ after a decision is made to see if it can be re-assessed, it is possible to address the problem of (some) false positives. Upper and lower bounds $\alpha$ and $\beta$ are defined via $P_F$ and $P_\alpha$ as $\alpha \geq P_F$ and $\beta \leq P_\alpha$. $P_F$ is the probability of correct identification and $P_\alpha$ being the probability of
false positive. Typical values for these constants are $\beta = 0.99$ and $\alpha = 0.01$. As shown by Wald (1947) the thresholds used to define success for the different hypothesis tests can be bound in terms of $P_F$ and $P_D$.

$$\eta_1 \leq \frac{P_D}{P_F} \quad \text{and} \quad \frac{1 - P_D}{1 - P_F} \leq \eta_0 \quad (7)$$

By setting the thresholds equal to the bounds in Equation 7, it is possible to express the thresholds exclusively in terms of the user defined parameters $\alpha$ and $\beta$:

$$\eta_1 \leftarrow \frac{\beta}{\alpha} \quad \text{and} \quad \eta_0 \leftarrow \frac{1 - \beta}{1 - \alpha} \quad (8)$$

Since the bounds $\eta_0$ and $\eta_1$ work for arbitrary values of the thresholds, they will also work for this choice:

$$\frac{\beta}{\alpha} \leq \frac{P_D}{P_F} \quad \text{and} \quad \frac{1 - P_D}{1 - P_F} \leq \frac{1 - \beta}{1 - \alpha} \quad (9)$$

Taking the reciprocal of the first inequality in Equation 9 and noting that $P_D$ must be between 0 and 1, $P_F < \frac{P_D}{P_F}$ providing the more convenient expression:

$$P_F < \frac{\alpha}{\beta} \equiv \frac{1}{\eta_1} \quad (10)$$

For the second inequality in Equation 9 the result is:

$$1 - P_D < \frac{1 - \beta}{1 - \alpha} \equiv \eta_0 \quad (11)$$

It is worth noting that $\eta_1$ may result in a false positive rate above the desired bound by a factor of $\frac{1}{\beta}$. Similarly the false negative rate may be affected by as much as $\frac{1}{1 - \alpha}$. This may or may not be significant based on the user selected values of $\alpha$ and $\beta$.

### 4.2 Implementation of Algorithm

For the algorithm implementation we used the Bro intrusion detection system, by Paxson et al. (1998), since it provides both high performance and rapid prototyping. We created a series of implementations differing mostly in how address space is partitioned to measure network locality. For subnet prefix based partitioning we were able to use native bro functionality. For the ASN based partitioning we created a new function call based on the publicly available API libGeoIP developed by Maxmind (2000).

In order to track single and double flux interactions, all A-record and NS-record requests and responses are processed from the network traffic stream. For A-record responses, the record name is associated with the set of addresses in the reply. For responses providing NS record information, the set of A-records within the reply identifying the name servers are recorded as well. The idea here is that by identifying fluxing activity in the field of active A-records, then small fast tables can be used to track NS record changes that can be used to identify Double Flux.

As an example, once an NS record is identified as fluxing for a given domain, all other domains also associated with the set of addresses in the reply will be identified as well. Results from the analyzer look like:

```
FastFluxDomain ns2.kqimitate.com FastFluxA identified with following domains: 
azwhen.cn barelieve.cn bbcaxr.cn nybetter.cn nzthan.cn
```

For the test programs to be able to scale to useful numbers (both in terms of time and traffic volume), it is necessary to do reasonably aggressive state maintenance. By noticing that $>99\%$ of all normal traffic did not exhibit fluxing characteristics at all, we decided that reaping a table entry could take place after a time period equal to the TTL value of the A-record or a minimum value. There will be some issues with churn using this method, but it has not proven to be a problem at this time. We realize that static whitelists might be useful in a production environment, so the functionality is implemented but not used in this version.

### 4.3 Addressing Algorithmic Vulnerabilities in TRW

In Kang et al. (2007) a general attack on the general class of positive reward based methods was proposed. A specific example of this attack named z-scan was demonstrated for TRW scan detection algorithm. For z-Scan the attacker can successfully scan an address space protected by TRW detection with a number of hosts proportional to the log of the address space. This represents a huge improvement over a naive attacker. A natural question then arises - can this style of attack be used against our detection algorithm?

By sharing information between the various systems being used to scanning, an attacker can (1) avoid needless duplication of scanning destination space and (2) to share information about good services. This forces an oscillation in the likelihood state rather than convergence to a decision of scanner identification.

While the collusion problem does not pose a significant issue with flux detection, it should be possible to generate a set of addresses returned from the name server in such a way as to always return pairs of identical addresses. For example, rather than returning the set A1, A2, A3 for an A-record query, you might return A1, A1, A2, A2, A3, A3. This would provide the same effect of eliminating the hypothesis resolution.

By setting a maximum threshold for new prefixes or watching the total count of hypothesis questions before resolution, detecting this attack should be simple enough. Oscillatory behavior is unusual enough that alarming on it should be effective.

### 4.4 Original Holz Implementation and Further Efficiencies

The original algorithm proposed by Holz et al. (2008) provided a simple linear expression for determining the fluxing behavior of an A-record:

$$\text{flux}(n_A, n_{ASN}) = 1.32 \cdot n_A + 18.54 \cdot n_{ASN} \quad (12)$$

Here the constants are derived via a 10-fold cross validation on their corpus of test data, which is used in defining a hyper-plane between a set of addresses defined as fast flux and a set of addresses defined as known good based on the Alexa Top 500 list provided by Alexa (2009) and from the Open Directory Project by Netscape (2009). Variables $n_A$ and $n_{ASN}$ represent the number of associated unique A-records and ASN numbers for the A-record in question. When the return value of this function exceeds a threshold value, the record in question is assumed to be fluxing.

When Equation 12 is implemented against live traffic, issues with false positives become quite apparent, most likely from the lack of CDN membership in the Top 500 list. In order to address this,
a second equation was proposed in Campbell (2008) which notes that the ratio of nA and nASN provides a way for correcting against large numbers of unique IP addresses associated with a small number of ASNs. All other constants and variables retain their original meaning.

\[ f_{\text{flux}}(n_A, n_{\text{ASN}}) = \left( \frac{n_{\text{ASN}}}{n_A} \right) (1.32 - n_A + 18.54 - n_{\text{ASN}}) \]  

(13)

The rationale for this change in addressing problems differentiating CDN and FFDN networks was that CDN infrastructure tends to contain more local redundancy than FFDN which must be somewhat ad-hoc. In the Evaluation section it is Equation 13 that is used for the results rather than Equation 12.

5 Evaluation

In this section the experimental setup, configuration and results are discussed. For developing our methodology and running initial tests, a series of runs were done on unfiltered DNS traffic as seen on the border of a large national laboratory consisting of nearly 6000 users. These data collections were used to observe long-term behavior of the different DNS record type as well as the usefulness of the traffic multiplier described in 3.5.

Data presented in Section 3 is derived from two traffic-gathering sessions. The first, used in Table 1, was gathered over a 48-hour period in December 2008. The data for the remainder of subsections was taken from a 48-hour period in January 2009. The only significant difference is the settings of the traffic multiplier mechanism. For the first sample we re-issued queries at periods of 5 min, 1 hour, 24 hours and 36 hours. For the second sample the lookup times were more aggressive at 5 min, 30 min, 30 min, 30 min.

For the actual evaluation the same configuration was used except that the data was from three 12-hour intervals. The rationale for using three different intervals was not only to stress a shorter time window for the detection scenarios, but also to have different traffic profiles (day vs. night, different days) in order to exercise the different strengths and weaknesses of the various implementations. In order to provide a source of true positives to test against, a list of known FFDN domains was queried during the time that the detectors were running. To ensure this did not bias the results, an equal number of CDN hosts and FFDN were resolved at the same time. The CDN list is chosen from the most aggressive examples extracted from data gathered for Section 3.1.

5.1 Variation of Test Cases and Network Segmentation

To identify the most effective form of address partitioning for locality, we looked at a number of options. From Holz, et al. (2008) we started with a routing abstraction - the Autonomous System Unit (ASN). An ASN is a set of local subnets abstracted together in an efficient data structure to exchange local IP information with other entities on the Internet to help them make routing decisions. Details can be found in RFC1935 by Bates (1996). This seems like a natural unit for analysis in that an ASN follows logical business order rather than subnet boundaries. Multiple subnets can be represented by a single ASN, providing some notion of order not based on strict sub-netting. In addition to segmenting on ASN boundaries, we also explored using a variety of subnet prefixes.

In total we created five different analyzers based on differing segmentation and behaviors using dynamic whitelisting in all cases. These types using the new TRW analyzer are: ASN with and without dynamic blacklisting, and subnet prefixes on /8, /12 and /16 boundaries, additionally we used Equation 13, this is described in detail in Campbell (2008).

5.2 Determination of \( \theta_0 \) and \( \theta_1 \)

For our hypothesis testing, the most significant parameters describe the likelihood that a prefix will be new for a fluxing (FFDN) or not fluxing (CDN) lookup. There is a tension here since if the CDN value is too low, then aggressive CDN behavior will be misclassified as flux like behavior, while if it is too high then actual fluxing behavior will be misidentified.

To estimate the values, we took a set of known FFDN and CDN hosts and ran them through a resolver every 5 minutes for two days. In both cases the list of hosts was generated from known FFDN lists Arbor (2008) and our own records. The initial values of \( \theta_1 \) were calculated from the value of repeat/total prefixes for each of the observed records in the CDN data set, then averaged over all records. For \( \theta_0 \) we did the same using known CDN data.

In terms of repeatability we have concerns about the accuracy of the estimated values. For \( \theta_0 \) there is some leeway for low values, as FFDN tend to be somewhat aggressive in their initial behavior. \( \theta_1 \) needs to strike the balance between a value that is too low (driving up the instance of false positives) and too high (which will decrease the number of identified CDN records).

5.3 Test Results

As previously described, the series of trials took place over three 12 hour intervals on live traffic with all analyzers seeing the same traffic. This section evaluates the performance of the new detection algorithm looking at both false positive and false negative results.

In each case, the number referred to as ‘test cases’ is the number of true positives injected into live network traffic. In several instances the number of identified FFDN hosts was higher than maximum true positive value due to additional fast flux activity on the network. To correct for this, identified FFDN hosts were compared against the test case list and any additional activity is not reported.

For Table 2, each of the larger columns (for example /8 Subnet) contain three smaller columns which are the three different test scenarios.

From these results we think that the most accurate version of the algorithm in this series of tests was the /12 subnet prefix. This was principally due to the low false negative values in conjunction with our thoughts on the false positive numbers discussed below.

There was a number of interesting things we observed which should be pointed out. For previous runs of the ASN and modified Holz (Equation 13) we had seen a somewhat different distribution of true and false positives. In these cases, no additional seeding of representative FFDN or CDN records took place. From this perspective, the results are somewhat artificial but represent the best way to measure false positive and negative results in such a large corpus of data.

Another point of interest was why the /12 networks provided a lower number of false positives than the ASN based analysis. Using ASNs seemed like a natural unit, as boundaries tend to follow business and routing lines, providing a type of metadata. Our
initial mental model was that for the sort of near random distribution expected for the IP addresses of compromised systems, there would be as wide (or wider) a distribution of ASN values than subnet values.

Examining sample false positives is revealing in understanding where the model fails. For the host csp.nyc3.verisign.com we get at a list of returned addresses and the associated ASNs that look like:

<table>
<thead>
<tr>
<th>IP Address</th>
<th>ASN</th>
</tr>
</thead>
<tbody>
<tr>
<td>199.7.48.72</td>
<td>AS36617 VeriSign, Inc</td>
</tr>
<tr>
<td>199.7.50.72</td>
<td>AS36619 VeriSign, Inc</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>199.7.59.72</td>
<td>AS36628 VeriSign, Inc</td>
</tr>
<tr>
<td>199.7.71.72</td>
<td>AS36622 VeriSign, Inc</td>
</tr>
<tr>
<td>216.168.253.58</td>
<td>AS7342 VeriSign, Inc</td>
</tr>
</tbody>
</table>

Here we ended up with a block of address space broken into small chunks, each of which have a large number of ASNs, this was unexpected.

The second interesting thing to look at was the failure of the dynamic black listing. Given the degree of overlap in FFDN networks, we expected better results. Our working hypothesis is that the same effects that are making problems for the ASN based algorithm are poisoning the blacklist with false positives. Since we are interested in using this capacity, it will warrant further investigation.

5.4 False Positive Consistencies

Looking at consistencies within the sets of false positives, we noticed a number of systems not falling into the normal categories defined by the taxonomy we started with. Our initial notion of host types is described in Table 1: all DNS traffic can be split into FFDN, CDN and everything else types with everything else being uninteresting. Looking closer at the everything group, there is a outlier set of hosts which were distributed. Typically the members of OCDNs are composed of volunteer companies, schools or individual users hosting some sort of information service. This distribution of membership creates the same sort of network non-locality seen in FFSN systems.

### Table 2: Comparison of Algorithms Across 3 Runs

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>True Pos</th>
<th>False Pos</th>
<th>False Neg</th>
<th>Test Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASN w/ Blacklist</td>
<td>50</td>
<td>29</td>
<td>5</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>53</td>
<td>30</td>
<td>7</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>28</td>
<td>9</td>
<td>51</td>
</tr>
<tr>
<td>ASN no Blacklist</td>
<td>50</td>
<td>17</td>
<td>1</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>53</td>
<td>22</td>
<td>7</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>22</td>
<td>3</td>
<td>51</td>
</tr>
<tr>
<td>/8</td>
<td>10</td>
<td>5</td>
<td>45</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>1</td>
<td>34</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>2</td>
<td>38</td>
<td>51</td>
</tr>
<tr>
<td>/12</td>
<td>55</td>
<td>16</td>
<td>0</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>52</td>
<td>6</td>
<td>8</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>51</td>
<td>9</td>
<td>0</td>
<td>51</td>
</tr>
<tr>
<td>/16</td>
<td>55</td>
<td>62</td>
<td>0</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>26</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>42</td>
<td>1</td>
<td>51</td>
</tr>
<tr>
<td>Modified Holz</td>
<td>50</td>
<td>5</td>
<td>5</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>52</td>
<td>3</td>
<td>8</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>5</td>
<td>3</td>
<td>51</td>
</tr>
</tbody>
</table>

6 Future Work

There is a great deal more work that needs to be done in this area. One of the most useful directions will be the development of a corpus of tools and algorithms that can be applied to non-independent probability ratios. There are a small number of example problems from other domains, which provide an excellent place to begin. Xinong has a number of articles Xinong (1995), Tan (1995) relating to Clinical Trials that would be a good place to start. A different approach can also be found in Blazek (2001). Additionally using some tools from the botnet community would help is defining signatures as in Giroire (2009).

Resolving problems around implanting reliable dynamic blacklisting should be interesting. A new technique known as Footprint Index in Honeybot (2007) and Emanuele et. al. (2008) has also been shown to have a very low degree of false positives and should be examined as a possible technique to couple with the work here to increase the overall effectiveness of this algorithm.

7 Summary

The literature on identifying Fast Flux Service Networks has generally been derived from examining static lists of known CDN and Fast Flux domains to arrive at an algorithm Honeybot (2007) and Holz, et al. (2008), or extracting a list of suspicious hosts from content related to illicit activities and applying some heuristic for FFSN identification Emanuele et. al. (2008), J. Nazario, T. Holz (2008) and Konte et al. (2000). These approaches gloss over the CDN networks that appear in live DNS traffic, which tend to generate either high false positives or high false negatives in algorithms such as proposed by Holz, et al. (2008). Approaches based on heuristics or manual inspection are useful for forensic analysis, but inappropriate for automated, real time detectors. Newer and more mature methods such as Caglayan (2010) have appeared recently that tackle this problem and complement the work here.

Our approach addresses the gap in the literature, by providing a real time, live traffic Fast Flux detector that successfully handles false positives from large CDNs. The TRW detection algorithm allows us to ask the questions; Is this Fast Flux? Is this not Fast Flux?, after every response to a DNS query. The detector converges on a response based on the relative likelihood of each hypothesis. This work is similar to ongoing work in Kevin (2009) which has a related model.

After a comparison of different methods to quantify network locality for FFSN detection, we find that the first 12 bits of a hosts IP address is more effective than ASNs or other subnet masks. We have also
demonstrated dynamic whitelisting of subnets as a viable means of reducing false positives, due to the high degree of overlap among CDN network prefixes. Once the whitelists have been populated through observing network traffic, the detector is able to accurately identify Fast Flux hosts that have been queried via DNS, typically within seconds or 10s of seconds of the initial query. In addition to providing insight into the DNS patterns of CDN and FFSN hosts, the detector is a useful tool in operational computer security, benefitting both researchers and practitioners in the field of computer security.

References


M. Konte, N. Feamster, J. Jung. Fast Flux Service Networks: Dynamics and Roles in Hosting Online Scams.


MaxMind. IP to ASN mapping Database. http://www.maxmind.com/app/asnum


Development and Evaluation of a Secure, Privacy Preserving Combinatorial Auction

Ben Palmer  Kris Bubendorfer  Ian Welch

School of Engineering and Computer Science
Victoria University Wellington,
PO Box 600, Wellington, New Zealand 6140,
Email: {Ben,Kris,Ian}@ecs.vuw.ac.nz

Abstract

The use of electronic auctions as a means of trading goods has increased year after year. eBay has gone from half a million registered users in 1998 to 88 million today. Businesses have also shown interest in using auctions. However, the traditional single good auction as used by eBay lacks the required ability to express dependencies between goods in complex procurement auctions leading to risky bidding strategies and sub optimal allocations. The use of combinatorial auctions, where bidders can place bids on combinations of goods, allows bidders to take advantage of any dependencies and auctioneers to generate optimal allocations of goods. In this paper we introduce a new algorithm for creating a combinatorial auction circuit that can be used to compute the result of a combinatorial auction by any garbled circuit auction protocol. In an electronic auction bids from competing parties are commercially sensitive information as bidders will not want their competitors finding out the value they place on a given item. Therefore, there has been considerable research into auction protocols that protect knowledge of all bids except the winning bid from everyone, including the auctioneer. The Garbled Circuit (GC) protocol as described by Naor, Pinkas and Sumner is an example of such an auction. However, it has only been used to provide privacy for single good auctions rather than combinatorial auctions and has been considered impractical for realistically sized auctions due to the protocol’s communication overheads. Using our algorithm for creating combinatorial auction circuits, the GC protocol can conduct combinatorial auction while keeping losing bid values secret. We have also conducted performance measurements on both the computation and communication overhead of the GC protocol using our combinatorial auction circuit. These experiments show that the communication overhead is low enough to allow its use for realistically sized auctions (6MB for an auction with 3 goods, a maximum price of 16, and 100 bidders).

Keywords: E-Commerce, Distributed Systems, Security.

1 Introduction

Electronic markets such as eBay and Trade Me have changed the way people buy and sell goods online. eBay and Trade Me support fixed price (the buy now button) and standard auctions using an open outcry English auction protocol. Through the use of such web-based auction services, auctions have become an understood and accepted way for people to trade goods.

A combinatorial auction differs from a normal auction by permitting bidders to express a preference for more than a single good. An arbitrary collection of items defined by the bidder, can have a combined value greater than the sum of the individual items. Bids can be made that are conditional upon obtaining the entire set of desired items. As a simple example, consider a real estate auction, where three adjacent lots (A, B and C) are up for sale. The developer of a retail centre needs a minimum of 2 adjacent lots. If this was treated as 3 separate auctions, the value of lot B (to the developer) would be greater than A or C as winning A or C without B would have no value. The various bidder strategies in this auction are complex, involve risk, and are dependent on the order of the auctions due to the dependencies between the lots. The inability of the single good auction to express such dependencies can lead to sub optimal allocations. A combinatorial auction permits bidders to express these dependencies and thereby enable the auction to result in optimal allocations of goods to bidders.

Garbled circuits are a software technique first presented by Yao (Yao 1982) as a solution to the Millionaire’s Problem, in which two millionaires wish to determine who is richer – without revealing their actual wealth to the other. A garbled circuit involves the creation of a set of Boolean gates in software to compute a function, and then the garbling of the circuit to obfuscate the input and intermediate values, but still allow execution of the function. The principle idea of a garbled circuit is to act as a replacement for the trusted party in transactions between mutually distrustful parties.

Trust is a concept that we humans implicitly understand, but have difficulty in applying this understanding digitally. Trust takes into account the role of the entity, the degree of potential loss and sometimes prior experience or experience of those trusted by you. However, trust can be misplaced, and the degree of risk underestimated. A trusted entity is not necessarily trustworthy. This applies to electronic auctions in particular, as the social mechanisms that enforce trustworthy behaviour in traditional auctions are missing.

Imagine the following scenario from (Bubendorfer et al. 2009). Bob and Jane have surplus resources and wish to sell these resources via Alice, their auctioneer. The auction is a sealed bid reverse auction (or tender), where clients issue requests for resources and resource providers bid (and compete) to supply them. Alice’s auction house is hosted using resources provided by Sam. When a client submits a resource request to Alice, Alice creates an auction and advertises the new auction to Bob and Jane. Bob and Jane
respond by submitting their bids to Alice. At the end of the auction, Alice examines the bids and declares the winner of the auction.

In this scenario bid privacy can be compromised in a number of ways. Alice can freely examine the bids from Bob and Jane. She can then leak this information to others giving them a competitive advantage. Sam could also obtain this information directly from the memory allocated to Alice, or if it were encrypted, extract Alice’s key from memory. If Alice or Sam were also resource providers, then the incentive to cheat is considerable.

One way to solve these problems is to ensure that bids are kept private, that is, hidden from Alice and Sam. At first it seems that this is impossible, as Alice would be unable to compute the winner of the auction. However, we can utilise garbled circuits that enable Alice to compute the outcome of the auction, without revealing anything other than the winner and also resource providers, then the incentive to cheat is considerable.

As for the communication overhead, the tables that code the circuit can be sent from the AI to the auctioneer in advance, before the auction begins, possibly on a CD-ROM or DVD (Naor et al. 1999).

The above quote suggests that it may not be feasible to transmit an auction circuit over a network. However, we can show surprisingly that that it is indeed feasible and this is the case even for multiple good combinatorial auctions. We have implemented the garbled circuit auction protocol by Naor and Pinkas (Naor et al. 1999) and the Verifiable Proxy Oblivious Transfer (VPOT) protocol (Juels & Szydlo 2003) introduced by Juels and Szydlo to fix a problem with the original garbled circuits auction protocol. No performance benchmarks have been published for the garbled circuit auction protocol using VPOT before this work. Finally we compare the performance of this protocol with another well known privacy preserving combinatorial auction protocol based on threshold trust.

2 Related Work

There are two main approaches used to ensure the privacy of bidder valuations; threshold trust (Franklin & Reiter 1995, Yokoo & Suzuki 2002, Suzuki & Yokoo 2002, Harkavy et al. 1998, Peng et al. 2002, Bubendorfer & Thomson 2006) and two party trust (Lipmaa et al. 2002, Naor et al. 1999, Juels & Szydlo 2003, Cachin 1999, Kikuchi 2001). In threshold trust, the co-operation of some quorum of hosts is required to reconstruct a bid. Threshold trust is secure as long as the quorum of honest hosts can be met. To implement threshold trust, different protocols have used different techniques. A threshold El-Gamal homomorphic crypto system has been used to allow computation on encrypted bids while needing a quorum of hosts to decrypt the bids (Yokoo & Suzuki 2002). This homomorphic auction protocol is able to conduct combinatorial auctions. Polynomial secret sharing has also been used (Kikuchi 2001) and extended to conduct combinatorial auctions (Suzuki & Yokoo 2002). Threshold trust has been criticised for requiring a heterogeneous collection of hosts from different organisations willing to commit computing resources to host an auction (Lipmaa et al. 2002). It is easier to find two parties from separate organisations willing to conduct an auction for two party trust than to find a larger group of parties to conduct an auction using threshold trust.

Two party trust relies on a symmetric separation of duty between two parties with the information being kept private. The creation of a novel algorithm to construct combinatorial auction circuits. The resulting circuit is then used to compute the results of a combinatorial auction when given the number of goods, bidders, and the maximum price. Our combinatorial auction circuit can be used with any single good privacy preserving auction protocol, based on garbled circuits (Naor et al. 1999, Jakobsson & Juels 2000, Baudron & Stern 2001), to send it for combinatorial auctions. This is the first example of a combinatorial auction circuit to appear in the literature and we present the circuit and the algorithms used to generate it.

A criticism that is often levelled at garbled circuits is the communication overhead caused by the garbled circuit that is sent from the auction issuer to the auctioneer (Yokoo & Suzuki 2004, 2002, Perrig et al. 2001). Even the creators of the garbled circuit auction protocol state that:

As for the communication overhead, the tables that code the circuit can be sent from the AI to the auctioneer in advance, before the auction begins, possibly on a CD-ROM or DVD (Naor et al. 1999).

Unfortunately this protocol is restricted to five or six bidders in real world situations and a malicious auctioneer could collude with a bidder to break the assumptions of the protocol.

3 A Combinatorial Auction Circuit

A circuit is a network of Boolean gates with a set of inputs, a set of intermediate gates, and a set of outputs gates. Figure 1 shows a simple worked example of an auction circuit. As long as the two parties do not have a quorum of hosts to decrypt the bids (Yokoo & Suzuki 2003). A novel auction protocol has been developed where an auctioneer uses a third party to obliviously compare bid values (Cachin 1999). In this protocol one of the parties learns a partial ordering of the bids, and if the other party colludes with a bidder, then that bidder could see all the comparisons. A similar auction protocol to garbled circuits that does not use an auction issuer but where instead bidders perform the role of the auction issuer has also been developed (Baudron & Stern 2001). A novel auction protocol has been developed where an auctioneer uses a third party to obliviously compare bid values (Cachin 1999). In this protocol one of the parties learns a partial ordering of the bids, and if the other party colludes with a bidder, then that bidder could see all the comparisons. A similar auction protocol to garbled circuits that does not use an auction issuer but where instead bidders perform the role of the auction issuer has also been developed (Baudron & Stern 2001).
3.1 Building Blocks

We make use of the single good 1st price circuit of Kurosawa and Ogata (Kurosawa & Ogata 2002) as a building block for our combinatorial auction circuit. A 1st price auction returns the highest bid as the winner. The circuit is constructed of NOT, AND, OR, XOR, and SELECT gates. A SELECT gate has three inputs, if the first input is true it outputs the second input, and if the first input is false it outputs the third input. The single good auction circuit by Kurosawa and Ogata uses a technique they term bit slicing where the bits of the various bids are compared from most significant to least significant. This is in contrast to the standard first price circuit that computes the millionaires problem comparing each bidder’s bid in turn. We also use a basic add circuit that given two bitwise values as input, outputs the sum of these two values.

Combinatorial auctions can be represented as an auction graph (Figure 2(a)) where nodes represent goods, links between nodes represent a subset of goods, and each complete path through the graph represents an allocation of the goods. The optimal path through an auction graph is the path that returns the highest revenue. The auction graph representation of combinatorial auctions has been used in several previous works (Yokoo & Suzuki 2002, Suzuki & Yokoo 2002).

3.2 The Complete Circuit

The auction graph representation, the 1st price circuit and the add circuit are used to create a circuit to compute the optimal value for a combinatorial auction along with the winning bidders and prices. Figure 2 shows a three good auction graph in 2(a) and 2(b) shows the construction of the resulting combinatorial auction circuit. Every link in the auction graph has a 1st price circuit that outputs the maximum bid for that link. Every node in the auction graph except the last node has an add circuit that adds the maximum bid for the incoming link to the bids on the outgoing link. The last node has a final 1st price circuit that outputs the optimal value for the combinatorial auction.

An auction circuit for combinatorial auctions needs to compute and output not only the optimal value for the auction, but also which bidders won which goods and at what price. Each 1st price circuit outputs the maximum bid for that link and the associated bidder. These values are combined for every link in a path by using a SELECT gate to output the winning bidder only if that link is on the optimal path. Further SELECT gates are used to output the winning price for a bidder on a link only if that link is on the optimal path and if that bidder had the maximum price for that link.

3.3 Circuit Creation Algorithm

We now present our algorithm used in the creation of our combinatorial auction circuit. It takes as input the number of bidders, prices, and goods and outputs a circuit for computing the result of a combinatorial auction. The outputs are a series of bits for each bidder that indicate if that bidder won any of the links on the auction graph and the winning bids they need to pay for each link. In case of tie break the circuit outputs both winning bidders and the auctioneer would need to choose some other way of deciding the winner, such as a coin toss.
Algorithm 1 is the main algorithm used in the creation of the circuit. The algorithm is split into two parts. The first part calculates the optimal path through the auction graph by calculating the maximum bid for each link in the graph, adding together the maximum bids for each path, and then calculating the optimal path based on the maximum bids for each path. The second part of the algorithm calculates the winning bidders and prices using SELECT gates for each bidder and every link and path in the auction graph.

Algorithm 1

Procedure CreateCombinatorialAuctionCircuit

Input: nBidders, nPrices, nGoods

Output: AuctionCircuit AC

1. (* Create the Auction Graph *)
2. AuctionGraph Graph ← CreateAuctionGraph(nGoods)
3. (* Loop Through All Paths *)
4. for Paths i ∈ Graph
5.   (* Loop Through Links on Path *)
6.    for Links j ∈ i
7.      Create 1st Price Circuit with inputs of the bids for link j
8.      (* Get the Max Bid for Path i *)
9.      if Number Links on Path i > 1
10.     for Links j i ∈ i
11.        AddOutputs(j, j+1, AC)
12. (* find the optimal path *)
13. Create 1st Price Circuit with Inputs of the Final Add Circuits For Each Path
14. (* Now find what bidders won for what price *)
15. (* Loop Through All Bidders *)
16. for Bidder i ∈ nBidders
17.   (* Loop Through All Paths *)
18.    for Paths j ∈ Graph
19.      (* Loop Through Links on Path *)
20.     for Links k ∈ j
21.      WinningBiddersPrices (i, j, k, nPrices, AC)
22. return AC

Algorithm 2 is a helper method that is used to add the outputs of the 1st Price Circuit for each link in a path together to get the maximum price for a path in the auction graph. These maximum prices for each path are then compared in the final 1st Price Circuit that outputs the optimal path for an auction.

Algorithm 2

Procedure AddOutputs

Input: Link j, Link j + 1, AuctionCircuit AC

1. if (j is the first link in the path)
2. Create Add Circuit with inputs of the maximum bids for link j and j + 1
3. else
4. Create Add Circuit with inputs of the maximum bids for link j + 1 and the output of the previous Add Circuit

Algorithm 3 is executed for every bidder and every link in every path of the graph. The first SELECT gate outputs the winning bidder of the 1st Price Circuit for this link in the auction graph provided the link is on the optimal path. The second two SELECT gates output the winning price of the 1st Price Circuit for this bidder and link in the auction graph provided the path is on the optimal path and this bidder was the winner of the link.

Algorithm 3

Procedure WinningBiddersPrices

Input: Bidder i, Path j, Link k, nPrices, AuctionCircuit AC

4 Garbled Circuits Auction Protocol

A garbled circuit is a Boolean circuit for computing the result of some function that has been obfuscated by one party to hide the input and intermediate values of the gates of the circuit. When presented with a garbled circuit any party can calculate the result of the function when provided with the garbled input values to the circuit and an output mapping from garbled outputs to the actual output of the original circuit. In the garbled circuit auction protocol, a Boolean circuit is created that outputs the result of the auction (Naor et al. 1999). This circuit is then garbled by a party known as the auction issuer and sent to the auctioneer. Using the garbled circuit created by the auction issuer, the auctioneer is then able to compute the result of the auction after discovering the garbled inputs of the garbled circuit. As long as the auction issuer does not reveal a set of random values it used when garbling the circuit, the input and intermediate values remain hidden from the auctioneer. The verifiable proxy oblivious transfer (VPOT) protocol (Juels & Szydl0 2003) addresses a security flaw in the original garbled circuit auction protocol where the auction issuer could change bids without detection. Figure 3 shows the parties in the garbled circuits auction protocol. The bidders and the client only need to have a connection to the auctioneer, and the auctioneer is the only party that needs a connection to the auction issuer.

The basic steps of a Sealed-Bid auction using the garbled circuit protocol are:

1. (* Creates gate to work out the winning bidders and prices for this link in the graph *)
2. Create a SELECT gate with 3 inputs. The first input is the output of the final 1st Price Circuit for path j, the second input is the output of the 1st Price Circuit for bidder i and link k, and the third input is the output of the final 1st Price Circuit for path j
3. for m < nPrices
4. Create a SELECT gate with 3 inputs. The first input is the output of the final 1st Price Circuit for path j, the second input is the winning price for link k at price m, and the third input is the output of the final 1st Price Circuit for path j
5. Create a SELECT gate with 3 inputs. The first input is the output of the 1st Price Circuit for bidder i and link k, the second input is the previous SELECT node, and the third input is the output of the 1st Price Circuit for bidder i and link k
• The client contacts the auctioneer with details of the auction they wish to run.
• The auctioneer advertises details of the auction including the number of goods, number of prices, and the auction issuer being used.
• The auction issuer constructs a garbled circuit for the auction based on how many bidders, goods, and the number of bits in the price as well as a mapping from garbled outputs to outputs.
• The auction issuer sends the garbled circuit and output mapping to the auctioneer.
• The auction issuer, auctioneer, and bidders use a protocol called verifiable proxy oblivious transfer (VPOT) which results in the auctioneer learning the garbled values of the inputs, and the auction issuer and bidders learning no new information.
• The auctioneer executes the garbled circuit using the garbled input and decodes the output using the output mapping sent by the auction issuer.

More details of the garbled circuit auction protocol, including the algorithms used, can be found in the Appendix.

4.1 Security

The security of the garbled circuit auction protocol comes from the garbling of the circuit that is done by the auction issuer. This garbled circuit is then sent to the auctioneer to execute. As long as the auction issuer does not collude with the auctioneer losing bid values are kept secret. During the garbling of the circuit, each wire connecting the nodes in the circuit is assigned a randomly generated value and a randomly generated permutation of the values of the wire that is used to create the garbled value of the wire. A gate table is then created for each node in the circuit that maps the garbled input of the node to the garbled output. A publicly known random function is used to create the gate table ensuring that knowledge of one combination of the garbled inputs of a node does not reveal the other garbled outputs. The VPOT protocol is then executed by the bidders, auctioneer, and the auction issuer after which the auctioneer learns the garbled inputs of the circuit and can execute the circuit to find the garbled outputs. A mapping is provided by the auction issuer that maps the garbled outputs of the circuit to the actual output. Parties in the garbled circuit auction protocol are assumed to be passive adversaries, although in the original paper verification techniques are discussed which can extend the garbled circuit auction protocol to handle active adversaries. A more detailed security analysis of the garbled circuit auction protocol can be found in the original paper (Naor et al. 1999), and a paper presenting the VPOT protocol contains a detailed analysis of the security of VPOT (Juels & Szydlo 2003).

5 Circuit Size

As stated in the introduction, one of the main criticisms of the garbled circuit auction protocol is the size of the garbled circuit, which is composed of gate tables and an output mapping, that has to be sent from the auction issuer to the auctioneer. Even the creators of the garbled circuit auction protocol suggest sending the gate tables on a CD-ROM or DVD (Naor et al. 1999). In order to investigate these claims, in this section we quantify the size of the gate tables for different combinatorial auctions. We first investigate the complexity of the circuit before providing experimental results on the size of the circuit.

![Figure 4: Circuit Size vs Number of Bidders](image)

5.1 Complexity

Table 1 shows the upper bound on the number of gates used in our combinatorial auction circuit where \( g \) is the number of goods, \( b \) the number of bidders, and \( p \) the bits in the price. There are \( 2^b \) possible unique combinations of goods. There are \( B_g \) possible unique allocations of the \( g \) goods where \( B_g \) is the Bell Number for the number of goods. For every allocation there can be at most \( g \) links in the graph so we assume there are \( g \) links for every allocation.

The largest factor influencing the size of the circuit is the number of goods \( g \). Increasing the number of bidders \( b \) results in a linear growth in the number of nodes in the circuit. Increasing the number of bits in the price \( p \) results in a linear increase in the size of the circuit and an exponential increase in the number of available prices. More available prices mean that bids can be more finely expressed – increasing the bid granularity of the auction protocol. For example, with \( p = 4 \) there are 16 = \( 2^4 \) available prices but with \( p = 5 \) there are 32 = \( 2^5 \) possible prices. When the number of goods \( g \) is increased, the total number of nodes in the auction circuit increases exponentially. When the number of goods is increased linearly the number of possible combinations of goods increases exponentially as there are \( 2^g \) possible combinations of goods.

5.2 Experimental Results

To calculate the size of the garbled circuit, we have taken the number of two input gates in the combinatorial auction circuit and multiplied them by 4 and then by 128. This is because for every two input gate there are 4 entries in the gate table and every entry is 128 bits. The size of the output mapping is not included in this calculation, but is significantly smaller than the size of the gate tables. We have quantified the size of the garbled circuit in respect to the number of bidders, number of goods, and the number of bits in the price. Other than the variable under test, the default parameters selected were ten bidders, three goods, and four bits in the price (for a maximum bid of sixteen).

Figure 4 shows the size of the garbled circuit increasing linearly as the number of bidders increases. The size of the garbled circuit is proportional to \( \ln \) (maximum bid) as shown in Figure 5.

Figure 6 shows the size of the garbled circuit (shown on a logarithmic scale) increasing exponentially as the number of goods increases.

The size of the garbled circuits in these tests would not require a CD or DVD to be sent from the auction issuer to the auctioneer and could be sent over the network. For example, an auction with 3 goods, a maximum bid of 16, and 100 bidders has a garbled
Table 1: Number of Nodes in the Auction Circuit

<table>
<thead>
<tr>
<th>No. of Input Nodes</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Output Nodes</td>
<td>( b(p + (2^g) + p) )</td>
</tr>
<tr>
<td>No. of AND Nodes</td>
<td>( 2B_{bg}(b + 1) )</td>
</tr>
<tr>
<td>No. of OR Nodes</td>
<td>( B_{bg}(1 + p) + B_{bg}(b + 1) )</td>
</tr>
<tr>
<td>No. of SELECT Nodes</td>
<td>( B_{bg}(2p + 1) + B_{bg} )</td>
</tr>
<tr>
<td>No. of XOR Nodes</td>
<td>( 2B_{bg} )</td>
</tr>
<tr>
<td>Total No. of Nodes</td>
<td>( 2^{g+1}bp + B_{bg}(4b + 5) + B_{bg}(3p + 2) + bp )</td>
</tr>
</tbody>
</table>

circuit of approximately 5MB. A larger auction with 5 goods, a maximum bid of 200, and 50 bidders has a garbled circuit size of approximately 85MB. The size of the garbled circuits gets very large for large number of goods but, if some combinations of goods can be removed as invalid, the garbled circuit size would drop. It is also worth noting that construction of a more compact combinatorial auction circuit with less gates would decrease the size of the garbled circuit.

6 Performance Results

We have tested the performance of the garbled circuits protocol in respect to the number of bidders, number of goods, and the number of bits in the price. Other than the variable under test, the default parameters selected for performance measurements were ten bidders, three goods, and four bits in the price. The test machines were a group of four Dell Optiplex GX755s each with an Intel Core 2 Duo processor and 2048MB DDR SDRAM. The auction time recorded is the total time to compute the auction, this includes the creation of the circuit, the garbling of the circuit, the VPOT protocol to learn the garbled inputs, and the garbled circuit execution time.

The time taken to complete the auction increases linearly as the number of bidders increases. Figure 7 shows the time taken to complete the auction when the number of bids in the price is increased. The relationship between the number of bids in the price and the time taken appears to be linear. Increasing the number of bids in the price by 1 bit increases the maximum bid by a power of 2. Figure 9 illustrates the relationship between the time taken to compute the auction and the maximum bid.

Figure 10 compares the performance of garbled circuits with the performance (Bubendorfer & Thom-
Auction Time vs Number of Goods

![Auction Time vs Number of Goods](image)

Figure 10: Auction Time vs No of Goods

Auction Time vs Maximum Bid

![Auction Time vs Maximum Bid](image)

Figure 11: Auction Time vs Maximum Bid

the time taken to compute the auction and the maximum bid linearly. When using the garbled circuit auction protocol, increasing the number of bits in the price increases the time taken to compute the auction linearly but increases the maximum bid exponentially.

7 Conclusions

This paper has shown the development of an algorithm to construct a circuit composed of Boolean gates that can compute the result of a combinatorial auction. When combined with a privacy preserving auction protocol, based on general circuit evaluation, the algorithm can be used to conduct combinatorial auctions where only winning bids are made public. This is the first example of a combinatorial auction circuit to appear in the literature.

We have presented the concept of an auction circuit and described some of the building blocks we have used to create our algorithm. The algorithm to construct a combinatorial auction circuit is presented in detail. The size of the circuit created by our algorithm is presented. The size of the circuit grows linearly with the number of bidders. The size of the circuit increases exponentially with the number of goods as the number of possible combinations of goods in the auction also increases exponentially. The circuit size increases linearly as the number of available prices increases exponentially which provides an advantage for auctions where a large range of bids is required. We have shown that the communication overhead is feasible (6MB for an auction with 3 goods, a maximum price of 16, and 100 bidders). The garbled circuit auction protocol has also been shown to give comparable performance results to the homomorphic combinatorial auction protocol by Yokoo and Suzuki (Yokoo & Suzuki 2002). The garbled circuit auction protocol has also been shown to give comparable performance results to the homomorphic combinatorial auction protocol by Yokoo and Suzuki (Yokoo & Suzuki 2002).

A Garbled Circuit Algorithms

This appendix describes our interpretation of and algorithms for the original single good garbled circuit auction protocol, and a simple worked example of a garbled circuit. This appendix explains ideas first presented in the original paper on the garbled circuit auction protocol (Naor et al. 1999), more details can also be found in the paper on the VPOT protocol (Juels & Szydlo 2003).

A.1 Table of Definitions

The following terms are used in the description of garbled circuits:

- Client: The entity that requests the auctioneer to conduct an auction.
- Auctioneer: Takes the details from the client and runs the auction. Communicates with the auction issuer to get the garbled circuit and garbled input values.
- Auction Issuer: Assists in running the auction. Should be from a separate organisation than the auctioneer. Garbles circuits and then assists the auctioneer in learning the garbled inputs.
- Bidder: Bids on items in the auctions.
- Auction Circuit: Circuit composed of Boolean gates that can be used to compute the result of an auction.
A.2 Garbled Circuit Generation

To garble a circuit, the auction issuer executes the following algorithm on the nodes and wires of the auction circuit.

Algorithm 4
Procedure GarbleCircuit
Input: AuctionCircuit AC, RandomFunction F
Output: GateTable GT, OutputMapping OM
1. (* Assign random values to the wires *)
2. for wire \( i \in AC \)
3. \( \) Randomly generate \( W_0^i \) and \( W_1^i \) corresponding to 0 and 1.
4. Choose a random permutation over \( \{0,1\} \), \( \pi_i : b_i \mapsto c_i \).
5. (* Construct function tables for every node *)
6. for node \( k \in AC \) with input nodes \( i,j \)
7. \( c_i \leftarrow 0 \) to 1
8. \( \) for \( c_j \leftarrow 0 \) to 1
9. \( GT(k)(c_i, c_j) \leftarrow \)
10. GetGTVvalue\((i, j, k)\)
11. (* Construct output mapping *)
12. for output wire \( k \in AC \)
13. \( OM(k, 0) \leftarrow \{ W_0^k, \pi_k(0) \} \)
14. \( OM(k, 1) \leftarrow \{ W_1^k, \pi_k(1) \} \)

Algorithm 4 garbles an auction circuit. The first step is to assign random values to every wire of the auction circuit. Every wire has a value corresponding to 0 and 1 \((W_0^i, W_1^i)\) assigned to it as well as a random mapping of its output \( \pi \) that maps the wires value \( b \) to \( c \).

For every node in the auction circuit a table is constructed that, when the garbled input of the node, outputs the garbled output. If the node is an output node, an output mapping is also produced mapping the garbled output of the node to the actual output. These steps can only be performed with the knowledge of the random values assigned to all the wires. Algorithm 5 details the calculation done for an entry in the gate table. The tables for each node and the output mappings are then sent to the auctioneer to execute the circuit.

A.3 Executing a Circuit

The following algorithm is executed by the auctioneer after it has received the GateTable and OutputMapping arrays from the auction issuer. The auctioneer will also have received the garbled inputs after completing the VPOT protocol with the bidders and auction issuer.

Algorithm 6
Procedure ExecuteCircuit
Input: AuctionCircuit AC, GateTables GT, OutputMapping OM, GarbledInputs GI, RandomFunction F
Output: ActualValues AV
1. (* Reset All Nodes *)
2. for Nodes \( k \in AC \)
3. \( \) Computed\((k) \leftarrow \)false
4. (* Compute All Nodes *)
5. repeat
6. for Node \( k \) with input nodes \( i,j \)
7. \( \) if \((\)Computed\((i) \cap \)Computed\((j)\)) \( \cup \)
8. \( \) (in \( GI \) \( \cap \) (in \( GI \))
9. \( \) GarbledOutput\(_k \leftarrow \)
10. GetGO\((i,j,k,GT)\)
11. (* Convert Garbled Output to Actual Output *)
13. if \((\)GarbledOutput\(_k \) = OM\((o, 1)\))
14. \( \) then AV\((o) \leftarrow 1 \)
15. else AV\((o) \leftarrow 0 \)

Algorithm 6 executes a garbled circuit given the auction circuit, gate tables, output mapping, garbled inputs, and random function. It loops through all the nodes in the auction circuit until they have all been computed. The gate tables are used to compute the garbled output of a node \( k \) with input wires \( i \) and \( j \). Inputs \( i \) and \( j \) will have garbled input values of \( W_i^{b_i}, c_i \) and \( W_j^{b_j}, c_j \). From the garbled inputs the values \( c_i, c_j, W_i^{b_i}, W_j^{b_j} \) and \( b_i, b_j \) can be extracted from the concatenated garbled inputs. Then the garbled output can be computed using algorithm 7. Algorithm 7 uses the entry in the gate table for \( c_i \) and \( c_j \) as well as the output of the random function with seed \( W_i^{b_i} \) and input \( c_j \) and with seed \( W_j^{b_j} \) and input \( c_i \). The output mapping is used to convert the garbled output to the actual output for an output node.

Algorithm 7
Procedure GetGO
Input: InputNode i, InputNode j, Node k, GateTables GT
Output: bit \[ \] GarbledOutput
1. GarbledOutput \( \leftarrow \)
2. \( F(W_i^{b_i}, c_j) \oplus F(W_j^{b_j}, c_i) \oplus GT[k](c_i, c_j) \)
3. return GarbledOutput

A.4 A Simple Garbled Circuit

Figure 12 illustrates a small garbled circuit with an AND and an OR gate as well as the 'Random Values Assigned to Wires' which are the random values and
permutation computed by the auction issuer and kept secret from any of the other parties taking part in the protocol. The auction issuer would have executed algorithm 4 to produce the random values assigned to wires, the gate tables and the garbled output to output mapping. The ‘Random Function F’ is available to any party in the protocol. The garbled value of a wire is set to \(W^0, c\) so for wire \(Z\) the garbled value of 0 is \(01, 0\).

To execute the circuit in Figure 12 the auctioneer would take the following steps:

- Find out the garbled input values. For say \(V = 1\), \(W = 1\), and \(Y = 0\) the output should be 1. The garbled input value for \(V\) is 001, for \(W\) is 010, and for \(Y\) is 010. The garbled input value is the garbled value of the wire for the input value.

- Now we need to execute the gates. To execute the AND gate we use our garbled inputs and the gate table. The output is 001 ⊕ 111 ⊕ 100 = 010. The garbled output to output mapping shows we can see the output of the garbled circuit is 1.

This is a small example that shows how a garbled circuit works. A circuit that executes an auction has thousands of gates depending on the parameters of the circuit.

References


Comparison of Low-Latency Anonymous Communication Systems - Practical Usage and Performance

Thorsten Ries, Andriy Panchenko, Radu State and Thomas Engel

Interdisciplinary Centre for Security Reliability and Trust
University of Luxembourg
Email: {thorsten.ries, andriy.panchenko, radu.state, thomas.engel}@uni.lu

Abstract

The most popular system for providing practical low-latency anonymity on the Internet is Tor. However, many other tools besides Tor exist as both free and commercial solutions. In this paper, we consider five most popular low-latency anonymisation services that represent the current state of the art: single-hop proxies (Perfect Privacy and free proxies) and Onion Routing based solutions (Tor, I2P, and Jon-Donym). We assess their usability and rank them in regard to their anonymity. We also assess their efficiency and reliability. To this end, we define a set of metrics and present extensive measurements based on round-trip time, inter-packet delay variation and throughput. Apart from the technical realization, economic aspects are also crucial for anonymous communication systems. In order to attract more users, which is mandatory in order to improve anonymity per se, systems need to exhibit a certain pay-off. We therefore define an economic model that takes all relevant aspects into consideration. In this paper, we describe the results obtained, lessons learned, and provide guidance for selecting the most appropriate system with respect to a set of requirements.

1 Introduction

For various reasons, people want to protect their identity when communicating over the Internet. Doing so, they protect their privacy. Freedom of expression may be one motivation, while another reason may be a company or customer with the need to stay anonymous1 for certain business transactions.

Based on this need, the aim of this paper is to compare existing implementations of anonymising systems with respect to users' requirements such as performance and usability, also taking into account aspects of anonymity and security as well as the real costs, i.e., monetary costs the user faces. To this end, we assessed five tools that represent the different approaches and the current state-of-the-art in practical anonymisation: free proxies, Perfect Privacy 2, Jon-Donym 3, Tor 4, and I2P 5.

1The term anonymity derives from the Greek word ἀνώνυµια and means "without a name" or "namelessness".

2http://www.perfect-privacy.com

3https://anonymous-proxy-servers.net/en/index.html

4http://torproject.org

5http://www.i2p2.de/

Copyright ©2011, Australian Computer Society, Inc. This paper appeared at the 9th Australasian Information Security Conference (AISC 2011), Perth, Australia, January 2011. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 116, Colin Boyd and Josef Pilenský, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.

In recent years, research in anonymity has been very active, with many approaches developed. However, only a very few of these reached wide-scale deployment and are used in practice. The predominant system in use today is Tor, developed by Dingledeii et al. (2004). Tor is considered to be a low-latency anonymisation tool, which means that data is supposed to be delivered within a reasonable time, allowing the usage of interactive applications such as web browsing.

In contrast, high-latency systems such as Mixmaster and Mixminion, developed by Moeller et al. (2003) and Danezis et al. (2003) respectively, provide a high degree of anonymity and should be considered for exchange of "more sensitive" information. As a drawback, communications like anonymous web browsing would not practically possible because of the long delays. Beside these, several other anonymisation tools exist as both free or commercial solutions following different design approaches: current low-latency approaches can basically be divided into single-hop proxies and Onion Routing approaches, initially invented by Reed et al. (1998).

The easiest solution to hiding the identity of a user is the use of a single proxy server. Traffic is routed through a node that strips off the origin IP address and presents its own instead. The main problem of single-host proxies is that they are a single point of failure in regard to availability and trust.

The current step in the evolution of anonymous networks is Onion Routing, where messages are encrypted in layers and sent from one node to the next. At each hop one layer of encryption is removed (or added, depending on the direction) and the result further forwarded.

Further, users need to distinguish between services where one entity operates both the anonymisation nodes, and the information service (e.g., Perfect Privacy) and services where nodes can be operated by independent third parties (e.g., Tor, I2P).

However, independent of the used anonymisation technique, users' identities may still be discovered using other techniques such as information leakage at the application layer. This can be accomplished through analysis of the HTTP headers or by intersection attacks, using language or font presets for instance as proved by Raymond (2000) and Wright et al. (2003). Therefore, either a service to alter HTTP header information should be provided by proxy service operators, or it is recommended to use filtering proxy on the user side before sending the data to the anonymisation network.

This superficial classification of anonymisation systems already shows the complexity a user faces deciding upon an appropriate solution. During this selection process, several aspects are usually considered. In addition to the most important aspect, the degree of anonymity and performance plays a large...
Figure 1: Aspects of systems selection

In this paper, we examine all relevant incentives and combine them to provide usage guidance on already-deployed anonymity networks by classifying them and showing their strengths and weaknesses. Applying this guidance, users can select the anonymity service that best suits their needs in a concrete situation.

The remainder of this paper is organized as follows: firstly, we give an overview of the tools we compare. Section 2 describes the anonymity systems we used for comparison, and is followed by an overview of related work (Section 3). In Section 4, we briefly examine usability in regard to its impact on acceptance of the tools. Further, we measure and evaluate the performance in terms of round trip time (RTT), Inter-Packet Delay Variation (IPDV), and throughput. We classify anonymisers in regard to their efficiency in Section 5 before addressing practical issues of anonymity and security in Section 6. In Section 7, we calculate and discuss the aspect of reliability, before all relevant aspects are combined to describe the economic impact on a user’s decision process in Section 8. Finally, Section 9 concludes with the lessons learned and future work.

2 Anonymisation Systems

The simplest way of hiding someone’s identity is to use of a proxy server. The receiver of the message only gets the IP address of an intermediate server, not of the sender. The main drawback is that adversaries can easily de-anonymise users by compromising a server or simply providing one. However, this service may still assure a basic level of anonymity. Due to the simple setup, proxy servers are very common, either as free or commercial solutions and can be easily found in the Internet. The providers of these proxy servers are mostly unknown, so one does not know how trustworthy they actually are. Commercial services exist too, such as Perfect Privacy, which currently provides 48 servers in 23 countries worldwide, allowing users to choose either their preferred proxy or a self-defined cascade of proxies, meaning that several proxy servers are combined into a chain. This may increase anonymity and security against an external adversary, but still has the drawback that the service as a whole is operated by a single entity. Perfect Privacy offers a variety of ways of connecting users can simply use the servers as an HTTP- SOCKS proxy. In addition, users can connect to the proxy server via OpenVPN, PPTP VPN, or SSH. As long as the user does not use their own layer of encryption, the traffic from the proxy server to the destination is not encrypted and consequently completely visible to the server provider. This is true for all anonymisation tools presented here.

Another low-latency anonymisation approach provides the possibility of active mixing of the traffic together with Onion Routing. A popular example of using this approach is JonDonym. Started as an open source project at the TU Dresden, JonDonym (formerly known as JAP) became a popular tool to gain anonymity in the Internet. Users can choose between several fixed paths, known as cascades, with nodes provided by JonDonym operators and nodes operated by other organizations or individuals. Currently, there are 34 nodes in the network forming 16 cascades. The operators of JonDonym provide two kinds of service: a free service, usually having two nodes in a cascade with several hundreds users and a commercial service with usually three nodes in a cascade. Compared to the free service, the number of concurrent participants is relatively low (less than 100). Even though traffic mixing is supported in this approach, to the best of our knowledge, it is not activated because of performance issues.

Today’s most widely used anonymisation system is Tor. Also based on onion routing, Tor tries to provide an acceptable degree of anonymity, while allowing the use of interactive web applications. Recently, Dingle-dine (2009) showed that Tor has about 300,000 users daily and about 2,000 relaying nodes. The main difference from JonDonym is its volunteer-based node operation. In order to achieve optimal system performance, Tor currently relies on directory servers, which gather all relevant information about the network and provide information about the performance of nodes to the clients.

I2P is a system similar to Tor and JonDonym. In contrast to JonDonym and Tor, the main objective of I2P is communication within its own network and not with external services. As a consequence, there is currently only one outbound HTTP gateway responsible for all outgoing web traffic. Another difference from Tor and JonDonym is its fully-distributed network, which has no centralised server for coordination and organisation. Hence, the network consists of a set of nodes that communicate with each other in order to achieve anonymity. All traffic is encrypted using garble encryption, which combines multiple messages into one single message to make traffic analysis more difficult.

3 Related Work

In the many years since the establishment of the Internet, network performance has been an extensive field of research, showing different issues and optimizations in a large number of publications, e.g., by Keshav (1999). In recent years, physical networks, also known as overlay networks, were introduced to allow the easy creation of additional network services without modification of the underlying network. These have become a popular topic of interest in network research and shifted several network paradigms to the application layer. Peer-to-peer networks and other overlay network topologies were introduced to improve data exchange or to add additional functionality. Among these is anonymity, which elicited so much interest, that a special field of research, anonymous communication, was established.

Several surveys on anonymous communication systems exist, e.g., conducted by Kelly (2009) or Ren
et al. (2009). In the work of Pries et al. (2008), in which the authors describe the concepts of basic anonymous communication, as well as implemented systems, the need of low-latency anonymous communication systems is highlighted. However, most surveys focus on MixNet based schemes based on the approach of Chaum (1981) for anonymous remailers and Onion Routing (particularly on Tor); minor work has been conducted on other network routing-based techniques like Crowds and P2P networks such as Tarzan, which was developed by Friedman et al. (2002) and MorphMix, an approach by Rennhard et al. (2002). Due to its widespread usage with about 300,000 users daily, existing performance measurements in anonymous communication mainly concentrate on Tor. The main objective is the improvement of performance, for instance using alternative methods of path selection. Very often, authors of related publications concentrate on throughput improvements in Tor and either propose algorithms to achieve higher performance or higher anonymity as shown by Snader et al. (2008). In contrast, the importance of latency in anonymisation networks as performance metric is highlighted by Murdock et al. (2008). However, both publications consider only a single property, while our study combines these with the variance to determine the overall performance.

Other relevant matters in choosing the appropriate anonymisation system are rarely considered. The optimal system needs to be reliable, and also has to be usable and cheap. Economic aspects are covered by Acquisti et al. (2003) to build a general model in order to describe the incentives for participation in anonymous networks. This approach was elaborated by Ngan et al. (2010), going one step further and describing incentives for relaying traffic within Tor with the aim of an overall performance improvement.

The work of Dingledine et al. (2006) emphasizes the usability and the network effect in anonymisation networks. The authors argue the importance of usability to increase the user base and, consequently, on the achievable anonymity. Related to both usability and performance is the time needed for sending and receiving messages. Even when just surfing in the Internet, users expect an appropriate performance. If these expectations are not met, users will most likely not use the service. Various studies have attempted to find out the maximum tolerable time for loading a website. Different numbers can be found in literature, depending on the culture, etc., but recent studies, e.g., by Kopfesel (2006), and Wendelsky et al. (2007) conclude that about four seconds is a maximum tolerable delay for most users.

To the best of our knowledge, to date there has been no practical comparison of all relevant aspects (degree of anonymity, performance, usability, reliability, and cost) of already deployed low-latency anonymisation tools. This paper aims to close this gap.

4 Usability

As already mentioned, usability is a crucial aspect since it is essential to attract more users, which is a prerequisite for improving anonymity. The higher the number of participants, the better the theoretical anonymity due to the increased size of the anonymity set (as in the work of Pfitzmann et al. (2009) ). Consequently, providers of anonymity services aim to have a high number of users, which, in turn, is the main incentive for a degradation in the system’s performance. However, even before evaluating the systems’ features, the user informally evaluates the usability of the anonymisation system during installation and initial configuration. This is of particular importance, as she may already form a negative opinion of the system and may reject its further use.

To evaluate usability, we use the cognitive walkthrough (CW) method, developed by Wharton et al. (1992). Hereby users try to accomplish tasks with the aim of identifying usability issues. The particular evaluation was divided into three steps:

1. CW1: Installation of the anonymising software.
2. CW2: Configuration of the browser/other software.
3. CW3: Verification of the anonymised connection.

In the following, we describe these steps more in detail.

CW1: Installation of the anonymising software

Although some prerequisite software installations may be challenging to inexperienced users, all systems provide well documented websites to support users during the installation process. Very often, step-by-step instructions are given, which vary from a simple double-click (JonDonym) to some more advanced configuration being necessary (Tor and I2P).

CW2: Configuration of the browser/other software

As already mentioned, we tested both free and commercial systems. The two commercial systems, Perfect Privacy and the premium service of JonDonym, have to be paid for in advance. This can be done by credit card or anonymously by using vouchers (see Section 6). Thus, the process of paying makes some additional effort necessary, but is relatively easy to handle.

The aim of Tor is to protect data transport. For web browsing, there are no specific measures to hide potentially unmasking information as such as browser type, language settings, and so forth, which is sent by default to the web server. Therefore the developers highly recommend the installation of a local proxy server that modifies or deletes this information before sending the data. After the installation of the local proxy server, the final step is the same for all other tested systems: the users have to configure the application (in this case, the browser) in order to use a proxy server. Depending on the browser, the step of proxy configuration may be difficult for a less sophisticated user the first time because of the sometimes not obvious location of these settings. Only Tor simplifies this process by installing an add-on (Torbutton) that allows the proxy settings to be easily switched on and off.

CW3: Verification of the anonymised connection

Once the user has configured the browser or the additional software, she needs to verify whether the anonymisation service is running properly. On dedicated web sites that reveal the IP address of the connecting user, it is easy to check the system’s functionality. Some of these web sites provide additional information about security/anonymity issues, like the connecting IP address, HTTP header information and whether Java/Javascript is turned on in the browser. Except for finding an appropriate website, this step was found to be relatively easy to accomplish.
5 Performance

Probably the most important aspect for users on the Internet, even when acting anonymously, is performance. In particular, Round Trip Time (RTT), Inter-Packet Delay Variance (IPDV) and throughput have a significant influence on the overall performance as perceived by users. Because this has a direct impact on the user’s satisfaction, we examine these parameters in detail and calculate the overall efficiency.

5.1 Testbed environment

All measurements described in this section were performed between a client (running Ubuntu 10.04, Intel Core2 Duo, 3GHz, connected at 100Mbit/s to the campus network and with 300Mbit/s to the outside world) located at the University of Luxembourg and two web servers, one located in Luxembourg and one on St. Vincent Island. The basic measurement setup is depicted in Figure 2. In order to allow the comparison of all tools under the same conditions, we used the HTTP protocol as the least common denominator supported by all tools.

![Fig. 2: Testbed setup with either one proxy or a chain of intermediate nodes](image)

For RTT measurements, we chose the Apache web server benchmarking tool\(^9\). It allows measurement of the time a request needs to get about 200 bytes from an HTTP server. Even though this approach involves a certain overhead, it allows a relative comparison of the systems. In order to consider time-shifts and varying network usage at different times of a day, we repeated the measurements over six days.

Measurements of IPDV were conducted every minute over a period of four days, using a dedicated client-server application. We measured the inter-arrival time between every sequence sent with a one second interval in between. The main motivation of the following measurements is the question of whether it is possible to use applications such as VoIP over the anonymisation systems.

Finally, we measured the throughput for three consecutive days using GNU wget\(^10\). We chose to download files of two sizes (100KB and 1MB) to examine the interaction between the amount of the transferred data and the TCP slow start algorithm. We used these file sizes to identify differences between small and large files based on a recent report that states that the average size of a web site is 320KB (Google (2010))\(^11\). Thus, we cover cases of both smaller and larger files.

All measurements were performed using the already described anonymisation tools, applying the following settings:

- Free proxies (FP) were chosen from a web page listing free proxy servers ranked by their performance\(^12\). As these servers typically have a high fluctuation, we had to switch between servers during the test, causing significant downtimes (see Section 7).
- Perfect Privacy (PP) currently offers servers at 23 locations worldwide. Some locations provide only a single server, others up to eight for the purpose of load balancing. We used three randomly chosen nodes out of 48, located in Amsterdam, Moscow, and Chicago.
- JonDo, using three different random premium service cascades (out of nine), having three nodes each. Measurements were not performed using free cascades because the user limit is often reached and, consequently, the service continually becomes unavailable.
- Tor with its default configuration, changing circuits at least every 10 minutes.
- I2P, which also changes internal paths every 10 minutes, but uses always the same single outbound server with estimated 1,000 concurrent users\(^13\).

In addition, we performed the same measurements without any anonymisation tool. This information serves as the reference value to calculate the efficiency and performance losses of anonymisation tools.

![Fig. 3: CDF Luxembourg](image)

5.2 Round Trip Time

Network latency and RTT have a fundamental impact on end-to-end performance in computer networks. Voice over IP (VoIP) applications for instance require a RTT of less than 600ms\(^13\) to provide adequate quality.

Evaluating our measurements, significant differences were identified: the commercial approaches, Perfect Privacy and JonDo, show the lowest average RTTs together with the free proxy, while Tor and I2P are significantly slower by a factor of three to four. The Cumulative Distribution Function (CDF) plots (Fig. 3 and 4) show the fraction of measurements of RTT that are below a certain value. Taking Tor and I2P as an example, the tests on the server in Luxembourg show that Tor can achieve lower RTTs, but between 550ms and 1s, I2P

---

\(^9\) [http://httpd.apache.org/docs/2.0/programs/ab.html](http://httpd.apache.org/docs/2.0/programs/ab.html)


\(^12\) [http://stats.i2p/to](http://stats.i2p/to)

\(^13\) [http://www.itu.int/itu-t/aap/sg12aap/history/g.114/g114.html](http://www.itu.int/itu-t/aap/sg12aap/history/g.114/g114.html)
performs better, meaning that, for instance in 60% of the measurements, Tor showed a RTT of about 800ms, while I2P achieved about 720ms. The results show also that VoIP is only possible with Perfect Privacy (Amsterdam), FP, and JonDonym with some restrictions, because their RTTs are less than 400ms for at least 80% of the measurements.

During our measurements, Perfect Privacy shows three distinct levels of RTTs with each level fluctuating in only a narrow band of a few milliseconds (Fig. 5(a)]. Due to the usage of TCP packets, this pattern is most likely created by packet retransmits. The timeout of TCP packets on GNU/Linux is 3000ms and would explain the very constant additional delays. This suggests that there was congestion on the communication line or the proxy server. The same behaviour has been observed on other Perfect Privacy proxy servers as well. Tor instead shows a wider variance of RTT values (see Fig. 5(b)] due to the variety of possible circuits. Possible retransmits are not easily detectable in this plot.

5.3 Inter-Packet Delay Variation

Interactive real-time applications such as VoIP depend heavily on a constant IPDV. While multimedia streaming applications can compensate differing IPDV by the use of buffers, this is not possible in VoIP. In the sense of Quality of Service of VoIP, IPDV should be <100ms to avoid distortion.14

Figure 6(a) shows the IPDV observed at both servers without anonymisation service. The server in St. Vincent has a smoother and wider distribution, probably caused by the longer distance between client and server, compared to the server in Luxembourg. However, Figure 6 suggests that apart from Perfect Privacy and JonDonym, no other anonymisation service would be able to comply with the recommended value of IPDV for VoIP applications. The values for Tor and I2P are far too high for this kind of communication (Fig. 6(c)), most likely due to congestion. Perfect Privacy, the free proxy, and JonDonym in particular provide a IPDV of less than 50ms (cf. Fig. 6(b)], and satisfy the requirements for carrying VoIP traffic.

5.4 Throughput

In order to evaluate application performance within different scenarios, we measured throughput while transferring files with the sizes of 100KB and 1MB. Due to similarities of the results, only the throughput results to St. Vincent server are shown here.

The first CDF graph (Figure 7(b]) shows the throughput of an anonymisation system while trans-
For the throughput, we calculate the efficiency (Eq) for the transfer of 100KB and 1MB data, respectively as ratio of the mean throughput \(T_{\text{mean}}(AS_i)\) of the corresponding anonymisation system \(AS_i\) to the throughput \(T_{\text{no-anon}}\) measured without anonymisation tool (161KB/s and 734KB/s) using eq. 1.

\[
E_T(AS_i) = \frac{T_{\text{mean}}(AS_i)}{T_{\text{no-anon}}} \tag{1}
\]

RTT efficiency \(E_{RTT}\) is calculated similarly to \(E_T\): it is the ratio of the RTT without anonymisation \((RTT_{\text{no-anon}})\) and the mean RTT of \(AS_i\) \((RTT_{\text{mean}}(AS_i))\).

\[
E_{RTT}(AS_i) = \frac{RTT_{\text{no-anon}}}{RTT_{\text{mean}}(AS_i)} \tag{2}
\]

To further guide users’ decisions, Table 1 lists the efficiencies of throughput and RTT. The value of 1 is the reference value, accomplished without anonymisation and the higher the value, the more efficient is the system.

Apart from I2P, all tools show an acceptable efficiency for 100KB files, but the rates decrease tremendously for the large files. Here, JonDonym and Tor show throughput efficiencies of less than 30%, I2P only 5%.

RTT total efficiency values are calculated as the mean of Luxembourg and St. Vincent. Compared to bandwidth efficiencies, these values are even worse. While again the Perfect Privacy server in Amsterdam performed well, all other \(E_{RTT}\) values are below 40%. Altogether, these figures show that current anonymisation systems still suffer from poor performance.

### 5.5 Systems efficiency

Even though low-latency anonymisation systems in general provide the possibility is being used for interactive applications such as surfing on the Internet, the question remains how they compare to the behaviour and performance without any anonymisation. In this section, we calculate the efficiency of RTT and throughput. We did not include IPDV because as long as the required threshold value is not exceeded (e.g. 100ms for VoIP), the actual value is of no particular interest to the average user.

---

Figure 7: CDF Throughput

Ferring 1MB of data, I2P shows generally the lowest throughput, while the maximum throughput was achieved by the Perfect Privacy proxy in Amsterdam (1044KB/s). This value was even higher than the throughput measured without anonymisation (746KB/s). This is an astonishing result, because our tests were conducted utilising the HTTP header option no-cache, so that there should be no caching on the proxy server. The only explanation could be the usage of a more powerful connection via Amsterdam compared to the native connection.

As can be seen in Figure 7(a), the values for transmission of 100KB files were significantly lower, for Perfect Privacy and without anonymisation tool, by a factor of 4 - 4.5. This effect may result from TCP slow start, when the hosts involved try to achieve the highest throughput for this particular connection by adapting TCP window size. In Tor and I2P, the throughput is almost constant for the different file sizes. The reason for this behaviour, again, is presumed to be congestion within the network.

---

### Table 1: Throughput and RTT Efficiency

<table>
<thead>
<tr>
<th>Anonymisation System</th>
<th>(E_T) (100KB)</th>
<th>(E_T) (1MB)</th>
<th>(E_{RTT})</th>
<th>(E_{RTT})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free Proxy</td>
<td>1.10</td>
<td>0.88</td>
<td>0.27</td>
<td>0.37</td>
</tr>
<tr>
<td>PP Amsterdam</td>
<td>1.37</td>
<td>1.17</td>
<td>0.89</td>
<td>0.89</td>
</tr>
<tr>
<td>PP Moscow</td>
<td>0.36</td>
<td>0.24</td>
<td>0.27</td>
<td>0.27</td>
</tr>
<tr>
<td>PP Chicago</td>
<td>0.36</td>
<td>0.24</td>
<td>0.40</td>
<td>0.40</td>
</tr>
<tr>
<td>JonDonym</td>
<td>0.75</td>
<td>0.39</td>
<td>0.39</td>
<td>0.39</td>
</tr>
<tr>
<td>I2P</td>
<td>0.96</td>
<td>0.86</td>
<td>0.39</td>
<td>0.39</td>
</tr>
<tr>
<td>I2P</td>
<td>0.38</td>
<td>0.06</td>
<td>0.39</td>
<td>0.39</td>
</tr>
</tbody>
</table>

### 6 Anonymity and Security

Anonymity may be quantified using different metrics, as a survey of Kelly et al. (2008) shows, but none of which is comprehensive. In this section, we establish a thorough classification of anonymity for all tested anonymisation services. Because a quantitative comparison of all services is difficult up to impossible (as there is no existing metric that would consider all possible attacks) using existing approaches such as entropy, which is described by Diaz et al. (2002), we performed an educated anonymity/security appraisal and ranked attackers in regard to their ability and cost to de-anonymise users. This ranking is based on our subjective assessment and may differ from other classifications.

The idea is simple: we identified the different roles of adversaries against systems for anonymous communication. We then ranked these adversaries with respect to their power. In order to quantify the anonymity, we ranked the power of an adversary on a scale between 0 and 1. The value of 1 means that the adversary can de-anonymise the involved entities with a high probability, whereas the value of 0 means that the adversary is generally harmless with respect
to the considered anonymisation technique. For instance, while a web service provider has limited power to identify a user coming from an anonymisation network, an Internet exchange (IX) and a Government has much greater power and abilities. Figure 8 shows the results of our appraisal.

![Diagram of anonymisation systems]

Figure 8: Classification of anonymisation systems

No single low-latency anonymisation technique can provide an adequate protection against an attacker having a government or anonymisation service operator status. Therefore, users of anonymisation systems are required to trust the service operator. Using for instance Tor, users get good protection against the Web Service Provider, the Local Network Administrator (LNA), as well as the ISP. This is due to the encryption used between the sender and the first Tor node. A node operator and the External Party (EP) have some more power, as they can add as many nodes to the network as they have resources. Here, an External Party is defined as an entity outside the anonymisation system, that is trying to become a part of it. Hence, every other entity we consider in our categorization can be seen as an EP too.

An even more powerful attacker is the Internet Exchange, as it can observe a considerable amount of traffic between the Tor nodes. Recent studies, e.g., by Edman et al. (2009) show that there is a certain risk that provider of large Autonomous Systems (AS) can control a significant number of entry and exit nodes, hence this is also true for the corresponding Internet Exchange. Service Operator and Government or Secret Service are the most powerful players. They may have enough power to bias path selection, analyse all network traffic, break the encryption, or even apply non-technical means to achieve their goal (e.g., rubber-hose cryptanalysis). We also differentiate between the two available versions of JonDonym. While the free version provides a path-length of two nodes, the premium services always use three nodes. However, we rank the LNA higher for the premium service due to simplified fingerprinting, as proved by Panchenko et al. (2009) because of a smaller number of users.

Calculating the degree of anonymity using this classification, single values are weighted, summed and normalized:

\[
A = 1 - \frac{\sum_{i=1}^{n} (w_i \times a_i)}{\sum_{i=1}^{n} w_i},
\]

where \(a_i\) is the power of attacker \(i\), \(w_i\) the weight an user puts on the attacker \(a_i\) and \(n\) is the number of attackers considered (here: eight). Table 2 shows the degree of anonymity of the tested anonymisation services for two particular cases. Case 1 (C1) shows the values without consideration of any user-based weighting. C2 could be an employee using services like eBay at the workplace. Here, we mainly consider the LNA and the WSP as critical, i.e., giving them a higher weight (LNA: 10, WSP: 3). The other identities are weighted 1. Overall, Tor achieves the highest degree of anonymity and the free proxy the worst. The degree of anonymity of Tor is even higher than in I2P, mainly given because of the single outbound node I2P provides. Due to the higher ranking of a LNA, the free version of JonDonym may be more appropriate in C2.

<table>
<thead>
<tr>
<th>Anonymisation System</th>
<th>A (C1)</th>
<th>A (C2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free Proxy</td>
<td>0.18</td>
<td>0.32</td>
</tr>
<tr>
<td>Perfect Privacy</td>
<td>0.26</td>
<td>0.42</td>
</tr>
<tr>
<td>JonDonym</td>
<td>0.33</td>
<td>0.34</td>
</tr>
<tr>
<td>JonDonym (C2)</td>
<td>0.33</td>
<td>0.34</td>
</tr>
<tr>
<td>I2P</td>
<td>0.44</td>
<td>0.62</td>
</tr>
<tr>
<td>Tor</td>
<td>0.41</td>
<td>0.66</td>
</tr>
</tbody>
</table>

In addition to the degree of anonymity, other aspects, such as anonymous payment for the use of commercial anonymisation tools, are of relevance too, as they may directly influence the anonymity. For example, providing the real name and/or the bank account number would reveal the identity of the user to the company offering an anonymisation service.

The commercial service providers in this comparison, Perfect Privacy and JonDonym, offer an alternative by also accepting payments by anonymous payment schemes such as PaySafeCard or UKash. Users can anonymously buy a code in an ordinary shop and to pay for the anonymisation service with this code as with pre-paid telephone cards, but without any personal registration being required. Another possibility for ensuring anonymity during the payment process is the usage of anonymous credit cards, which work either like pre-paid cards or like gift cards. Pre-paid cards need to be charged before usage, gift cards can be bought already containing a certain balance.

Considering the difference between free and commercial service operators, we cannot preempt the user’s decision to which service is more trustworthy; users have to trust the operator in both cases. Only the operator’s intention may vary, and range from commercially-driven to belief in expression of freedom or the hope of creating a trap to harvest sensitive information.

7 Network Reliability

The next essential aspect, which is particularly important for user satisfaction, is the reliability of the network. We assess it in terms of the failure rate.
calculate the failure rate, all unanswered RTT benchmarking requests were counted during the period of experiment execution.

A common parameter to describe the failure rate is MTBF, which expresses the Mean Time Between Failures of a system. In this context, MTBF is calculated as the sum of the uptime periods divided by the number of downtimes:

$$MTBF = \frac{\Sigma(t_{\text{up}} - t_{\text{down}})}{n_{\text{down}}}$$

(4)

where $n_{\text{down}}$ is the number of failures.

We also calculated the Mean Time To Recovery (MTTR), which is computed in the same way as MTBF (Equation 5). In order to evaluate reliability, both factors need to be considered.

$$MTTR = \frac{\Sigma(t_{\text{up}} - t_{\text{down}})}{n_{\text{up}}}$$

(5)

Tables 3 and 4 show that the loss of RTT connections occurred by the free proxy, which is the result of proxy servers going offline periodically from time to time. During our experiment, we twice had to switch to a new proxy server. Another issue is related to the connection to the webserver in St. Vincent, which showed problems for two hours when some of the packets did not get through. The relatively high packet loss of JonDonym was caused by a service interruption of more than two hours. This interruption only affected JonDonym traffic to the server in Luxembourg while all other services were working well, including the connections to St. Vincent. This result suggests that there was a problem of connectivity between the final JonDonym relay and the server. However, the numbers only present a snapshot and may not necessarily reflect the long-term behaviour.

Table 3: Snapshot of MTBF and MTTR - St. Vincent (SV)

<table>
<thead>
<tr>
<th>AS</th>
<th>PL</th>
<th>MTBF (SV)</th>
<th>MTTR (SV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0.29%</td>
<td>0.0195</td>
<td>0.0004</td>
</tr>
<tr>
<td>Free Proxy</td>
<td>2.21%</td>
<td>0.1075</td>
<td>0.0011</td>
</tr>
<tr>
<td>PPA</td>
<td>0.29%</td>
<td>0.0105</td>
<td>0.0014</td>
</tr>
<tr>
<td>PPC</td>
<td>0.21%</td>
<td>0.0098</td>
<td>0.0016</td>
</tr>
<tr>
<td>JonDonym</td>
<td>0.36%</td>
<td>0.0154</td>
<td>0.0019</td>
</tr>
<tr>
<td>Tor</td>
<td>0.36%</td>
<td>0.0093</td>
<td>0.0012</td>
</tr>
<tr>
<td>FP</td>
<td>0.50%</td>
<td>0.0122</td>
<td>0.0015</td>
</tr>
</tbody>
</table>

Table 4: Snapshot of MTBF and MTTR - Luxembourg (Lux)

<table>
<thead>
<tr>
<th>AS</th>
<th>PL</th>
<th>MTBF (Lux)</th>
<th>MTTR (Lux)</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0%</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Free Proxy</td>
<td>2.00%</td>
<td>0.0354</td>
<td>0.0018</td>
</tr>
<tr>
<td>PPA</td>
<td>0.00%</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>PPC</td>
<td>0.02%</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>JonDonym</td>
<td>0.16%</td>
<td>0.0416</td>
<td>0.0010</td>
</tr>
<tr>
<td>Tor</td>
<td>0.00%</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>FP</td>
<td>0.22%</td>
<td>0.0039</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Fig. 9 illustrates the number of unanswered RTT requests during the measuring period. A high number of lost messages without the use of any anonymity system is a sign of a general network problem. However, the figure also shows the influence of path selection on reliability. It is again possible to see a high number of lost packets for FP. This result confirms our observations during previous tests.

Another observation from reliability tests is the influence of the server location. Excluding the free proxy due to its outages, connections to the server in St. Vincent summed up over all services show a higher loss in total (822) compared to Luxembourg (561). The reason for this issue may be a general network problem, not related to any anonymisation service. Normalising these results, the number of unanswered RTTs for all services, except I2P and the free proxy server, is quite low.

8 Economic aspects

Apart from the technical aspects of finding the appropriate anonymisation tool, users have also to decide on the economic value/cost. Some services rely on an active participation where users pay indirectly ($C_1$) by providing, e.g., computational resources. Using I2P for instance requires provision of bandwidth and computational power in order to use the network, while in Tor, users have the choice of either donating resources by acting as a relay node or as a client only. Paying indirect costs may be negligible in most cases, but may, on the other hand, limit the maximum achievable performance. We calculate $C_1$ as the sum of $C_T$, $C_r$ and $C_b$, where $C_r$ is the relaying/routing costs, $C_b$ costs of de- or encryption (computational effort) and $C_b$ the costs of providing additional bandwidth.

$$C_1 = \Sigma(C_T, C_r, C_b)$$

(6)

Using commercial anonymisation systems, users have to pay fees. These direct costs $C_d$ are based on usage time or data volume. For instance the business model of Perfect Privacy is based on a monthly fee, offering a data flat rate. JonDonym instead bills according to the amount of data. Consequently, the overall costs are calculated as a sum of the two costs:

$$C = (C_1 + C_d)$$

(7)

The payoff costs for every tool are then calculated as follows:

$$P_i = \frac{w_A \cdot A + w_E \cdot E + w_U \cdot U}{w_R \cdot R} - w_r \cdot C$$

(8)

where $A$ is the degree of anonymity, $E$ the efficiency, $R$ the reliability and $U$ the usability; $w_{[a,e,u,r,c]}$
are the different weights the user defines depending on her particular needs. Consequently, the calculation strongly depends on user requirements and has to take into account the actual situation.

At first glance, users main goal may be a high degree of anonymity together with a high efficiency and reliability of the system at low cost. This does not mean that users will not pay for such a service; statistics for Perfect Privacy show about 25,000 concurrent connections. Even though this might be not that accurate, it still shows that a fast and reliable service can attract many users willing to pay a certain amount of money. Another example is Tor, with about 300,000 daily users. Tor is also known for its high anonymity and reliability, even at no cost, but with the drawback of poor performance. For the launch of future anonymisation systems, especially for commercial products, the operators need to take all these aspects into consideration.

9 Conclusion and future work

In this paper, we have defined a set of metrics in order to compare and evaluate live already deployed state-of-the-art anonymous communication systems in regard to their anonymity, performance, reliability, usability, and cost.

Besides the installation, which is relatively simple for all systems, the usability of the tools is generally good and should not be a reason for non-use. Usability does not vary not much from the users’ point of view; they always have to configure their application, i.e., web browser, to use a proxy server, a process which is practically the same for all tools.

In order to provide a comprehensive comparison of the anonymisation tools, we ranked them in regard to the power of possible adversaries. Our classification is subjective and may vary from other opinions, but gives users an indication of strength and weaknesses of corresponding anonymisers. A future goal might be to simplify the presentation of the results and present them to users in a more appropriate way. This will be subject of further work. In addition, we measured throughput, RTT, IPDV and failure rate and calculated efficiencies. The results show that the proxy-based anonymisation systems outperform the Onion Routing approaches in throughput and RTT, but provide less anonymity. This trade-off applies to all systems and in the end, the user must decide which system best fits her requirements. However, web browsing is possible using all the tested tools, even though I2P in particular has long response times. Latency-critical applications like VoIP, which rely on highly responsive networks, are only usable to a certain extent with the the systems we examined.

An important finding is the efficiency of the throughput performance of single proxy solutions. They perform as well the native communication, sometimes even better.

We also observed that the selected anonymisation path and the recipient’s location have a strong influence on performance and reliability. The Perfect Privacy proxy in Amsterdam, which outperformed the communication without anonymisation, demonstrates this. In general, all Perfect Privacy proxies we tested, as well as Tor, showed particularly reliability in terms of successful connections. While JonDonym and I2P were slightly less reliable, the most unreliable service was the free proxy service due to a high fluctuation of nodes. This demonstrates the main problem with single-proxy solutions but makes them applicable to high-performance short term downloads.

Economics in anonymity is still an under-investigated field of research, with only a few publications. In this paper, we show that besides real costs in terms of money, all relevant aspects such as performance, anonymity, reliability, usability, and cost, need to be evaluated in order to calculate system’s payoff. However, as already mentioned, this calculation strongly depends on individual users’ preferences.

To summarise our results, we established a comparison table, containing all examined anonymisation systems. We classified the systems in five groups, expressed on a scale of very good (+ +), good (+), average(0), bad(-) and very bad(- -). Table 5 shows this classification.

<table>
<thead>
<tr>
<th>Anonymisation system</th>
<th>V</th>
<th>A</th>
<th>E</th>
<th>R</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free Proxy</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Perfect Privacy</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>JonDonym</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Tor</td>
<td>0</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I2P</td>
<td>0</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

U = Usability, A = Anonymity, E = Efficiency, R = Reliability and C = Cost.

Overall, Tor shows the best results, followed by I2P. They score well in all disciplines except performance, which is their main weak point. Here, single proxy solutions score with the best performance. Unfortunately, their degree of anonymity is poor and additionally reliability leaves much to be desired. JonDonym performs averagely, showing no particular strength or weakness. However, it is very difficult to consider all users’ requirements and it is finally up to them to evaluate the results in order to find the most appropriate solution.

To conclude, future work will be necessary in the following areas:

- Extending the usability evaluations by also involving less sophisticated users,
- Further investigation of the very high throughput via certain anonymisation paths,
- Economic aspects need to be evaluated in more detail, especially in regard to business purposes,
- It may be worthwhile to include social aspects into the proposed payoff function, e.g., particular group behaviour in an anonymous network.

Overall, this comparison shows the need and motivation to spend further effort on the improvement of existing anonymisation services or to work on alternative solutions.

Acknowledgement. This work has been partially supported by EC FP7 EFIPSANS project (INFSO-ICT-215549). Furthermore, we would like to thank Dominic for his extensive proof reading.

References


References

Köppell, S. (2006). Low Latency Anonymity Commu-
Keshaev, S. (1999). On individual and aggregate TCP
Kelly, D. (2009). A taxonomy for and analysis of
Author Index

Agnesse, Andrea, 9
Baba, Kensuke, 3
Bai, Yun, 51
Boyd, Colin, iii
Bubendorfer, Kris, 67
Campbell, Scott, 57
Chan, Steve, 57
Clark, Andrew, 23
Corney, Malcolm, 23
Gao, Xiaoying, 33
Ikeda, Daisuke, 3
Inenaga, Shunsuke, 3
Kaosar, Md. Golam, 15
Khan, Khaled, 51
Komisarczuk, Peter, 33
Lackner, Gntner, 41
Le, Van Lam, 33
Lee, Jason, 57
Mohay, George, 23
Nakamura, Toru, 3
Palmer, Ben, 67
Panchenko, Andriy, 77
Paulet, Russell, 15
Pedicini, Marco, 9
Pieprzyk, Josef, iii
Ries, Thorsten, 77
State, Radu, 77
Teufl, Peter, 41
Welch, Ian, 33, 67
Yasuura, Hiroto, 3
Yi, Xun, 15
Recent Volumes in the CRPIT Series

ISSN 1445-1336

Listed below are some of the latest volumes published in the ACS Series Conferences in Research and Practice in Information Technology. The full text of most papers (in either PDF or Postscript format) is available at the series website http://crpit.com.

Volume 91 - Computer Science 2009

Volume 92 - Database Technologies 2009

Volume 93 - User Interfaces 2009

Volume 94 - Theory of Computing 2009
Edited by Prabhjot Manjot, University of Ballarat and Rod Downey, Victoria University of Wellington. January, 2009. 978-1-920682-75-0.

Volume 95 - Computing Education 2009

Volume 96 - Conceptual Modelling 2009

Volume 97 - Health Data and Knowledge Management 2009

Volume 98 - Information Security 2009

Volume 99 - Grid Computing and e-Research 2009

Volume 100 - Safety Critical Systems and Software 2009

Volume 101 - Data Mining and Analytics 2009

Volume 102 - Computer Science 2010
Edited by Bernard Mans, Macquarie University, Australia and Mark Reynolds, University of Western Australia, Australia. January, 2010. 978-1-920682-83-5.

Volume 103 - Computing Education 2010

Volume 104 - Database Technologies 2010

Volume 105 - Information Security 2010

Volume 106 - User Interfaces 2010

Volume 107 - Parallel and Distributed Computing 2010
Edited by Jianwen Chen, Swinburne University of Technology, Australia, and Hajiv Ganji, University of New South Wales, Australia. January, 2010. 978-1-920682-88-0.

Volume 108 - Health Informatics and Knowledge Management 2010

Contains the proceedings of the Thirty-Second Australasian Computer Science Conference (ACSC2009), Wellington, New Zealand, January 2009.

Contains the proceedings of the Twentieth Australasian Database Conference (ADC2009), Wellington, New Zealand, January 2009.

Contains the proceedings of the Ninth Australasian User Interface Conference (AUIC2009), Wellington, New Zealand, January 2009.

Contains the proceedings of the Fifth Asia-Pacific Conference on Conceptual Modelling (APCCM2008), Wollongong, NSW, Australia, January 2008.

Contains the proceedings of the Australian Workshop on Grid Computing and e-Research (AusGrid 2009), Wellington, New Zealand, January 2009.

Contains the proceedings of the Second Australian Workshop on Health Data and Knowledge Management (HDKM 2009), Wellington, New Zealand, January 2009.

Contains the proceedings of the Eighth Australasian Data Mining Conference (AusDM 2009), Melbourne, Victoria, Australia, November, 2009.

Contains the proceedings of the Thirty-Third Australasian Computer Science Conference (ACSC 2010), Brisbane, Queensland, Australia, January 2010.

Contains the proceedings of the Twelfth Australasian Computing Education Conference (ACE2010), Brisbane, Queensland, Australia, January 2010.

Contains the proceedings of the Twenty-First Australasian Database Conference (ADC2010), Brisbane, Queensland, Australia, January 2010.

Contains the proceedings of the Eight Australasian Information Security Conference (AISC 2010), Brisbane, Queensland, Australia, January 2010.

Contains the proceedings of the Eleventh Australasian User Interface Conference (AUIC2010), Brisbane, Queensland, Australia, January 2010.

Contains the proceedings of the Fourth Australasian Workshop on Health Informatics and Knowledge Management (HIKM 2010), Brisbane, Queensland, Australia, January 2010.
Proceedings of the Ninth Australasian Information Security Conference (AISC 2011), Perth, Australia

Volume 109 - Theory of Computing 2010

Volume 110 - Conceptual Modelling 2010

Volume 112 - Advances in Ontologies 2009