Lingyu Wang
Basit Shafiq (Eds.)

LNCS 7964

Data and Applications Security and Privacy XXVII

27th Annual IFIP WG 11.3 Conference, DBSec 2013
Newark, NJ, USA, July 2013
Proceedings

ifip

Lecture Notes in Computer Science 7964

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

Lingyu Wang Basit Shafiq (Eds.)

Data and Applications Security and Privacy XXVII

27th Annual IFIP WG 11.3 Conference, DBSec 2013
Newark, NJ, USA, July 15-17, 2013
Proceedings

Springer

Volume Editors

Lingyu Wang
Concordia University
Concordia Institute for Information Systems Engineering (CIISE)
1455 de Maisonneuve Blvd. West, Montreal, QC H3G 1M8, Canada
E-mail: wang@ciise.concordia.ca

Basit Shafiq
Lahore University of Management Sciences (LUMS)
Department of Computer Science
DHA, Lahore Cantt., Lahore 54792, Pakistan
E-mail: basit@lums.edu.pk

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-39255-9 e-ISBN 978-3-642-39256-6
DOI 10.1007/978-3-642-39256-6
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013941388

CR Subject Classification (1998): C.2.0, K.6.5, C.2, D.4.6, E.3, H.4, C.3, H.2.7-8, E.1

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume contains the papers presented at the 27th Annual WG 11.3 Conference on Data and Applications Security and Privacy (DBSec 2013). The conference, hosted at Rutgers University, Newark, NJ, USA, during July 15–17, 2013, offered outstanding research contributions to the field of data and applications security and privacy. This year's conference, for the first time at DBSec, presented both the Best Paper Award and the Best Student Paper Award.

In response to the call for papers, 45 papers were submitted to the conference. Each paper was reviewed by at least three, and on average 4.04, members of the Program Committee, on the basis of its significance, novelty and technical quality. The review process was followed by intensive discussions over a period of one week. Of the papers submitted, 16 full papers and six short papers were accepted for presentation at the conference. The conference program also includes two invited talks by Johannes Gehrke (Cornell University) and Vincent Poor (Princeton University). The accepted papers cover diverse research themes, ranging from classic topics, such as access control and privacy, to emerging issues, such as security and privacy in data outsourcing, mobile computing, and cloud computing.

The success of this conference was the result of the efforts of many people. We would especially like to thank Claudio Agostino Ardagna (Publicity Chair), Reza Curtmola and Heechang Shin (Local Arrangements Chairs), and Vijayalakshmi Atluri (IFIP WG 11.3 Chair). We would also thank the Program Committee members and the external reviewers for their hard work in reviewing and discussing the papers. We gratefully acknowledge all authors who submitted papers for enhancing the standards of this conference. Last but not least, thanks to all the attendees. We hope you will enjoy reading the proceedings.

July 2013

Jaideep Vaidya
Soon Ae Chun
Lingyu Wang
Basit Shafiq

Organization

Executive Committee

General Chair

Jaideep Vaidya	Rutgers University, USA

General Co-chair

Soon Ae Chun	Rutgers University, USA

Program Chair

Lingyu Wang	Concordia University, Canada

Program Co-chair

Basit Shafiq	Lahore University of Management Sciences, Pakistan

Publicity Chair

Claudio Agostino Ardagna	University of Milan, Italy

Local Arrangements Chairs

Reza Curtmola	New Jersey Institute of Technology, USA
Heechang Shin	Iona College, USA

IFIP WG 11.3 Chair

Vijayalakshmi Atluri	Rutgers University, USA

Program Committee

Gail-Joon Ahn	Arizona State University, USA
Claudio Agostino Ardagna	Università degli Studi di Milano, Italy
Vijay Atluri	Rutgers University, USA
Joachim Biskup	Technische Universität Dortmund, Germany
Marina Blanton	University of Notre Dame, USA

David Chadwick	University of Kent, UK
Soon Ae Chun	Columbia University and City University of New York, USA
Frédéric Cuppens	TELECOM Bretagne, France
Nora Cuppens-Boulahia	Telecom Bretagne, France
Sabrina De Capitani di Vimercati	Università degli Studi di Milano, Italy
Mourad Debbabi	Concordia University, Canada
Josep Domingo-Ferrer	Rovira i Virgili University, Spain
Eduardo B. Fernandez	Florida Atlantic University, USA
Simone Fischer-Huebner	Karlstad University, Sweden
Sara Foresti	Università degli Studi di Milano, Italy
Ehud Gudes	Ben-Gurion University, Israel
Ragib Hasan	University of Alabama at Birmingham, USA
Sushil Jajodia	George Mason University, USA
Sokratis Katsikas	University of Piraeus, Greece
Adam J. Lee	University of Pittsburgh, USA
Yingjiu Li	Singapore Management University, Singapore
Javier Lopez	University of Malaga, Spain
Emil Lupu	Imperial College London, UK
Martin Olivier	University of Pretoria, South Africa
Stefano Paraboschi	Università di Bergamo, Italy
Wolter Pieters	Delft University of Technology, The Netherlands
Indrajit Ray	Colorado State University, USA
Indrakshi Ray	Colorado State University, USA
Kui Ren	State University of New York at Buffalo, USA
Kouichi Sakurai	Kyushu University, Japan
Pierangela Samarati	Università degli Studi di Milano, Italy
Basit Shafiq	Lahore University of Management Sciences, Pakistan
Heechang Shin	Iona College, USA
Anoop Singhal	National Institute of Standards and Technology, USA
Traian Marius Truta	Northern Kentucky University, USA
Jaideep Vaidya	Rutgers University, USA
Lingyu Wang	Concordia University, Canada
Meng Yu	Virginia Commonwealth University, USA
Xinwen Zhang	Huawei Research Center, USA
Jianying Zhou	Institute for Infocomm Research, Singapore
Zutao Zhu	Google Inc., USA

Additional Reviewers

Wanyu Zang
Wei Huo
Tobias Pulls
Yifei Wang
Wen Ming Liu
Chunhua Su
Jordi Soria-Comas
Sara Hajian
Dima Alhadidi
Mikhail Strizhov
Rose-Mharie Ahlfeldt
Md Munirul Haque
Bagus Santoso
William Garrison
Qingji Zheng
Zhan Qin
Chao Zhang
Yosr Jarraya
Rodrigo Roman
Cornelia Tadros
Meixing Le
Tarik Moataz
Liang Cai
Yihua Zhang
Shams Zawoad
Yoshikazu Hanatani
Daisuke Moriyama
Stere Preda
Junpei Kawamoto
Zhan Wang
Amril Syalim
Alessandra De Benedictis
Nurit Gal-Oz
Michael Emirkanian-Bouchard
Massimiliano Albanese
Tantan Liu
Rolando Trujillo-Rasua
Ruben Rios
Rasib H. Khan
Safaa Hachana
Ramadan Abdunabi
Ruoyu Wu

Table of Contents

Data Outsourcing

Mobile Computing

Short Papers

Extending Loose Associations to Multiple Fragments

Sabrina De Capitani di Vimercati[1], Sara Foresti[1], Sushil Jajodia[2], Giovanni Livraga[1], Stefano Paraboschi[3], and Pierangela Samarati[1]

[1] Università degli Studi di Milano, 26013 Crema, Italy
{firstname.lastname}@unimi.it
[2] George Mason University, Fairfax, VA 22030-4444, USA
jajodia@gmu.edu
[3] Università degli Studi di Bergamo, 24044 Dalmine, Italy
parabosc@unibg.it

Abstract. Data fragmentation has been proposed as a solution for protecting the confidentiality of sensitive associations when publishing data at external servers. To enrich the utility of the published fragments, a recent approach has put forward the idea of complementing them with *loose associations*, a sanitized form of the sensitive associations broken by fragmentation. The original proposal considers fragmentations composed of two fragments only, and supports the definition of a loose association between this pair of fragments. In this paper, we extend loose associations to multiple fragments. We first illustrate how the publication of multiple loose associations between pairs of fragments of a generic fragmentation can potentially expose sensitive associations. We then describe an approach for supporting the more general case of publishing a loose association among an arbitrary set of fragments.

Keywords: Loose associations, fragmentation, confidentiality constraints, privacy, data publishing.

1 Introduction

The strong need for sharing and disseminating information that characterizes our global internetworked society raises a number of privacy concerns (e.g., [8,12]). In fact, the vast amount of data collected and maintained in the digital infrastructure often includes sensitive information that must be adequately protected. There is then a clear trade off between the need of easily accessing, using, and distributing information, and the equally strong need of providing proper protection guarantees to sensitive information. Traditional solutions aimed at protecting data undergoing public or semi-public release are based on k-anonymity [17] and differential privacy [11], which protect respondents' identities and their sensitive information by releasing a sanitized version of the data. These solutions however are not applicable in scenarios characterized by the need of releasing

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 1–16, 2013.

non-modified information. Recent solutions have proposed the use of *fragmentation* for protecting sensitive associations among data [1,4,5]. Intuitively, fragmentation protects sensitive associations among different pieces of data by storing them in different fragments that cannot be joined. On one hand, fragmentation improves data accessibility and query performance since it allows the storage of plaintext values at the server side. On the other hand the utility of the published data may be compromised since fragmentation breaks the associations existing in the original data collection. This problem has been addressed by observing that often it may be sufficient to guarantee that sensitive associations cannot be precisely reconstructed (i.e., they can be reconstructed with a minimum degree k of uncertainty). For instance, if the association between patients' names and the disease they suffer from is sensitive, it would be sufficient to guarantee that a recipient cannot associate each patient with less than k possible diseases. In these scenarios, fragments can be complemented with *loose associations* [9], which permit to partially reconstruct the association between sub-tuples in fragments, while not precisely disclosing the association among attribute values that are considered sensitive. Loose associations partition the tuples in fragments in groups and release the associations between sub-tuples in fragments at the granularity of group (instead of the precise tuple-level association). Loose associations can then be used for evaluating aggregate queries, with limited errors in the result, and for data mining. The existing approach operates under the assumption that a fragmentation includes two fragments only, and produces a single loose association between this pair of fragments. A fragmentation may however include an arbitrary number of fragments, and the definition of a loose association may then consider the presence of multiple fragments. A naive solution would publish multiple loose associations, one for each pair of fragments involved in associations that need to be loosely released. Such an approach unfortunately opens the door to privacy breaches since associations that go beyond the two fragments are not considered, and could then be exposed (i.e., a recipient could be able to reconstruct them).

In this paper, we aim at overcoming such a limitation, proposing a solution for the definition of loose associations among arbitrary sets of fragments. The remainder of the paper is organized as follows. Section 2 introduces the basic concepts. Section 3 illustrates the privacy risks caused by the release of multiple loose associations. Section 4 presents our definition of loose association among an arbitrary set of fragments. Section 5 describes the heterogeneity properties ensuring that a loose association satisfies a given privacy degree. Section 6 discusses the advantages and some interesting properties enjoyed by our novel formulation. Section 7 discusses related work. Finally, Section 8 concludes the paper.

2 Basic Concepts

We consider a scenario where a data owner publishes a set $\mathcal{F} = \{F_1, \ldots, F_n\}$ of fragments of a private relation S. Sensitive associations among attributes in S

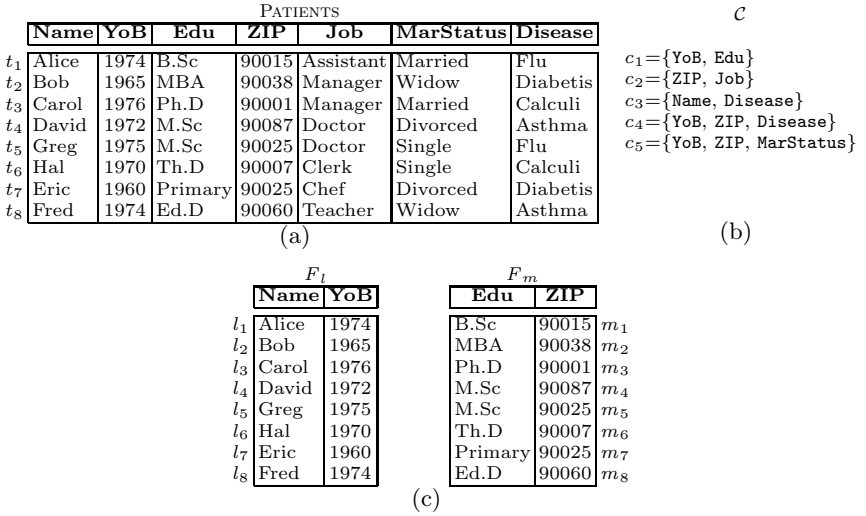


Fig. 1. An example of relation (a), a set $\mathcal{C}$ of confidentiality constraints over it (b), and a minimal fragmentation that satisfies the constraints in $\mathcal{C}$ (c)

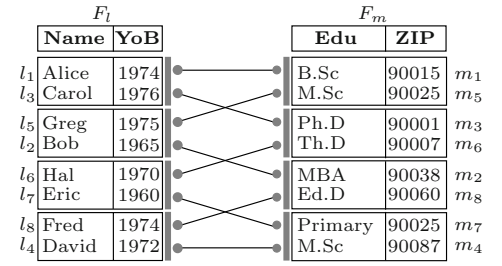
that should not be revealed by the release of $\mathcal{F}$ are modeled through *confidentiality constraints* [1].

Definition 1 (Confidentiality constraint). *Given a relation schema S, a* confidentiality constraint *c over S is a subset of the attributes in S.*

Figure 1(b) illustrates a set $\mathcal{C}$ of confidentiality constraints defined over relation PATIENTS in Figure 1(a). A fragmentation can be published only if it satisfies all the confidentiality constraints, that is, only if it does not disclose sensitive associations, neither directly in a single fragment (i.e., $\forall F \in \mathcal{F}$, $\forall c \in \mathcal{C} : c \not\subseteq F$), nor indirectly by joining fragments (i.e., fragments are disjoint, $\forall F_i, F_j \in \mathcal{F}$, $i \neq j$: $F_i \cap F_j = \emptyset$). In our discussion, we assume the released fragmentation to be *minimal*, meaning that merging fragments in $\mathcal{F}$ would violate at least a confidentiality constraint. This is in line with the idea that the data owner does not fragment relation S more than necessary. Figure 1(c) illustrates an example of minimal fragmentation of relation PATIENTS in Figure 1(a), which satisfies the constraints in Figure 1(b).

To mitigate information loss caused by the fact that fragmentation breaks the associations in the original relation, fragments can be complemented with *loose associations*, introduced in [9] for fragmentations composed of a single pair of fragments. To this aim, tuples in the two fragments are partitioned in groups, and information on the associations between tuples in fragments is released at the group (in contrast to tuple) level.

Given a fragmentation $\mathcal{F}=\{F_l, F_m\}$ and its instance $\{f_l, f_m\}$, tuples in f_l and f_m are first independently partitioned in groups of size at least k_l and k_m, respectively. To this aim, the data owner defines a *k-grouping* function for each



(a)

F_l

	Name	YoB	G_l
l_1	Alice	1974	ny1
l_2	Bob	1965	ny2
l_3	Carol	1976	ny1
l_4	David	1972	ny4
l_5	Greg	1975	ny2
l_6	Hal	1970	ny3
l_7	Eric	1960	ny3
l_8	Fred	1974	ny4

A

G_l	G_m
ny1	ez1
ny1	ez2
ny2	ez1
ny2	ez3
ny3	ez2
ny3	ez4
ny4	ez3
ny4	ez4

F_m

G_m	Edu	ZIP	
ez1	B.Sc	90015	m_1
ez3	MBA	90038	m_2
ez2	Ph.D	90001	m_3
ez4	M.Sc	90087	m_4
ez1	M.Sc	90025	m_5
ez2	Th.D	90007	m_6
ez4	Primary	90025	m_7
ez3	Ed.D	90060	m_8

(b)

Fig. 2. Graphical representation (a) and corresponding relations (b) of a 4-loose association between fragments F_l and F_m in Figure 1(c)

of the two fragments. A k-grouping function partitions tuples in a fragment instance f in groups of size at least k, by associating a group identifier with each tuple in f [9].

Definition 2 (k-Grouping). *Given a fragment F_i, its instance f_i, and a set GID_i of group identifiers, a* k-grouping *function over f_i is a surjective function $\mathcal{G}_i{:}f_i{\rightarrow}\mathsf{GID}_i$ such that $\forall g_i \in \mathsf{GID}_i :\mid \mathcal{G}_i^{-1}(g_i) \mid \geq k$.*

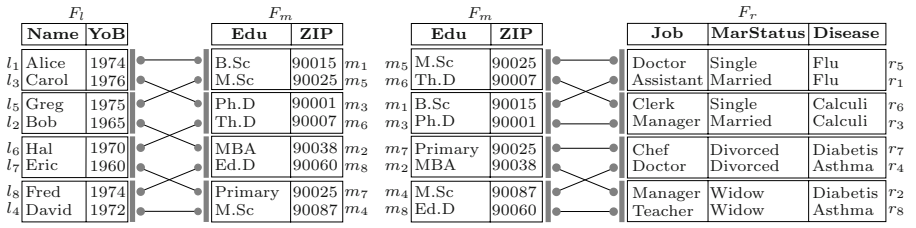
Notation (k_l,k_m)-grouping denotes a k_l-grouping over f_l and a k_m-grouping over f_m (note that k_l may be different from k_m). Once each tuple in f_l and in f_m is associated with a group identifier, the group-level relationships between tuples in f_l and in f_m are represented by an additional relation A. For each tuple t in the original relation, relation A includes a tuple containing the group where $t[F_l]$ appears in f_l and the group where $t[F_m]$ appears in f_m. For instance, Figure 2(a) represents a partition in groups of size $k_l{=}k_m{=}2$ of the tuples in fragments f_l and f_m in Figure 1(c). For simplicity, given a tuple t in the original relation, we denote with l (m, resp.) the sub-tuple $t[F_l]$ ($t[F_m]$, resp.) in fragment f_l (f_m, resp.). The edges connecting grey dots, which correspond to tuples in groups, represent the group-level associations between tuples in the two fragments implied by relation Patients in Figure 1(a). Figure 2(b) illustrates relation A and fragments F_l and F_m enriched with an attribute ($\mathtt{G}_l$ and $\mathtt{G}_m$, resp.) reporting the identifier of the group to which each tuple belongs.

The partitioning of the tuples in the two fragments should be carefully designed to guarantee that sensitive associations cannot be reconstructed exploiting A. Intuitively, a loose association between a pair of fragments $\{F_l, F_m\}$ enjoys a degree k of protection (referred to as k-looseness) if every tuple in A indistinguishably corresponds to at least k distinct associations among tuples in f_l and f_m with different values for the attributes involved in each confidentiality constraint c such that $c \subseteq F_l \cup F_m$. In fact, the release of a loose association between F_l and F_m only puts at risk constraints whose attributes are all represented by the two fragments. For instance, the first tuple in table A in Figure 2(b) corresponds to four possible tuples (i.e., $\langle l_1, m_1 \rangle$, $\langle l_1, m_5 \rangle$, $\langle l_3, m_1 \rangle$, $\langle l_3, m_5 \rangle$), all with different values for attributes `YoB` and `Edu` composing constraint c_1, which is the only constraint completely covered by F_l and F_m. In other words, the release of F_l, F_m, and A satisfies *k-looseness* for each $k \leq k_l \cdot k_m$, if for each group g_l in f_l (g_m in f_m, resp.), the union of the tuples in all the groups with which g_l (g_m, resp.) is associated in A is a set of at least k tuples with different values for the attributes in each constraint $c \subseteq F_l \cup F_m$ [9]. For instance, the association in Figure 2 satisfies k-looseness for any $k \leq 4$.

3 Problem and Motivating Example

Although effective for publishing loose associations between pairs of fragments, the proposal in [9] cannot be directly applied to the release of multiple loose associations between different pairs of fragments, since they might disclose sensitive associations. To illustrate the problem, consider a fragmentation $\mathcal{F}$ composed of 3 fragments, say F_l, F_m, and F_r. A straightforward approach to release group-level associations among these fragments consists in releasing two distinct loose associations between two pairs of fragments in $\mathcal{F}$ (e.g., one between F_l and F_m, and one between F_m and F_r). For instance, consider a fragmentation of relation Patients in Figure 1(a) that satisfies the constraints in Figure 1(b), composed of 3 fragments F_l={`Name`, `YoB`}, F_m={`Edu`, `ZIP`}, and F_r={`Job`, `MarStatus`, `Disease`}. Figure 3 illustrates a 4-loose association between F_l and F_m (A_{lm}), and a 4-loose association between F_m and F_r (A_{mr}) (note that tuples in f_m are partitioned according to two different grouping functions, one for each loose association).

Such an approach clearly releases useful information on the associations between the tuples in F_l and F_m, and between the tuples in F_m and F_r. The loose associations between F_l and F_m, and between F_m and F_r imply however an *induced association* between F_l and F_r: F_l can be loosely joined with F_m, which in turn can be loosely joined with F_r. Therefore, each tuple in f_l is associated with a group of tuples in f_m, each of which is in turn associated with a group of tuples in f_r. As an example, tuple l_7 in fragment f_l in Figure 3 is associated with tuples m_3, m_4, m_6, and m_7 in fragment f_m. In turn, m_3 and m_6 are associated with r_1, r_3, r_5, and r_6 in f_r. Tuples m_4 and m_7 are instead associated with r_2, r_4, r_7, and r_8 in f_r. Therefore, l_7 is possibly associated with any tuple in f_r. The induced association between F_l and F_r might then seem to enjoy a



(a)

F_l

	Name	YoB	G_l
l_1	Alice	1974	ny1
l_2	Bob	1965	ny2
l_3	Carol	1976	ny1
l_4	David	1972	ny4
l_5	Greg	1975	ny2
l_6	Hal	1970	ny3
l_7	Eric	1960	ny3
l_8	Fred	1974	ny4

F_m

	G_{m1}	Edu	ZIP	G_{m2}
m_1	ez11	B.Sc	90015	ez22
m_2	ez13	MBA	90038	ez23
m_3	ez12	Ph.D	90001	ez22
m_4	ez14	M.Sc	90087	ez24
m_5	ez11	M.Sc	90025	ez21
m_6	ez12	Th.D	90007	ez21
m_7	ez14	Primary	90025	ez23
m_8	ez13	Ed.D	90060	ez24

F_r

	G_r	Job	MarStatus	Disease
r_1	jmd1	Assistant	Married	Flu
r_2	jmd4	Manager	Widow	Diabetis
r_3	jmd2	Manager	Married	Calculi
r_4	jmd3	Doctor	Divorced	Asthma
r_5	jmd1	Doctor	Single	Flu
r_6	jmd2	Clerk	Single	Calculi
r_7	jmd3	Chef	Divorced	Diabetis
r_8	jmd4	Teacher	Widow	Asthma

A_{lm}

G_l	G_{m1}
ny1	ez11
ny1	ez12
ny2	ez11
ny2	ez13
ny3	ez12
ny3	ez14
ny4	ez13
ny4	ez14

A_{mr}

G_{m2}	G_r
ez21	jmd1
ez21	jmd2
ez22	jmd1
ez22	jmd2
ez23	jmd3
ez23	jmd4
ez24	jmd3
ez24	jmd4

(b)

Fig. 3. Graphical representation (a) and corresponding relations (b) of a 4-loose association A_{lm} between F_l and F_m, and a 4-loose association A_{mr} between F_m and F_r, with F_l, F_m, and F_r three fragments of relation PATIENTS in Figure 1(a)

protection degree equal to (or even greater than) those enjoyed by A_{lm} and A_{mr}. However, publishing loose associations A_{lm} and A_{mr} guarantees that sensitive associations involving only attributes in F_l and F_m, and only attributes in F_m and F_r are protected. It does not provide any guarantee on the protection of sensitive associations involving attributes stored in F_l and in F_r, which are possibly exposed by the induced association. This is due to the fact that the loose association between F_l and F_m requires tuples in f_l (f_m, resp.) associated with each group in f_m (f_l, resp.) to have different values for the attributes appearing in constraints $c \subseteq F_l \cup F_m$ (c_1, in our example). Analogously, the loose association between F_m and F_r requires tuples in f_m (f_r, resp.) associated with each group in f_r (f_m, resp.) to have different values for the attributes appearing in constraints $c \subseteq F_m \cup F_r$ (c_2, in our example). Constraints $c \subseteq F_l \cup F_m \cup F_r$ such that $c \cap F_l \neq \emptyset$ and $c \cap F_r \neq \emptyset$ (c_3, c_4, c_5, in our example) are instead ignored. To illustrate, the release of the fragments and loose associations in Figure 3 exposes the sensitive association between attributes `Name` and `Disease`, violating

constraint c_3 in Figure 1(b). In fact, tuple l_1 in f_l is associated with tuples m_1, m_3, m_5, and m_6 in f_m. In turn, m_1, m_3, m_5, and m_6 in f_m are all associated with tuples r_1, r_3, r_5, and r_6 in f_r. Thus, the observation of A_{lm} and A_{mr} reveals that l_1 is associated, in the original relation, with one among r_1, r_3, r_5, and r_6, but r_1[Disease]=r_5[Disease]=Flu and r_3[Disease]=r_7[Disease]=Calculi. Therefore, either association ⟨Alice,Flu⟩ or association ⟨Alice,Calculi⟩ belongs to relation PATIENTS with the same probability. The degree of protection for constraint c_3 offered by the release of the two loose associations in Figure 3 is then 2 (and not 4 as for constraints c_1 and c_2). Note that the release of arbitrary loose associations may completely expose sensitive associations. For instance, assume that r_3[Disease]=r_6[Disease]=Flu. The released associations would still be 4-loose, but they reveal that Alice suffers from Flu.

The privacy breach described above represents a serious issue for the data owner since it exposes sensitive associations that she is not explicitly publishing. She could then be unaware of the fact that the released fragments and loose associations expose sensitive associations. In the remainder of this paper, we illustrate our proposal for counteracting such a privacy problem. Our intuition is to define a single loose association encompassing all the fragments among which the data owner needs to publish group-level associations. In this way, we aim at defining one loose association only that takes into consideration *all* the confidentiality constraints among attributes stored by the released fragments. Intuitively, since all the published fragments are involved in the same loose association, publishing this association does not imply the disclosure of induced associations that can be exploited by malicious recipients to precisely reconstruct sensitive associations. As we will detail in the following sections, starting from this loose association, the data owner may then choose to either release it as a whole, or use it to build an arbitrary set of loose associations, with the guarantee that no sensitive association be improperly exposed.

4 Loose Associations

Given a fragmentation $\mathcal{F}$ of a relation S and a set $\mathcal{C}$ of confidentiality constraints over S, we define a loose association among the fragments in $\mathcal{F}$ (note that our approach also permits to define a loose association among an arbitrary subset of fragments in $\mathcal{F}$). For the sake of readability, we refer the discussion to a fragmentation $\mathcal{F}$=$\{F_l, F_m, F_r\}$ composed of 3 fragments, while definitions are formulated on fragmentations composed of an arbitrary number of fragments. In line with previous works on fragmentation [1,4,9], we assume that data recipients do not possess any additional knowledge besides released fragments, loose associations, and confidentiality constraints defined by the data owner.

The first step necessary for the definition of a loose association among the fragments in $\mathcal{F}$ is the identification of the subset of confidentiality constraints in $\mathcal{C}$ that are *relevant* for $\mathcal{F}$. A constraint is relevant for a set $\{F_1, \ldots, F_n\}$ of fragments if it includes only attributes represented by the fragments in $\{F_1, \ldots, F_n\}$. Indeed, any other constraint cannot be violated by the release of a loose association among fragments in $\{F_1, \ldots, F_n\}$.

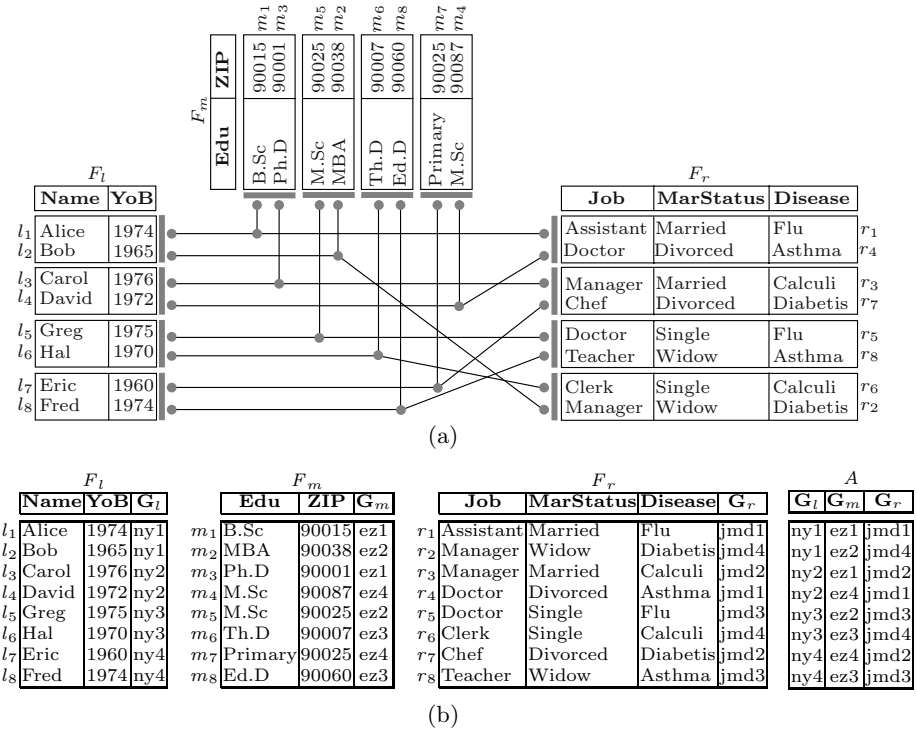


Fig. 4. Graphical representation (a) and corresponding relations (b) of a 4-loose association among three fragments F_l, F_m, and F_r of relation PATIENTS in Figure 1(a)

Definition 3 (Relevant constraints). *Given a set $T = \{F_1, \ldots, F_n\}$ of fragments and a set $\mathcal{C}$ of confidentiality constraints, the set $\mathcal{C}_T$ of* relevant constraints *for T is defined as $\mathcal{C}_T = \{c \in \mathcal{C} : c \subseteq F_1 \cup \ldots \cup F_n\}$.*

For instance, the only constraint in Figure 1(b) relevant for the set of fragments in Figure 1(c) is c_1 as it is the only constraint whose attributes belong to the set {`Name`, `YoB`, `Edu`, `ZIP`}.

Given a fragmentation $\mathcal{F}=\{F_1, \ldots, F_n\}$, the tuples in each fragment are partitioned according to different grouping functions, which may adopt different protection parameters (thus generating groups of different size). A $(k_1, \ldots, k_n)$-grouping is a set $\{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$ of grouping functions defined over fragments in $\{f_1, \ldots, f_n\}$ (i.e., a set of k_i-groupings over f_i, i=1, …, n). As an example, Figure 4 illustrates a (2,2,2)-grouping involving fragments F_l={`Name`, `YoB`}, F_m={`Edu`, `ZIP`} and F_r={`Job`, `MarStatus`, `Disease`} of relation PATIENTS in Figure 1(a), and the corresponding group association. It is easy to see that tuple t_1 in relation PATIENTS is represented in fragments F_l, F_m, and F_r by tuples l_1, m_1, and r_1, respectively. The association among l_1, m_1, and r_1 is represented

by tuple ⟨ny1,ez1,jmd1⟩ in A, which defines an association among the groups to which l_1, m_1, and r_1 belong.

A group association A can be safely released only if it cannot be exploited to reconstruct, totally or in part, sensitive associations among the released fragments. A (k_l,k_m,k_r)-grouping guarantees that each tuple in A corresponds to $k_l \cdot k_m \cdot k_r$ different associations among tuples in f_l, f_m, and f_r. However, some tuples represented by these $k_l \cdot k_m \cdot k_r$ associations might have the same values for the attributes in a relevant constraint, thus reducing in practice the protection degree enjoyed by the published group association. To guarantee that a group association A does not expose relevant confidentiality constraints, each tuple in A must refer to k distinct associations among sub-tuples in fragments that do not have the same values for the attributes in relevant constraints. A group association satisfying this property is said to be *k-loose*. To compare the values assumed in fragments by the attributes in relevant constraints, we formally introduce the *alike* relationship between tuples as follows.

Definition 4 (Alike). *Given a fragmentation $\mathcal{F}=\{F_1,\ldots,F_n\}$ with its instance $\{f_1,\ldots,f_n\}$, and the set $\mathcal{C}_\mathcal{F}$ of confidentiality constraints relevant for $\mathcal{F}$, $t_i,t_j \in f_z$, $z=1,\ldots,n$, are said to be* alike *with respect to a constraint $c \in \mathcal{C}_\mathcal{F}$, denoted $t_i \simeq_c t_j$ iff $c \cap F_z \neq \emptyset \wedge t_i[c \cap F_z] = t_j[c \cap F_z]$. Two tuples are said to be* alike *with respect to a set $\mathcal{C}_\mathcal{F}$ of relevant constraints, denoted $t_i \simeq_{\mathcal{C}_\mathcal{F}} t_j$, if they are alike with respect to at least one constraint $c \in \mathcal{C}_\mathcal{F}$.*

Definition 4 states that given a fragmentation $\mathcal{F}$, two tuples in a fragment instance f_i are alike if they have the same values for the attributes in a constraint relevant for $\mathcal{F}$. For instance, with reference to the (2,2,2)-grouping in Figure 4, $r_4 \simeq_{c_3} r_8$ since $r_4[\texttt{Disease}]=r_8[\texttt{Disease}]=$Asthma. Since we are interested in evaluating the alike relationship w.r.t. the set $\mathcal{C}_\mathcal{F}$ of relevant constraints, in the following we omit the subscript of the alike relationship whenever clear from the context (i.e., we write $t_i \simeq t_j$ instead of $t_i \simeq_{\mathcal{C}_\mathcal{F}} t_j$). The alike relationship guides the definition of k-loose group associations among arbitrary sets of fragments, as formally defined in the following.

Definition 5 (k-Looseness). *Given a fragmentation $\mathcal{F} = \{F_1,\ldots,F_n\}$ with its instance $\{f_1,\ldots,f_n\}$, the set $\mathcal{C}_\mathcal{F}$ of confidentiality constraint relevant for $\mathcal{F}$, and a group association A over $\{f_1,\ldots,f_n\}$, A is said to be* k-loose *w.r.t. $\mathcal{C}_\mathcal{F}$ iff $\forall c \in \mathcal{C}_\mathcal{F}$, $\forall F_i \in \mathcal{F} : c \cap F_i \neq \emptyset$ and $\forall g_i \in \mathsf{GID}_i$, $\exists F_j \in \mathcal{F} : c \cap F_j \neq \emptyset$ that satisfies the following condition: let $T = \bigcup_z \{\mathcal{G}_j^{-1}(g_z) \mid (g_i,g_z) \in A[G_i,G_j]\} \Longrightarrow \mid T \mid \geq k$, and $\forall t_x,t_y \in T$, $x \neq y$, $t_x \not\simeq_c t_y$.*

k-Looseness guarantees that sensitive associations represented by relevant constraints cannot be reconstructed with confidence higher than $1/k$. According to the definition above, a group association A is k-loose if each tuple in A corresponds to k possible tuples in the original relation that are not alike w.r.t. any relevant constraint. Note that, however, there are cases in which the alike requirement can be relaxed. In fact, whenever a value v in the domain of an attribute is considered not sensitive (e.g., because it characterizes the majority

of the tuples in the original relation), the alike relationship may consider such a value as *neutral*. In this case, even if t_i[Attr]=t_j[Attr]=v, $t_i \not\approx t_j$.

The definition of k-looseness translates into the satisfaction of a different condition depending on whether the considered constraint involves two (like in [9]) or more fragments.

- *Constraints between two fragments.* k-Looseness requires that, for each group g_l in f_l, the union of the tuples in all the groups g_m in f_m with which g_l is associated is a set including at least k tuples that are not alike w.r.t. c (and viceversa). With reference to the example in Figure 4, c_1 cannot be reconstructed since each group in f_l is associated with two different groups in f_m including tuples that do not contain duplicates for Edu and viceversa.
- *Constraints among more than two fragments.* k-Looseness requires to break the association between at least two of the fragments storing attributes in c to guarantee that the sensitive association represented by c cannot be reconstructed. We then need to guarantee that, for each group g_l in f_l, the union of the tuples in all the groups g_m in f_m with which g_l is associated *or* the union of the tuples in all the groups g_r in f_r with which g_l is associated is a set of at least k tuples that are not alike w.r.t. c. Clearly, this property must hold also for each group g_m in f_m and for each group g_r in f_r. For instance, consider the fragments and group association in Figure 4 and constraint c_5 over them. Sensitive associations among YoB, ZIP, and MarStatus cannot be reconstructed even if group ny2 in f_l is associated with groups jmd1 and jmd2 in f_r whose tuples have the same values for attribute MarStatus. In fact, group ny2 is associated with groups ez1 and ez4 in f_m, which do not include tuples that are alike w.r.t. c_5 (i.e., tuples in ez1 and ez4 have all different values for ZIP).

This definition of k-looseness implies that the release of a (k_l,k_m,k_r)-grouping induces a k-loose association with k=$min(k_l \cdot k_m, k_m \cdot k_r, k_l \cdot k_r)$. In fact, the constraints relevant for $\{F_l,F_m\}$ ($\{F_m,F_r\}$ and $\{F_l,F_r\}$, resp.) enjoy a protection degree k_{lm}=$k_l \cdot k_m$ (k_{mr}=$k_m \cdot k_r$ and k_{lr}=$k_l \cdot k_r$, resp.). Constraints relevant for $\{F_l,F_m,F_r\}$ enjoy the minimum protection degree among k_{lm}, k_{mr}, and k_{lr} since, as illustrated above, it is not required that all the associations among the attributes in the constraints be broken. Figure 4(b) illustrates the 4-loose association induced by the (2,2,2)-grouping in Figure 4(a). This association guarantees the same protection degree k=k_{lm}=k_{mr}=k_{lr}=4 to each pair of fragments (and then also to $\mathcal{F}$).

5 Heterogeneity Properties

In this section, we enhance and extend the heterogeneity properties (i.e., group, association, and deep heterogeneity), originally proposed to guarantee that a group association between two fragments is k-loose, to provide the same guarantee to group associations defined on an arbitrary number of fragments.

Group Heterogeneity. This property guarantees that groups do not include tuples with the same values for the attributes in relevant constraints. In this way, the minimum size k_i of the groups in fragment F_i, $i = 1, \ldots, n$, reflects the minimum number of different values in the group for each subset of attributes that appear together in a relevant constraint.

Property 1 (Group heterogeneity). Given a fragmentation $\mathcal{F} = \{F_1, \ldots, F_n\}$ with its instance $\{f_1, \ldots, f_n\}$, and the set $\mathcal{C}_\mathcal{F}$ of constraints relevant for $\mathcal{F}$, grouping functions $\mathcal{G}_i$ over f_i, $i = 1, \ldots, n$, satisfy *group heterogeneity* iff $\forall f_i \in \{f_1, \ldots, f_n\}$, $\forall t_z, t_w \in f_i$: $t_z \simeq t_w \implies \mathcal{G}_i(t_z) \neq \mathcal{G}_i(t_w)$.

The definition of this property is similar to the one operating on two fragments, as it is local to the tuples in each fragment. It however operates on a different set of constraints, that is, the set of constraints relevant for $\mathcal{F}$. For instance, in Figure 4 the grouping functions defined for the three fragments satisfy group heterogeneity for $\mathcal{C}_\mathcal{F} = \{c_1, \ldots, c_5\}$. On the contrary, the groupings for the three fragments in Figure 3 do not satisfy group heterogeneity for $\mathcal{F} = \{F_l, F_m, F_r\}$ since, for example, $r_1 \simeq_{c_3} r_5$ and they belong to the same group. However, these groupings satisfy group heterogeneity for $\mathcal{F}_1 = \{F_l, F_m\}$ (where c_1 is the only relevant constraint) and for $\mathcal{F}_2 = \{F_m, F_r\}$ (where c_2 is the only relevant constraint).

Association Heterogeneity. For loose associations between two fragments, this property requires that A cannot have duplicates. This simple condition is however not sufficient in our (more general) scenario. In fact, association heterogeneity aims at guaranteeing that, for each constraint c in $\mathcal{C}_\mathcal{F}$, each group in f_i is associated with at least k_i different groups in *at least* one of the fragments storing attributes in c (i.e., groups in f_j such that $c \cap F_j \neq \emptyset$). If a group in f_i is associated with one group in f_j only, it is easier for an observer to reconstruct the correct associations among the tuples in these two groups (and therefore to violate constraints). This condition implies that A cannot have two tuples with the same group identifier for all the fragments storing attributes composing a constraint (for constraints involving more than two fragments, it is sufficient that one of the values in the tuple be different).

Since we consider minimal fragmentations, there exists at least one relevant constraint for each pair of fragments in $\mathcal{F}$ (i.e., $\forall \{f_i, f_j\} \subseteq \mathcal{F}$, $i \neq j$, $\exists c \in \mathcal{C}$ s.t. $c \subseteq F_i \cup F_j$, Theorem A.2 in [9]). Therefore, a group association A satisfies association heterogeneity if it does not have two tuples with the same group identifier for any pair of fragments in $\mathcal{F}$.

Property 2 (Association heterogeneity). A group association A satisfies *association heterogeneity* iff $\forall (g_{i_1}, \ldots, g_{i_n}), (g_{j_1}, \ldots, g_{j_n}) \in A$: $i_z = j_z \implies i_w \neq j_w$, $w = 1, \ldots, n$ and $w \neq z$.

Intuitively, association heterogeneity requires that the projection over A of any subset of two attributes does not contain duplicate tuples. It is immediate to see that the group association in Figure 4 satisfies association heterogeneity.

Deep Heterogeneity. This property guarantees that a group in f_i cannot be associated with different groups in f_j including duplicated values for the attributes in a relevant constraint $c \subseteq F_i \cup F_j$. The groups in f_j with which a group in f_i is associated may be composed of tuples with exactly the same values for the attributes in c, limiting the protection offered by the loose association. For instance, groups jmd1 and jmd3 in Figure 4 have the same values for attribute `Disease` (i.e., Flu and Asthma). Therefore, a group in f_l cannot be associated with both jmd1 and jmd3 because of constraint c_3 (otherwise, the association between F_l and F_r would be 2-loose instead of 4-loose).

Deep heterogeneity imposes diversity by looking at the values behind the groups. The definition of deep heterogeneity over pairs of fragments requires that the groups in fragment f_i with which a group in f_j is associated in A do not contain alike tuples. A straightforward approach to extend deep heterogeneity would require diversity over all the fragments storing the attributes composing the constraint. In other words, considering a constraint c composed of attributes stored in fragments $\{F_1, \ldots, F_n\}$, all the groups in each fragment f_i $(i = 1, \ldots, n)$ with which a group in f_j $(j = 1, \ldots, n,\ i \neq j)$ is associated in A should not contain tuples that are alike w.r.t. c. This condition is more restrictive than necessary to define a k-loose association. In fact, it is sufficient, for each fragment F_j, to break the association with *one* of the fragments F_i $(i = 1, \ldots, n,\ i \neq j)$ storing the attributes in c. For instance, with reference to the example in Figure 4, it is sufficient that each group in f_l be associated with groups of non-alike tuples in either f_m or f_r to guarantee that the sensitive association modelled by c_5 is not exposed.

Property 3 (Deep heterogeneity). Given a fragmentation $\mathcal{F} = \{F_1, \ldots, F_n\}$ with its instance $\{f_1, \ldots, f_n\}$, and the set $\mathcal{C}_\mathcal{F}$ of constraints relevant for $\mathcal{F}$, a group association A over $\mathcal{F}$ satisfies *deep heterogeneity* iff $\forall c \in \mathcal{C}_\mathcal{F}$; $\forall F_z \in \mathcal{F}$, $F_z \cap c \neq \emptyset$; $\forall\ (g_{i_1}, g_{i_2}\ \ldots\ g_{i_n}), (g_{j_1}, g_{j_2} \ldots\ g_{j_n}) \in A$ the following condition is satisfied:

$$i_w = j_w \Longrightarrow \bigvee_{l=1,\ldots,n,\ l \neq w} \nexists t_x, t_y\colon t_x \in \mathcal{G}_l^{-1}(g_{i_l}),\ t_y \in \mathcal{G}_l^{-1}(g_{j_l}),\ t_x \simeq_c t_y.$$

Given a constraint c whose attributes appear in fragments $\{F_{i_1}, \ldots, F_{i_j}\}$, deep heterogeneity is satisfied w.r.t. c if the set of tuples in the groups $\{g_{x_1}, \ldots, g_{x_w}\}$ in f_{i_x} with which a group g_y in f_{i_y} is associated are not alike w.r.t. c, for at least one fragment f_{i_x}, $x = 1, \ldots, j$ and $x \neq y$. This property must be true for all the groups in each fragment F_{i_x}, $x = 1, \ldots, j$. This guarantees that, for each constraint, no association can be precisely reconstructed by an observer. An example of group association that satisfies deep heterogeneity is illustrated in Figure 4. Note that deep heterogeneity is satisfied even though group ny2 in f_l is associated with groups jmd1 and jmd2 in f_r, which include tuples $r_1 \simeq_{c_5} r_3$ and $r_4 \simeq_{c_5} r_7$. In fact, group ny2 is also associated with groups ez1 and ez4 in f_m that do not include tuples that are alike w.r.t. c_5 (i.e., with the same value for `ZIP`).

If the three properties above are satisfied by a $(k_1, \ldots, k_n)$-grouping and its induced group association, the group association is k-loose with $k \leq min(k_i \cdot k_j)$

$\forall i, j = 1, \ldots, n$, $i \neq j$, as stated by the following theorem (the proof has been omitted for space constraints).

Theorem 1. *Given a fragmentation $\mathcal{F} = \{F_1, \ldots, F_n\}$ with its instance $\{f_1, \ldots, f_n\}$, the set $\mathcal{C}_\mathcal{F}$ of constraints relevant for $\mathcal{F}$, and a $(k_1, \ldots, k_n)$-grouping that satisfies Properties 1, 2, and 3, the group association A induced by the $(k_1, \ldots, k_n)$-grouping is k-loose w.r.t. $\mathcal{C}_\mathcal{F}$ (Definition 5) for each $k \leq \min(k_i \cdot k_j)$, with $i, j = 1, \ldots, n$, $i \neq j$.*

As a consequence of the above theorem, the protection degree that a $(k_1, \ldots, k_n)$-grouping that satisfies Properties 1, 2, and 3 offers may be different for each confidentiality constraint c in $\mathcal{C}_\mathcal{F}$. Indeed, the protection degree for a constraint c is $min(k_i \cdot k_j)$, where $F_i, F_j \in \{F \in \mathcal{F} : F \cap c \neq \emptyset\}$.

In this paper, for space constraints, we do not discuss how to compute a k-loose association among an arbitrary set of fragments. We note however that the solution in [9] can be extended to our scenario, by properly modifying the enforcement of the above heterogeneity properties.

6 Discussion

The consideration of all the constraints relevant for the fragments involved in the loose association guarantees that no constraint can be violated. Thus, our loose association defined over an arbitrary set of fragments does not suffer from the confidentiality breach illustrated in Section 3, mainly caused by the fact that confidentiality constraints relevant for the fragments involved in induced associations are ignored. As an example, with reference to the 4-loose association in Figure 4, each tuple in A corresponds to four different associations of (different) values for attributes `Name` and `Disease`. This guarantees that constraint c_3 is satisfied, while it is violated by the example in Figure 3.

The release of a k-loose association among a set $\mathcal{F}$ of fragments is equivalent to the release of $2^n - n$, with $n = |\mathcal{F}|$, k-loose associations (one for each subset of fragments in $\mathcal{F}$). Indeed, the projection over a subset of attributes in A represents a k-loose association for the fragments corresponding to the projected attributes.

Observation 1. *Given a fragmentation $\mathcal{F}=\{F_1, \ldots, F_n\}$, a subset $\{F_i, \ldots, F_j\}$ of $\mathcal{F}$, and a k-loose association $A(\mathtt{G}_1, \ldots, \mathtt{G}_n)$ over $\mathcal{F}$, group association $A'(\mathtt{G}_i, \ldots, \mathtt{G}_j) = \pi_{(\mathtt{G}_i, \ldots, \mathtt{G}_j)}(A)$ is a k-loose association over $\{F_i, \ldots, F_j\}$.*

For instance, with reference to the 4-loose association in Figure 4, the projection of attributes $\mathtt{G}_l$, $\mathtt{G}_m$ in A is a 4-loose association between F_l and F_m.

Since a k-loose association defined over a set $\mathcal{F}$ of fragments guarantees that sensitive associations represented by constraints in $\mathcal{C}_\mathcal{F}$ are properly protected, the release of multiple loose associations among arbitrary (and possibly overlapping) subsets of fragments in $\mathcal{F}$ provides the data owner with the same protection guarantee. The data owner can therefore decide to release either one loose association A encompassing the associations among the fragments in $\mathcal{F}$, or a subset of loose associations defined among arbitrary subsets of fragments in $\mathcal{F}$ by projecting the corresponding attributes from A.

Observation 2. *Given a fragmentation* $\mathcal{F}=\{F_1,\ldots,F_n\}$ *and a* k*-loose association* $A(\mathtt{G}_1,\ldots,\mathtt{G}_n)$ *over it, the release of an arbitrary set of* k*-loose associations* $\{A_1(\mathtt{G}_h,\ldots,\mathtt{G}_i),\ldots,A_m(\mathtt{G}_j,\ldots,\mathtt{G}_k)\}$ *with* $\{\mathtt{G}_h,\ldots,\mathtt{G}_k\}\subseteq\{\mathtt{G}_1,\ldots,\mathtt{G}_n\}$ *provides the same protection guarantee as the release of* A*.*

For instance, with reference to our examples above, aiming at releasing two distinct 4-loose associations, the data owner can release the 4-loose associations obtained projecting $\langle \mathtt{G}_l,\mathtt{G}_m\rangle$ and $\langle \mathtt{G}_m,\mathtt{G}_r\rangle$ from the 4-loose association in Figure 4. This solution does not suffer from the privacy breach illustrated in Section 3, while providing associations between groups of the same size (i.e., the same utility for data recipients).

The two observations above need to be considered if the data owner is interested in releasing more than one loose association among arbitrary subsets of fragments in $\mathcal{F}$. On the contrary, if the loose associations of interest operate on disjoint subsets of fragments (i.e., no fragment is involved in more than one loose association), they can be defined independently from each other without risks of unintended disclosure of sensitive associations.

Observation 3. *Given a fragmentation* $\mathcal{F}$*, and a set* $\{\mathtt{F}_1,\ldots,\mathtt{F}_n\}$ *of subsets of fragments in* $\mathcal{F}$ *(i.e..,* $\mathtt{F}_i\subseteq\mathcal{F}$*,* $i=1,\ldots,n$*), the release of* n *loose associations* A_i*,* $i=1,\ldots,n$ *is* safe *if* $\forall i,j=1,\ldots,n$ *with* $i\neq j$*,* $\mathtt{F}_i\cap\mathtt{F}_j=\emptyset$*.*

7 Related Work

Several research efforts have addressed the problem of protecting privacy in data publishing, proposing approaches based on either sanitizing (e.g., [6,10,11,13,14,15,16,20]) or fragmenting data (e.g., [1,2,4,5,7]) before their release. Our approach provides the same privacy guarantees as the well-known k-anonymity and ℓ-diversity (with $\ell=k$) approaches, while releasing complete and truthful information thanks to the adoption of a different protection technique. Most fragmentation works, although showing similarities with our proposal, address a problem orthogonal to ours, as they aim at breaking sensitive associations while maximizing the ability of recipients of evaluating queries on fragments.

The works closest to ours complement fragmented data with information on their association, without disclosing sensitive information [7,9,21]. Our proposal is however more general, since these solutions operate on two fragments only, while we consider an arbitrary number of fragments when defining loose associations. Anatomy [21] considers the specific problem of protecting the association between respondents' identities and a sensitive attribute while our solution permits to protect any association among attributes. Also, Anatomy partitions the original relation in groups of ℓ tuples before splitting attributes in two disjoint fragments. Hence, it can be considered a specific instance of our approach where $k_l=1$ and $k_r=\ell$ (or viceversa).

The work in [7] does not take into consideration the possible existence of duplicate values for non-key attributes, exposing therefore to frequency-based attacks sensitive associations on non-key attributes. Our heterogeneity properties

overcome this issue, by preventing the presence of duplicates in the same group of tuples.

Our work may bring some resemblance with the proposals in [3,18,19]. The work in [19] adopts horizontal and vertical fragmentation to protect privacy of sparse multidimensional data (e.g., transactional data). The approach in [3] focuses instead on protecting recommendation data expressed by customers (i.e., Netflix movie ratings). Besides operating on different data models, both these proposals differ from our work since they are specifically targeted at protecting respondents' identities and their association with sensitive attributes. Also, they both adopt a dual approach with respect to loose associations, requiring homogeneity of values in fragments. The work in [18] addresses the problem of destroying the correlation between two disjoint subsets of attributes, preserving as much as possible the other correlations. Our approach does not aim at destroying correlations among attributes, as our goal is to preserve them as much as possible, while satisfying privacy constraints. Also, the solution in [18] adopts masking techniques, while our approach maintains data truthfulness.

Alternative approaches to protect privacy in data release are based on differential privacy [10,11]. Although addressing a similar problem, differential privacy cannot be directly applied to the considered scenario. In fact, all these approaches introduce noise in the dataset that depends on the expected users queries. Our approach instead does not make assumptions on users queries and aims at releasing truthful data.

8 Conclusions

We presented an approach for extending the definition of loose association to multiple fragments. We first described the exposure risks that characterize the release of multiple loose associations between pairs of fragments, and then presented an approach supporting the definition of a loose association among an arbitrary number of fragments. We also discussed some properties of the proposed solution.

Acknowledgements. This work was supported in part by the EC within the 7FP under grant agreement 257129 (PoSecCo), by the Italian Ministry of Research within PRIN project "GenData 2020" (2010RTFWBH), and by Google, under the Google Research Award program.

References

1. Aggarwal, G., et al.: Two can keep a secret: A distributed architecture for secure database services. In: Proc. of CIDR 2005, Asilomar, CA, USA (January 2005)
2. Biskup, J.: Dynamic policy adaptation for inference control of queries to a propositional information system. JCS 20(5), 509–546 (2012)
3. Chang, C., Thompson, B., Wang, H., Yao, D.: Towards publishing recommendation data with predictive anonymization. In: Proc. of ASIACCS 2010, Beijing, China (April 2010)

4. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Combining fragmentation and encryption to protect privacy in data storage. ACM TISSEC 13(3), 22:1–22:33 (2010)
5. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Selective data outsourcing for enforcing privacy. JCS 19(3), 531–566 (2011)
6. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Samarati, P.: k-Anonymous data mining: A survey. In: Aggarwal, C., Yu, P. (eds.) Privacy-Preserving Data Mining: Models and Algorithms. Springer (2008)
7. Cormode, G., Srivastava, D., Yu, T., Zhang, Q.: Anonymizing bipartite graph data using safe groupings. PVLDB 1(1), 833–844 (2008)
8. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Livraga, G.: Enforcing subscription-based authorization policies in cloud scenarios. In: Cuppens-Boulahia, N., Cuppens, F., Garcia-Alfaro, J. (eds.) DBSec 2012. LNCS, vol. 7371, pp. 314–329. Springer, Heidelberg (2012)
9. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Fragments and loose associations: Respecting privacy in data publishing. PVLDB 3(1), 1370–1381 (2010)
10. De Capitani di Vimercati, S., Foresti, S., Livraga, G., Samarati, P.: Protecting privacy in data release. In: Aldini, A., Gorrieri, R. (eds.) FOSAD 2011. LNCS, vol. 6858, pp. 1–34. Springer, Heidelberg (2011)
11. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006)
12. Jhawar, R., Piuri, V., Samarati, P.: Supporting security requirements for resource management in cloud computing. In: Proc. of CSE 2012, Paphos, Cyprus (December 2012)
13. Kifer, D., Gehrke, J.: Injecting utility into anonymized datasets. In: Proc. of SIGMOD 2006, Chicago, IL, USA (June 2006)
14. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and ℓ-diversity. In: Proc. of ICDE 2007, Istanbul, Turkey (April 2007)
15. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: ℓ-Diversity: Privacy beyond k-anonymity. ACM TKDD 1(1), 3:1–3:52 (2007)
16. Raeder, T., Blanton, M., Chawla, N.V., Frikken, K.: Privacy-preserving network aggregation. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010, Part I. LNCS, vol. 6118, pp. 198–207. Springer, Heidelberg (2010)
17. Samarati, P.: Protecting respondents' identities in microdata release. IEEE TKDE 13(6), 1010–1027 (2001)
18. Tao, Y., Pei, J., Li, J., Xiao, X., Yi, K., Xing, Z.: Correlation hiding by independence masking. In: Proc. of ICDE 2010, Long Beach, CA, USA (March 2010)
19. Terrovitis, M., Mamoulis, N., Liagouris, J., Skiadopoulos, S.: Privacy preservation by disassociation. PVLDB 5(10), 944–955 (2012)
20. Wang, K., Fung, B.: Anonymizing sequential releases. In: Proc. of KDD 2006, Philadelphia, PA, USA (August 2006)
21. Xiao, X., Tao, Y.: Anatomy: Simple and effective privacy preservation. In: Proc. of VLDB 2006, Seoul, Korea (September 2006)

Database Fragmentation with Encryption: Under Which Semantic Constraints and A Priori Knowledge Can Two Keep a Secret?*

Joachim Biskup and Marcel Preuß

Technische Universität Dortmund, Dortmund, Germany
{biskup,preuss}@ls6.cs.tu-dortmund.de

Abstract. Database outsourcing to semi-honest servers raises concerns against the confidentiality of sensitive information. To hide such information, an existing approach splits data among two supposedly mutually isolated servers by means of fragmentation and encryption. This approach is modelled logic-orientedly and then proved to be confidentiality preserving, even if an attacker employs some restricted but nevertheless versatile class of a priori knowledge to draw inferences. Finally, a method to compute a secure fragmentation schema is developed.

Keywords: A Priori Knowledge, Confidentiality Constraint, Fragmentation, Inference-Proofness, Logic, Outsourcing, Semi-Honest Server.

1 Introduction

Database outsourcing faces two directly conflicting goals: it should both reduce storage and processing costs by storing data on external servers as well as provably comply with confidentiality requirements – in particular with privacy concerns – in spite of storing data externally [10]. A basic solution presented in [2,9] aims at resolving this conflict by means of the combined usage of fragmentation and encryption: a client's database relation is losslessly decomposed into (at least) two vertical fragments each of which is maintained by a different semi-honest server; sensitive data is split into harmless parts, either by breaking an association or by separating an encrypted piece of data from the cryptographic key employed; moreover, the servers are (postulated to be) mutually isolated and each attacker is assumed to have access to at most one server.

Consequently, due to splitting, each attacker (identified with a server) only has accesses to non-sensitive data and, due to losslessness, an authorized user (identified with the client) can still reconstruct the original data while, due to isolation, only authorized users can do so.

Example 1. We consider the relational instance about medical data shown in the upper half of Fig. 1. Suppose that social security numbers (`SSN`) should be hidden, as well as associations between a patient identified by his name (`Name`)

* This work has been supported by the DFG under grant SFB 876/A5.

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 17–32, 2013.

R	SSN	Name	Illness	HurtBy	Doctor
	1234	Hellmann	Borderline	Hellmann	White
	2345	Dooley	Laceration	McKinley	Warren
	3456	McKinley	Laceration	Dooley	Warren
	3456	McKinley	Concussion	Dooley	Warren

F_1	tid	SSN	Name	HurtBy	Doctor
	1	e_S^1	Hellmann	e_H^1	White
	2	e_S^2	Dooley	e_H^2	Warren
	3	e_S^3	McKinley	e_H^3	Warren
	4	e_S^4	McKinley	e_H^4	Warren

F_2	tid	SSN	HurtBy	Illness
	1	κ_S^1	κ_H^1	Borderline
	2	κ_S^2	κ_H^2	Laceration
	3	κ_S^3	κ_H^3	Laceration
	4	κ_S^4	κ_H^4	Concussion

Fig. 1. A relational instance containing sensitive data items and associations together with a possible fragmentation with encryption

and an illness treated (`Illness`), between a patient (`Name`) and a person who caused an illness (`HurtBy`), and between an illness (`Illness`) and a person having caused that illness (`HurtBy`), respectively. The lower half of Fig. 1 exhibits a possible fragmentation with encryption: The sensitive association between `Name` and `Illness` is "broken" by separating the attribute `Name` in the fragment F_1 from the attribute `Illness` in the fragment F_2. The sensitive associations between `Name` and `HurtBy` and between `Illness` and `HurtBy` are made "invisible" by using encryption for the attribute `HurtBy` such that ciphertexts are stored in fragment F_1 and corresponding keys in fragment F_2. The sensitive attribute `SSN` is similarly treated by encryption. The newly introduced tuple identifiers (`tid`) ensure the losslessness of the vertical decomposition (see, e.g., [1]).

At first glance two semi-honest servers seem to "keep the secrets" declared in a confidentiality policy. However, a second thought raises some doubts on the actual achievements: though each server only stores data that is non-sensitive per se, an attacker might still be able to infer sensitive information by exploiting his a priori knowledge obtained from further sources. In particular, this a priori knowledge might comprise semantic constraints to be satisfied by the relation being decomposed and individual fact data stemming from the "outside world".

Example 2. Suppose an attacker has access to the fragment F_1 and knows a priori that Doctor `White` is a psychiatrist only treating patients suffering from the `Borderline`-syndrome. The attacker can then conclude that patient `Hellmann` suffers from the illness `Borderline`-syndrome, thereby violating the requirement that associations between a patient and an illness treated should be hidden. Moreover, if this attacker additionally knows that all patients suffering from the `Borderline`-syndrome have hurt themselves, the attacker can conclude that patient `Hellmann` has been hurt by `Hellmann`, thereby revealing an association between a patient and a person who caused an illness.

The first violation is enabled by a priori knowledge connecting a fact shown in the visible fragment with a fact in the hidden fragment, namely by means of the *constant symbols* `White` and `Borderline`. Similarly, the second violation is caused by a priori knowledge that connects two concepts across the decomposition, namely the concept of a patient and the concept of a hurt creator, where

a concept will be formally represented by a *variable* ranging over the domain of an attribute. Such connections might "transfer information" between the visible fragment and the hidden fragment. In other words, an attacker a priori knowing such connections might infer hidden information from visible information. Next, we introduce a more abstract example in a more formal way.

Example 3. The client maintains a relational schema with relational symbol R, attribute set $A_R = \{a_1, a_2, a_3, a_4\}$ and the functional dependency $a_2 \rightarrow a_3$ as a semantic constraint. Confidentiality interests are expressed by a set $\mathcal{C} = \{\{a_1, a_3\}, \{a_4\}\}$ of two confidentiality constraints: $\{a_1, a_3\}$ is intended to require to hide the associations between values of the attributes a_1 and a_3, and $\{a_4\}$ requires to hide single values of attribute a_4. The a priori knowledge comprises the functional dependency and a sentence expressing the following: "for some specific values b and c for the attributes a_2 and a_3, resp., there exist a value X_1 for attribute a_1 and a value X_4 for attribute a_4 such that the tuple $(a_1 : X_1\,,\, a_2 : b\,,\, a_3 : c\,,\, a_4 : X_4)$ is an element of the relational instance r". Furthermore, fragment F_1 has attribute set $A_{F_1} = \{tid, a_1, a_2, a_4\}$ and fragment F_2 attribute set $A_{F_2} = \{tid, a_3, a_4\}$ such that the common attribute a_4 is encrypted.

Let fragment F_1 exhibit a tuple $(tid : no\,,\, a_1 : a\,,\, a_2 : b\,,\, a_4 : ran)$, where no is a tuple identifier and ran results from encryption. Combining the a priori knowledge with the tuple exhibited, an attacker might infer that the value a for attribute a_1 is associated with the value c for attribute a_3, thereby violating the confidentiality constraint $\{a_1, a_3\}$. Thus fragment F_1 is not inference-proof under the given assumptions. In contrast, fragment F_2 is harmless.

The a priori knowledge relates the fragments F_1 and F_2 by means of both the functional dependency using variables and the association fact about a and b dealing with constant symbols. Though taken alone, each of these items might be harmless, their combination turns out to be potentially harmful. The next example indicates that for the same underlying situation one fragmentation satisfying required confidentiality constraints might be better than another one.

Example 4. Modifying Example 3 such that $A_{F_1} = \{tid, a_1, a_4\}$ and $A_{F_2} = \{tid, a_2, a_3, a_4\}$ would block the harmful inference. For, intuitively, the crucial fact about the association of a with b does not span across the decomposition.

More generally, we will investigate the following problems in this article:

- Given a fragmentation, identify conditions on the a priori knowledge to provably disable an attacker to infer sensitive information.
- Given some a priori knowledge, determine a fragmentation such that an attacker cannot infer sensitive information.

Our solutions will be based on a logic-oriented modelling of the fragmentation approach presented in [2,9] within the more general framework of Controlled Interaction Execution, CIE, as surveyed in [3]. This framework assists a database owner in ensuring that each of his interaction partners can only obtain a dedicated inference-proof view on the owner's data: each of these views does not

contain information to be kept confidential from the respective partner, even if this partner tries to employ inferences by using his a priori knowledge and his general awareness of the protection mechanism. Our main achievements can be summarized as follows and will be elaborated in the remainder as indicated:

- We formalize the fragmentation approach of [2,9] (Sect. 2).
- We provide a logic-oriented modelling of that approach (Sect. 3).
- We exhibit sufficient conditions to achieve confidentiality (Sect. 4).
- We propose a method to compute a suitable fragmentation (Sect. 5).

These results extend the previous work [5] in which a more simple approach to fragmentation proposed in [7] – splitting a relational instance into one externally stored part and one locally-held part without resorting to encryption – is formally analyzed to be inference-proof. In particular, the previous work is extended by a more detailed formal modelling of fragmentation including encryption of values, a more expressive class of sentences representing an attacker's a priori knowledge and a method to compute an inference-proof fragmentation.

2 Confidentiality by Fragmentation

In this section, we briefly formalize and extend the approach to fragmentation proposed in [2,9]. All data is represented within a single relational instance r over a relational schema $\langle R|A_R|SC_R\rangle$ with relational symbol R and the set $A_R = \{a_1, \ldots, a_n\}$ of attributes, for simplicity assumed to have the same type given by the infinite set $\mathcal{U}$ of values. Moreover, the set SC_R contains some semantic (database) constraints, which must be satisfied by the relational instance r.

The idea for achieving confidentiality basically lies in splitting the original instance r vertically (i.e., by projections on subsets of A_R) into two fragment instances f_1 and f_2 each of which is stored on *exactly one* of the two external servers instead of r. Those confidentiality requirements which cannot be satisfied by just splitting instance r are satisfied by encrypting the values of some attributes. Each "encrypted attribute" is contained in f_1 – storing ciphertexts – as well as in f_2 – storing globally unique cryptographic keys.

We assume an encryption function $Enc : \mathcal{U} \times \mathcal{U} \to \mathcal{U}$ satisfying the group properties to achieve perfect (information-theoretic) security. A value of $\mathcal{U}$ might be used not only as a plaintext but also as a cryptographic key and a ciphertext. The decryption function is defined by $Dec(e, \kappa) = v$ iff $Enc(v, \kappa) = e$.

Definition 1 (Fragmentation). *Given a relational schema $\langle R|A_R|SC_R\rangle$, a* vertical fragmentation $(\mathcal{F}, \mathcal{E})$ *of $\langle R|A_R|SC_R\rangle$ contains a set $\mathcal{E} \subseteq A_R$ of so-called "encrypted attributes" and a set $\mathcal{F} = \{\langle F_1|A_{F_1}|SC_{F_1}\rangle, \langle F_2|A_{F_2}|SC_{F_2}\rangle\}$ in which $\langle F_1|A_{F_1}|SC_{F_1}\rangle$ and $\langle F_2|A_{F_2}|SC_{F_2}\rangle$ are relational schemas called* fragments *of $(\mathcal{F}, \mathcal{E})$ both containing the distinguished attribute $a_{\mathtt{tid}} \notin A_R$ for tuple identifiers. Moreover, for $i \in \{1, 2\}$, it holds that*

	$A_{F_i} \setminus A_R$	$(A_{F_1} \setminus \mathcal{E}) \cap A_R$	$\mathcal{E} \cap A_{F_i} \cap A_R$	$(A_{F_2} \setminus \mathcal{E}) \cap A_R$
A_R		$a_1, \ldots, a_h$	$a_{h+1}, \ldots, a_k$	$a_{k+1}, \ldots, a_n$
A_{F_1}	$a_{\mathtt{tid}}$	$a_1, \ldots, a_h$	$a_{h+1}, \ldots, a_k$	
A_{F_2}	$a_{\mathtt{tid}}$		$a_{h+1}, \ldots, a_k$	$a_{k+1}, \ldots, a_n$

Fig. 2. Rearrangement of columns of r, f_1 and f_2

(i) $A_{F_i} := \{a_{tid}\} \cup \bar{A}_{F_i}$ *with* $\bar{A}_{F_i} \subseteq A_R$,
(ii) $SC_{F_i} := \{a_{tid} \rightarrow \bar{A}_{F_i}\}$ *with* $a_{tid} \rightarrow \bar{A}_{F_i}$ *being a functional dependency declaring* a_{tid} *as a primary key,*
(iii) $\bar{A}_{F_1} \cup \bar{A}_{F_2} = A_R$ *and* $\bar{A}_{F_1} \cap \bar{A}_{F_2} = \mathcal{E}$.

Given a relational instance r over $\langle R|A_R|SC_R\rangle$, the fragment instances *f_1 and f_2 over $\langle F_1|A_{F_1}|SC_{F_1}\rangle$ and $\langle F_2|A_{F_2}|SC_{F_2}\rangle$ are created by inserting exactly both the tuples ν_1 into f_1 and ν_2 into f_2 for each tuple $\mu \in r$. Thereby,*

(a) $\nu_1[a_{tid}] = \nu_2[a_{tid}] = v_\mu$ *s.t.* v_μ *is a globally unique tuple identifier,*
(b) $\nu_i[a] = \mu[a]$ *for* $i \in \{1, 2\}$ *and for each attribute* $a \in (\bar{A}_{F_i} \setminus \mathcal{E})$,
(c) $\nu_1[a] := Enc(\mu[a], \kappa)$ *and* $\nu_2[a] := \kappa$ *for each* $a \in \mathcal{E}$ *s.t.* κ *is a cryptographic key being random but globally unique for each value of each tuple.*

W.l.o.g. we suppose that $A_R := \{a_1, \ldots, a_h, a_{h+1}, \ldots, a_k, a_{k+1}, \ldots, a_n\}$ is the set of attributes of $\langle R|A_R|SC_R\rangle$ and that the columns of the instances r, f_1 and f_2 are rearranged as visualized in Fig. 2. The columns $h+1, \ldots, k$ differ in the interpretation of the values stored in the instances r, f_1 and f_2: although each of the tuples $\mu \in r$, $\nu_1 \in f_1$ and $\nu_2 \in f_2$ assign values to the attributes $a_{h+1}, \ldots, a_k$, $\mu[a_j]$ is a plaintext value, $\nu_1[a_j]$ is a ciphertext value and $\nu_2[a_j]$ is a cryptographic key. In contrast, for $a_1, \ldots, a_h$ ($a_{k+1}, \ldots, a_n$, respectively) corresponding tuples of r and f_1 (r and f_2) share the same combination of values.

To enable an authorized user having access to both fragment-instances f_1 and f_2 to query all information contained in the original instance r, fragmentation ensures that in f_1 and f_2 exactly those two tuples $\nu_1 \in f_1$ and $\nu_2 \in f_2$ corresponding to a tuple of r share the same unique tuple ID (item (a) of Def. 1). Thus, if $\nu_1[a_{\mathtt{tid}}] = \nu_2[a_{\mathtt{tid}}]$, two tuples $\nu_1 \in f_1$ and $\nu_2 \in f_2$ can be recomposed to a tuple of r with the help of a binary operation denoted by $\diamond$.

As the goal is to achieve confidentiality by fragmentation, a formal declaration of confidentiality requirements is indispensable. In [2,9] this is obtained by defining a set of so-called confidentiality constraints on schema level.

Definition 2 (Confidentiality Constraint). *A* confidentiality constraint *c over a relational schema $\langle R|A_R|SC_R\rangle$ is a non-empty subset $c \subseteq A_R$.*

Semantically, a confidentiality constraint c claims that each combination of values allocated to the set $c \subseteq A_R$ of attributes in the original instance r over schema $\langle R|A_R|SC_R\rangle$ should neither be contained completely in the unencrypted part of f_1 nor be contained completely in the unencrypted part of f_2.

Definition 3 (Confidentiality of Fragmentation). *Let $\langle R|A_R|SC_R\rangle$ be a relational schema, $(\mathcal{F}, \mathcal{E})$ a fragmentation of $\langle R|A_R|SC_R\rangle$ according to Def. 1*

and $\mathcal{C}$ a set of confidentiality constraints over $\langle R|A_R|SC_R\rangle$ according to Def. 2. $(\mathcal{F},\mathcal{E})$ is confidential *w.r.t. $\mathcal{C}$ iff $c \not\subseteq (A_{F_1} \setminus \mathcal{E})$ and $c \not\subseteq (A_{F_2} \setminus \mathcal{E})$ for each $c \in \mathcal{C}$.*

Example 5. The fragmentation depicted in Fig. 1 is confidential w.r.t. the set $\mathcal{C} = \{c_1, c_2, c_3, c_4\}$ of confidentiality constraints such that $c_1 = \{\texttt{SSN}\}$, $c_2 = \{\texttt{Name}, \texttt{Illness}\}$, $c_3 = \{\texttt{Name}, \texttt{HurtBy}\}$, and $c_4 = \{\texttt{Illness}, \texttt{HurtBy}\}$.

3 A Logic-Oriented View on Fragmentation

In this section we will present a logic-oriented modelling of fragmentation, for conciseness mostly focussing on the attacker's point of view resulting from his knowledge of the fragment instance f_1, which is supposed to be known to him.

To set up the universe of discourse, we start by defining the set $\mathcal{P}$ of predicate symbols of a language $\mathscr{L}$ of first-order logic with equality. First, to model the attacker's knowledge about the fragment instance f_1, we need the predicate symbol $F_1 \in \mathcal{P}$ with arity $k+1 = |A_{F_1}|$ (including the additional tuple ID attribute plus k original attributes (cf. Fig. 2)). Second, to capture the attacker's awareness of the fragmentation, in particular his partial knowledge about the hidden original instance r and the separated second fragmentation instance f_2, we additionally use the predicate symbols R with arity $n = |A_R|$ and F_2 with arity $n - h + 1 = |A_{F_2}|$. Additionally, the distinguished predicate symbol $\equiv \notin \mathcal{P}$ is available in $\mathscr{L}$ for expressing equality.

We employ the binary function symbols E and D for modelling the attacker's knowledge about the encryption function *Enc* and the inverse decryption function *Dec*. Finally, we denote tuple values by elements of the set *Dom* of constant symbols, which will be employed as the universe of (Herbrand) interpretations for $\mathscr{L}$ as well. In compliance with CIE (e.g., [4,6]) this set is assumed to be fixed and infinite. Further, we have an infinite set *Var* of variables.

As usual, the formulas contained in $\mathscr{L}$ are constructed inductively using the quantifiers $\forall$ and $\exists$ and the connectives $\neg$, $\wedge$, $\vee$ and $\Rightarrow$. Closed formulas, i.e., formulas without free occurrences of variables, are called sentences. This syntactic specification is complemented with a semantics which reflects the characteristics of databases by means of so-called DB-Interpretations according to [4,6]:

Definition 4 (DB-Interpretation). *Given the language $\mathscr{L}$ described above, an interpretation $\mathcal{I}$ over a universe $\mathcal{U}$ is a* DB-Interpretation *for $\mathscr{L}$ iff*

(i) Universe $\mathcal{U} := \mathcal{I}(Dom) = Dom$,
(ii) $\mathcal{I}(v) = v \in \mathcal{U}$ for every constant symbol $v \in Dom$,
(iii) $\mathcal{I}(E)(v,\kappa) = e$ iff $Enc(v,\kappa) = e$, for all $v,\kappa,e \in \mathcal{U}$,
(iv) $\mathcal{I}(D)(e,\kappa) = v$ iff $Dec(e,\kappa) = v$, for all $v,\kappa,e \in \mathcal{U}$,
(v) every $P \in \mathcal{P}$ with arity m is interpreted by a finite relation $\mathcal{I}(P) \subset \mathcal{U}^m$,
(vi) the predicate symbol $\equiv \notin \mathcal{P}$ is interpreted by $\mathcal{I}(\equiv) = \{(v,v) \mid v \in \mathcal{U}\}$.

If item (v) is instantiated by taking the instances r, f_1 and f_2 as interpretations of P=R, F_1, and F_2, respectively, the resulting DB-Interpretation $\mathcal{I}_{r,f_1,f_2}$ – or just $\mathcal{I}_r$ for short if f_1 and f_2 are derived from r according to Def. 1 – is called induced *by r (and f_1 and f_2).*

The notion of *satisfaction/validity* of formulas in $\mathscr{L}$ by a DB-Interpretation is the same as in usual first-order logic. A set $\mathcal{S} \subset \mathscr{L}$ of sentences *implies/entails* a sentence $\Phi \in \mathscr{L}$ (written as $\mathcal{S} \models_{DB} \Phi$) iff each DB-Interpretation $\mathcal{I}$ satisfying $\mathcal{S}$ (written as $\mathcal{I} \models_M \mathcal{S}$) also satisfies Φ (written as $\mathcal{I} \models_M \Phi$).

Considering an attacker knowing the fragment instance f_1, the attacker's *positive* knowledge about the tuples explicitly recorded in f_1 can be simply modelled logic-orientedly by adding an atomic sentence $F_1(\nu[a_{\texttt{tid}}], \nu[a_1], \ldots, \nu[a_k])$ for each tuple $\nu \in f_1$. As the original instance r – and so its fragment instance f_1 – is assumed to be *complete*[1], each piece of information expressible in $\mathscr{L}$ which is *not* contained in r (f_1, resp.) is considered to be *not* valid by Closed World Assumption (CWA). The concept of DB-Interpretations fully complies with the semantics of *complete* relational instances. Accordingly, an attacker knows that each of the infinite combinations of values $(v_{\texttt{tid}}, v_1, \ldots, v_k) \in \mathit{Dom}^{k+1}$ not contained in any tuple of f_1 leads to a valid sentence $\neg F_1(v_{\texttt{tid}}, v_1, \ldots, v_k)$.

As this *negative* knowledge is not explicitly enumerable, it is expressed implicitly by a so-called completeness sentence (cf. [4]) having a universally quantified variable X_j for each attribute $a_j \in A_{F_1}$ (sentence (2) of Def. 5 below). This completeness sentence expresses that every constant combination $(v_{\texttt{tid}}, v_1, \ldots, v_k) \in \mathit{Dom}^{k+1}$ (substituting the universally quantified variables $X_{\texttt{tid}}, X_1, \ldots X_k$) either appears in f_1 or satisfies the sentence $\neg F_1(v_{\texttt{tid}}, v_1, \ldots, v_k)$. By construction, this completeness sentence is satisfied by any DB-Interpretation induced by f_1.

Example 6. For the medical example, the knowledge implicitly taken to be *not* valid by CWA can be expressed as the following completeness sentence:

$$
\begin{array}{l}
(\forall X_t)(\forall X_S)(\forall X_N)(\forall X_H)(\forall X_D)\ [\\
(X_t \equiv 1 \ \wedge\ X_S \equiv e_S^1 \ \wedge\ X_N \equiv \text{Hellmann} \ \wedge\ X_H \equiv e_H^1 \ \wedge\ X_D \equiv \text{White}) \ \vee\\
(X_t \equiv 2 \ \wedge\ X_S \equiv e_S^2 \ \wedge\ X_N \equiv \text{Dooley} \ \wedge\ X_H \equiv e_H^2 \ \wedge\ X_D \equiv \text{Warren}) \ \vee\\
(X_t \equiv 3 \ \wedge\ X_S \equiv e_S^3 \ \wedge\ X_N \equiv \text{McKinley} \ \wedge\ X_H \equiv e_H^3 \ \wedge\ X_D \equiv \text{Warren}) \ \vee\\
(X_t \equiv 4 \ \wedge\ X_S \equiv e_S^4 \ \wedge\ X_N \equiv \text{McKinley} \ \wedge\ X_H \equiv e_H^4 \ \wedge\ X_D \equiv \text{Warren}) \ \vee\\
\neg F_1(X_t, X_S, X_N, X_H, X_D) \qquad]
\end{array}
$$

Based on the explanations given so far, an attacker's knowledge about the fragment instance f_1 can be formalized logic-orientedly as follows:

Definition 5 (Logic-Oriented View on f_1). *Given a fragment instance f_1 over $\langle F_1 | A_{F_1} | SC_{F_1} \rangle$ according to Def. 1 with $A_{F_1} = \{a_{tid}, a_1, \ldots, a_k\}$, the positive knowledge contained in f_1 is modelled in $\mathscr{L}$ by the set of sentences*

$$db^+_{f_1} := \{F_1(\nu[a_{tid}], \nu[a_1], \ldots, \nu[a_k]) \mid \nu \in f_1\} \ . \tag{1}$$

The implicit negative knowledge contained in f_1 is modelled in $\mathscr{L}$ by the singleton set $db^-_{f_1}$ containing the completeness sentence

$$(\forall X_{tid}) \ldots (\forall X_k) \left[\bigvee_{\nu \in f_1} \left(\bigwedge_{a_j \in A_{F_1}} (X_j \equiv \nu[a_j]) \right) \vee \neg F_1(X_{tid}, X_1, \ldots, X_k) \right] . \tag{2}$$

[1] Though not explicitly stated in [2,9], in this article we follow the usual intuitive semantics of complete instances.

Moreover the functional dependency $a_{\mathtt{tid}} \to \{a_1, \ldots, a_k\} \in SC_{F_1}$ *is modelled in* $\mathscr{L}$ *by the singleton set* fd_{F_1} *containing the sentence*

$$(\forall X_{\mathtt{tid}})\,(\forall X_1)\ldots(\forall X_k)\,(\forall X'_1)\ldots(\forall X'_k)\,[\,F_1(X_{\mathtt{tid}}, X_1, \ldots, X_k) \wedge F_1(X_{\mathtt{tid}}, X'_1, \ldots, X'_k) \Rightarrow (X_1 \equiv X'_1) \wedge \ldots \wedge (X_k \equiv X'_k)\,]\ . \tag{3}$$

Overall the logic-oriented view on f_1 *in* $\mathscr{L}$ *is* $db_{f_1} := db^+_{f_1} \cup db^-_{f_1} \cup fd_{F_1}$.

Proposition 1. *Under the assumptions of Def. 5, the sentences (1), (2) and (3) of* db_{f_1} *are satisfied by the DB-interpretation* $\mathcal{I}_r$*, i.e.,* $\mathcal{I}_r \models_M db_{f_1}$.

Proof. Direct consequence of the definitions. □

An attacker is assumed to know the process of fragmentation as well as the schemas $\langle R|A_R|SC_R\rangle$ and $\langle F_2|A_{F_2}|SC_{F_2}\rangle$ of the instances kept hidden from him. Thus he can infer that for each tuple $\nu_1 \in f_1$ there are tuples $\nu_2 \in f_2$ and $\mu \in r$ satisfying the equation $\nu_1 \diamond \nu_2 = \mu$. So, an attacker knows all values assigned to the set $(A_{F_1} \setminus \mathcal{E}) \cap A_R$ of unencrypted attributes in μ from his knowledge of ν_1, whereas in general he only knows the existence of values for the remaining attributes (sentence (4) of Def. 6 below). Similarly, the attacker is not able to infer the cleartext values assigned to the attributes of $\mathcal{E}$ in μ: by the group properties of the encryption function, each ciphertext considered might be mapped to each possible cleartext without knowing the specific key hidden in fragment f_2.

Next, an attacker knows that a tuple $\nu_2 \in f_2$ can only exist if also corresponding tuples $\nu_1 \in f_1$ and $\mu \in r$ satisfying the equation $\nu_1 \diamond \nu_2 = \mu$ exist (sentence (5) of Def. 6 below). According to the if-part of sentence (6) this requirement analogously holds for the existence of each tuple of r. The only-if-part of sentence (6) describes the fact that the (hypothetical) knowledge of both tuples $\nu_1 \in f_1$ and $\nu_2 \in f_2$ with $\nu_1[a_{\mathtt{tid}}] = \nu_2[a_{\mathtt{tid}}]$ would enable the attacker to reconstruct the tuple $\mu \in r$ satisfying $\mu = \nu_1 \diamond \nu_2$ completely.

Based on the one-to-one correspondence between each tuple $\mu \in r$ and a tuple $\nu_1 \in f_1$ ($\nu_2 \in f_2$, resp.), observing that two different tuples $\nu_1, \nu'_1 \in f_1$ are equal w.r.t. the values allocated to the unencrypted attributes of $(A_{F_1} \setminus \mathcal{E}) \cap A_R$, an attacker can reason that there are also two tuples $\mu, \mu' \in r$ which are equal w.r.t. the values allocated to these attributes, but differ in at least one of the values allocated to $A_R \setminus (A_{F_1} \setminus \mathcal{E})$ (sentence (7) (sentence (8) in case of f_2) of Def. 6 below). Otherwise, the instance r would have duplicates.

Summarizing, and for now neglecting semantic constraints, an attacker's logic-oriented view on the (hidden) instances r and f_2 can be modelled as follows:

Definition 6 (Fragmentation Logic-Oriented). *Let* $(\mathcal{F}, \mathcal{E})$ *be a fragmentation of a relational schema* $\langle R|A_R|SC_R\rangle$ *with instance* r *and let* f_1 *and* f_2 *be the corresponding fragment instances over the fragments* $\langle F_1|A_{F_1}|SC_{F_1}\rangle \in \mathcal{F}$ *and* $\langle F_2|A_{F_2}|SC_{F_2}\rangle \in \mathcal{F}$ *according to Def. 1.*

The knowledge about r and f_2 deduced from the knowledge of f_1 is expressed by

$$\begin{array}{l}(\forall X_{\mathit{tid}})\,(\forall X_1)\ldots(\forall X_h)\,(\forall X_{h+1})\ldots(\forall X_k)\,\big[\\ \quad F_1\,(X_{\mathit{tid}}, X_1, \ldots, X_h, X_{h+1}, \ldots, X_k)\\ \quad\quad \Rightarrow\\ \quad (\exists Y_{h+1})\ldots(\exists Y_k)\,(\exists Z_{k+1})\ldots(\exists Z_n)\,\big[\\ \quad\quad F_2\,(X_{\mathit{tid}}, Y_{h+1}, \ldots, Y_k, Z_{k+1}, \ldots, Z_n) \wedge\\ \quad\quad R\,(X_1, \ldots, X_h, D\,(X_{h+1}, Y_{h+1}), \ldots, D\,(X_k, Y_k), Z_{k+1}, \ldots, Z_n)\,\big]\big]\ ;\end{array} \tag{4}$$

the knowledge about r and f_1 deduced from the knowledge of f_2 is expressed by

$$\begin{array}{l}(\forall X_{\mathit{tid}})\,(\forall X_{h+1})\ldots(\forall X_k)\,(\forall X_{k+1})\ldots(\forall X_n)\,\big[\\ \quad F_2\,(X_{\mathit{tid}}, X_{h+1}, \ldots, X_k, X_{k+1}, \ldots, X_n)\\ \quad\quad \Rightarrow\\ \quad (\exists Y_1)\ldots(\exists Y_h)\,(\exists Z_{h+1})\ldots(\exists Z_k)\,\big[\\ \quad\quad F_1\,(X_{\mathit{tid}}, Y_1, \ldots, Y_h, Z_{h+1}, \ldots, Z_k) \wedge\\ \quad\quad R\,(Y_1, \ldots, Y_h, D\,(Z_{h+1}, X_{h+1}), \ldots, D\,(Z_k, X_k), X_{k+1}, \ldots, X_n)\,\big]\big]\ ;\end{array} \tag{5}$$

the knowledge about f_1 and f_2 deduced from the knowledge of r as well as the knowledge about r deduced from f_1 and f_2 is expressed by

$$\begin{array}{l}(\forall X_1)\ldots(\forall X_h)\,(\forall X_{h+1})\ldots(\forall X_k)\,(\forall X_{k+1})\ldots(\forall X_n)\,\big[\\ \quad R\,(X_1, \ldots, X_h, X_{h+1}, \ldots, X_k, X_{k+1}, \ldots, X_n)\\ \quad\quad \Leftrightarrow\\ \quad (\exists Z_{\mathit{tid}})\,(\exists Y_{h+1})\ldots(\exists Y_k)\,\big[\\ \quad\quad F_2\,(Z_{\mathit{tid}}, Y_{h+1}, \ldots, Y_k, X_{k+1}, \ldots, X_n) \wedge\\ \quad\quad F_1\,(Z_{\mathit{tid}}, X_1, \ldots, X_h, E\,(X_{h+1}, Y_{h+1}), \ldots, E\,(X_k, Y_k))\,\big]\big]\ ;\end{array} \tag{6}$$

the knowledge about inequalities in r based on f_1 is expressed by

$$\begin{array}{l}(\forall X_{\mathit{tid}})\,(\forall X'_{\mathit{tid}})\,(\forall X_1)\ldots(\forall X_h)\,(\forall X_{h+1})\ldots(\forall X_k)\,\left(\forall X'_{h+1}\right)\ldots(\forall X'_k)\,\big[\\ \quad \big[F_1\,(X_{\mathit{tid}}, X_1, \ldots, X_h, X_{h+1}, \ldots, X_k) \wedge\\ \quad\quad F_1\left(X'_{\mathit{tid}}, X_1, \ldots, X_h, X'_{h+1}, \ldots, X'_k\right) \wedge (X_{\mathit{tid}} \not\equiv X'_{\mathit{tid}})\big]\\ \quad\quad \Rightarrow\\ \quad (\exists Y_{h+1})\ldots(\exists Y_n)\,(\exists Z_{h+1})\ldots(\exists Z_n)\,\big[\\ \quad\quad R\,(X_1, \ldots, X_h, Y_{h+1}, \ldots, Y_k, Y_{k+1}, \ldots, Y_n) \wedge\\ \quad\quad R\,(X_1, \ldots, X_h, Z_{h+1}, \ldots, Z_k, Z_{k+1}, \ldots, Z_n) \wedge \bigvee_{j=h+1}^{n} (Y_j \not\equiv Z_j)\,\big]\big]\ ;\end{array} \tag{7}$$

and the knowledge about inequalities in r based on f_2 is expressed by

$$\begin{array}{l}(\forall X_{\mathit{tid}})\,(\forall X'_{\mathit{tid}})\,(\forall X_{h+1})\ldots(\forall X_k)\,\left(\forall X'_{h+1}\right)\ldots(\forall X'_k)\,(\forall X_{k+1})\ldots(\forall X_n)\,\big[\\ \quad \big[F_2\,(X_{\mathit{tid}}, X_{h+1}, \ldots, X_k, X_{k+1}, \ldots, X_n) \wedge\\ \quad\quad F_2\left(X'_{\mathit{tid}}, X'_{h+1}, \ldots, X'_k, X_{k+1}, \ldots, X_n\right) \wedge (X_{\mathit{tid}} \not\equiv X'_{\mathit{tid}})\big]\\ \quad\quad \Rightarrow\\ \quad (\exists Y_1)\ldots(\exists Y_k)\,(\exists Z_1)\ldots(\exists Z_k)\,\big[\\ \quad\quad R\,(Y_1, \ldots, Y_h, Y_{h+1}, \ldots, Y_k, X_{k+1}, \ldots, X_n) \wedge\\ \quad\quad R\,(Z_1, \ldots, Z_h, Z_{h+1}, \ldots, Z_k, X_{k+1}, \ldots, X_n) \wedge \bigvee_{j=1}^{k} (Y_j \not\equiv Z_j)\,\big]\big]\ .\end{array} \tag{8}$$

This view on r and f_2 is referred to as the set of sentences db_R containing the sentences (4), (5), (6), (7) and (8).

Strictly speaking, db_R alone does not provide *any* knowledge about the relational instance r; instead, only the combination of db_{f_1} and db_R describes the knowledge about r that is available to an attacker. The essential part of this insight is formally captured by the following proposition.

Proposition 2. *Under the assumptions of Def. 6, the sentences (4), (5), (6), (7) and (8) of db_R are satisfied by the DB-Interpretation $\mathcal{I}_r$, i.e., $\mathcal{I}_r \models_M db_R$.*

Proof. Omitted. See the informal explanations before Definition 6. □

Note that – in contrast to sentence (6) – the equivalence does *not* hold for the sentences (4) and (5), as it can be shown by a straightforward example.

Finally, we have to model the confidentiality policy logic-orientedly. A confidentiality constraint $c \subseteq A_R$ claims that each combination of (cleartext-)values allocated to the attributes of c should not be revealed to an attacker completely. To specify this semantics more precisely, it is assumed that c only protects those combinations of values which are explicitly allocated to the attributes of c in a tuple of r. In contrast, an attacker may get to know that a certain combination of values is *not* allocated to the attributes of c in any tuple of r.

The wish to protect a certain combination of values $(v_{i_1}, \ldots, v_{i_\ell}) \in Dom^{|c|}$ is modelled as a "potential secret" in the form of a sentence $(\exists \boldsymbol{X})\, R(t_1, \ldots, t_n)$ in which $t_j := v_j$ holds for each $j \in \{i_1, \ldots, i_\ell\}$ and all other terms are existentially quantified variables. To protect *each* of the infinitely many combinations, regardless of whether it is contained in a tuple of r or not, we use a single open formula with free variables $X_{i_1}, \ldots, X_{i_\ell}$ like an open query as follows.

Definition 7 (Confidentiality Policy). *Let $\mathcal{C}$ be a set of confidentiality constraints over schema $\langle R|A_R|SC_R\rangle$ according to Def. 2. Considering a confidentiality constraint $c_i \in \mathcal{C}$ with $c_i = \{a_{i_1}, \ldots, a_{i_\ell}\} \subseteq \{a_1, \ldots, a_n\} = A_R$ and the set $A_R \setminus c_i = \{a_{i_{\ell+1}}, \ldots, a_{i_n}\}$, constraint c_i is modelled as a* potential secret

$$\Psi_i(\boldsymbol{X_i}) \;:=\; (\exists X_{i_{\ell+1}}) \ldots (\exists X_{i_n})\, R(X_1, \ldots, X_n)\,,$$

which is a formula in the language $\mathscr{L}$. Thereby $\boldsymbol{X_i} = (X_{i_1}, \ldots, X_{i_\ell})$ is the vector of free variables contained in $\Psi_i(\boldsymbol{X_i})$. The set containing exactly one potential secret $\Psi_i(\boldsymbol{X_i})$ constructed as above for every confidentiality constraint $c_i \in \mathcal{C}$ is called potsec($\mathcal{C}$). Moreover, the expansion ex(*potsec*($\mathcal{C}$)) *contains all ground substitutions over Dom of all formulas in potsec($\mathcal{C}$).*

Example 7. For our example, $c_2 = \{\texttt{Name}, \texttt{Illness}\}$ is modelled as $\Psi_2(\boldsymbol{X_2}) := (\exists X_S)(\exists X_H)(\exists X_D) R(X_S, X_N, X_I, X_H, X_D)$ with free variables $\boldsymbol{X_2} = (X_N, X_I)$.

4 Inference-Proofness of Fragmentation

Until now the logic-oriented modelling of an attacker's view only comprises knowledge the attacker can deduce from the outsourced fragment instance f_1, which is supposed to be visible to him. Additionally, however, the attacker might also employ a priori knowledge to draw harmful inferences.

Example 8. As in Example 2, suppose the attacker knows that Doctor `White` is a psychiatrist only treating patients suffering from the `Borderline`-syndrome:

$$(\forall X_S)(\forall X_N)(\forall X_I)(\forall X_H)[R(X_S, X_N, X_I, X_H, \texttt{White}) \Rightarrow (X_I \equiv \texttt{Borderline})] \ .$$

This knowledge enables the attacker to conclude that patient `Hellmann` suffers from the illness `Borderline`-syndrome, thereby violating confidentiality constraint $c_2 = \{\texttt{Name}, \texttt{Illness}\}$. Moreover, let the attacker additionally know that all patients suffering from the `Borderline`-syndrome have hurt themselves:

$$(\forall X_S)(\forall X_N)(\forall X_H)(\forall X_D)[R(X_S, X_N, \texttt{Borderline}, X_H, X_D) \Rightarrow (X_N \equiv X_H)] \ .$$

The attacker can then draw the conclusion that patient `Hellmann` has been hurt by `Hellmann`, thereby violating $c_3 = \{\texttt{Name}, \texttt{HurtBy}\}$.

Following the framework of CIE [3], we aim at achieving a sophisticated kind of confidentiality taking care of an attacker's (postulated) a priori knowledge. This a priori knowledge is modelled as a finite set *prior* of sentences in $\mathscr{L}$ containing only R and $\equiv$ as predicate symbols. Moreover, we always assume that the semantic constraints SC_R declared in the relational schema are publicly known, i.e., $SC_R \subseteq \mathit{prior}$. Intuitively, we then would like to guarantee that a fragmentation is *inference-proof* in the sense that – from the attacker's point of view – each of the potential secrets might *not* be true in the original relational instance r. More formally: for each potential secret $\Psi_i(\boldsymbol{v_i}) \in \mathrm{ex}(\mathit{potsec}(\mathcal{C}))$ there should exist an alternative instance r' over $\langle R|A_R|SC_R\rangle$ that witnesses the non-entailment $db_{f_1} \cup db_R \cup \mathit{prior} \not\models_{DB} \Psi_i(\boldsymbol{v_i})$. Clearly, deciding on non-entailment, equivalently finding a suitable witness, is computationally infeasible in general. Accordingly, we will have to restrict on approximations and special cases.

Regarding approximations, we might straightforwardly require for the witness r' that for at least one $m \in \{i_1, \ldots, i_\ell\}$ the value v_m appearing in the potential secret must *not* occur under the attribute a_m. Accordingly, we could try to substitute v_m in the original instance r by a newly selected constant symbol v^* to obtain r'. However, we also have to preserve indistinguishability of r and r' by the attacker, and thus m has to be chosen such that $a_m \notin (A_{F_1} \setminus \mathcal{E})$. Furthermore, to fully achieve indistinguishability, the alternative instance r' has to coincide with the original instance r on the part visible in fragment f_1, i.e., $\mathcal{I}_{r'} \models_M db_{f_1}$, and modifying the original instance r into the alternative r' should preserve satisfaction of the a priori knowledge, i.e., $\mathcal{I}_{r'} \models_M \mathit{prior}$.

Regarding special cases, we will adapt two useful properties known from relational database theory [1]. *Genericity* of a sentence in $\mathscr{L}$ perceives constant symbols as being atomic and uninterpreted. Intuitively, all knowledge about a constant symbol arises from its occurrences in the relational instance r. Clearly, sentences with "essential" occurrences of constant symbols will not be generic. But in general "essential" occurrences of constant symbols are difficult to identify. Moreover, renaming v_m by v^* should not modify the fragment f_1 that is visible to the attacker. *Typedness* restricts the occurrences of a variable within a sentence to a single attribute (column), and thus prevents a "transfer of information" from a visible attribute to a hidden one.

We will now state our main result about the achievements of fragmentation with encryption regarding preservation of confidentiality against an attacker who only has access to one of the fragment instances, here exemplarily to fragment instance f_1. Facing the challenges discussed above, this main result exhibits a sufficient condition for confidentiality. An inference-proof fragmentation of the running example in terms of Theorem 1 is presented in Example 9 of Sect. 5.

Theorem 1 (Inference-Proofness on Schema Level). *Let $\langle R|A_R|SC_R\rangle$ be a relational schema with $A_R = \{a_1, \ldots, a_n\}$ and $(\mathcal{F}, \mathcal{E})$ be a fragmentation with fragment $\langle F_1|A_{F_1}|SC_{F_1}\rangle \in \mathcal{F}$ that is confidential w.r.t. a set $\mathcal{C}$ of confidentiality constraints. Moreover, let $SC_R \subseteq prior$ be a set of sentences in $\mathscr{L}$ containing only R and $\equiv$ as predicate symbols, satisfying the following restrictions:*

- Untyped dependencies with constants: *each $\Gamma \in prior$ is in the syntactic form of $(\forall \boldsymbol{x})(\exists \boldsymbol{y})[\bigvee_{j=1,\ldots,p} \neg A_j \vee A_{p+1}]$ with A_l being an atom of the form $R(t_{l,1}, \ldots, t_{l,n})$ or $(t_{p+1,1} \equiv t_{p+1,2})$ and $t_{j,i}$ is a variable or a constant symbol; moreover, w.l.o.g., equality predicates may only occur positively, and there might also be a conjunction of positively occurring R-atoms.*
- Satisfiability: *prior is DB-satisfiable and each $\Gamma \in prior$ is not DB-tautologic (and thus: each $\Gamma \in prior$ is range-restricted and does not contain an existentially quantified variable in the negated atoms (premises)).*
- Compatibility with $(\mathcal{F}, \mathcal{E})$ and $\mathcal{C}$: *there is a subset $M \subseteq \{h+1, \ldots, n\}$ s.t.*
  (1) $M \cap \{i_1, \ldots, i_\ell\} \neq \emptyset$ for each $c_i \in \mathcal{C}$ with $c_i = (a_{i_1}, \ldots, a_{i_\ell})$;
  (2) for each $\Gamma \in prior$ there exists a partitioning $\mathcal{X}_1^\Gamma \dot\cup \mathcal{X}_2^\Gamma = Var$ s.t.
    (i) for each atom $R(t_1, \ldots, t_n)$ of Γ
      - *for all $j \in \{1, \ldots, n\} \setminus M$ term t_j can either be a (quantified) variable of $\mathcal{X}_1^\Gamma$ or a constant symbol of Dom,*
      - *for all $j \in M$ term t_j must be a (quantified) variable of $\mathcal{X}_2^\Gamma$,*

    (ii) for each atom $(X_i \equiv X_j)$ of Γ either $X_i, X_j \in \mathcal{X}_1^\Gamma$ or $X_i, X_j \in \mathcal{X}_2^\Gamma$,
    (iii) for each atom $(X_i \equiv v)$ of Γ with $v \in Dom$ variable X_i is in $\mathcal{X}_1^\Gamma$.

Then, inference-proofness *is achieved: For each instance r over $\langle R|A_R|SC_R\rangle$ with fragment instance f_1 such that $\mathcal{I}_r \models_M prior$ and for each potential secret $\Psi_i(\boldsymbol{v_i}) \in \mathrm{ex}(potsec(\mathcal{C}))$ we have $db_{f_1} \cup db_R \cup prior \not\models_{DB} \Psi_i(\boldsymbol{v_i})$,*
i.e., there exists an alternative instance r' over $\langle R|A_R|SC_R\rangle$ s.t.

(a) $\mathcal{I}_{r'} \models_M db_{f_1} \cup db_R \cup prior$, and
(b) $\mathcal{I}_{r'} \not\models_M \Psi_i(\boldsymbol{v_i})$.

Proof (sketch). Consider any $\Psi_i(\boldsymbol{v_i}) \in \mathrm{ex}(potsec(\mathcal{C}))$ with $\boldsymbol{v_i} = (v_{i_1}, \ldots, v_{i_\ell})$. Then $c_i := \{a_{i_1}, \ldots, a_{i_\ell}\} \in \mathcal{C}$, and thus by the assumptions there is an attribute $a_m \in c_i$ with $m \in M$; moreover, either $a_m \in \mathcal{E}$ or $a_m \in (\bar{A}_{F_2} \setminus \mathcal{E})$.

Starting the construction of r' and thus of the induced $\mathcal{I}_{r'}$, to ensure $\mathcal{I}_{r'} \models_M db_{f_1}$ according to Proposition 1, we define $f_1' := f_1$ and $\mathcal{I}_{r'}(F_1) := f_1'$.

Continuing the construction of $\mathcal{I}_{r'}$, we select a constant symbol $v^* \neq v_m$ from the infinite set $\mathcal{U}$ that does not occur in the finite active domain of $\pi_M(r)$ and define a bijection $\varphi : \mathcal{U} \to \mathcal{U}$ such that $\varphi(v_m) = v^*$ and no value of $\pi_M(r)$ is mapped to v_m. Then we extend φ to a tuple transformation φ^* that maps a

value v for an attribute $a_j \in A_R$ with $j \in M$ to $\varphi(v)$ and each value for an attribute $a_j \in A_R$ with $j \notin M$ to itself, and define $r' := \varphi^*[r]$. Accordingly, the predicate symbol R is interpreted by $\mathcal{I}_{r'}(R) := r'$.

The instance r' and its fragment instance f_1' together uniquely determine the corresponding fragment instance f_2' – whose constructability is guaranteed by the group properties of *Enc* – and thus we define $\mathcal{I}_{r'}(F_2) := f_2'$.

By the selection of v^* and the definition of φ, we immediately have $\mathcal{I}_{r'} \not\models_M \Psi_i(\boldsymbol{v_i})$, and thus $\mathcal{I}_{r'}$ complies with property (b). Furthermore, by the construction and according to Proposition 2, $\mathcal{I}_{r'} \models_M db_R$. Finally, we outline the argument to verify the remaining part of property (a), namely $\mathcal{I}_{r'} \models_M prior$.

We consider the following $\Gamma \in prior$ (other cases are treated similarly):

$$(\forall \boldsymbol{x})(\exists \boldsymbol{y})[\bigvee_{j=1,\ldots,p} \neg R(t_{j,1},\ldots,t_{j,n}) \vee R(t_{p+1,1},\ldots,t_{p+1,n})]\,,$$

where $\{t_{j,1},\ldots,t_{j,n}\} \subseteq \boldsymbol{x} \cup Dom$ for $j \in \{1,\ldots,p\}$ and $\{t_{p+1,1},\ldots,t_{p+1,n}\} \subseteq \boldsymbol{x} \cup \boldsymbol{y} \cup Dom$. To demonstrate $\mathcal{I}_{r'} \models_M \Gamma$, we inspect any variable substitution $\sigma' : \boldsymbol{x} \to Dom$. If there exists $j \in \{1,\ldots,p\}$ such that $\mathcal{I}_{r'}^{\sigma'} \models_M \neg R(t_{j,1},\ldots,t_{j,n})$, we are done.

Otherwise, for all $j \in \{1,\ldots,p\}$ we have $\mathcal{I}_{r'}^{\sigma'} \not\models_M \neg R(t_{j,1},\ldots,t_{j,n})$ and thus for each tuple $\mu_j' := (\sigma'(t_{j,1}),\ldots,\sigma'(t_{j,n}))$ we have $\mu_j' \in r'$. Since $r' := \varphi^*[r]$, for all $j \in \{1,\ldots,p\}$ there exists $\mu_j \in r$ such that $\varphi^*[\mu_j] = \mu_j'$. Now exploiting the properties of the set M – essentially, for each term exactly one case of the definition of φ^* applies – we can construct a variable substitution $\sigma : \boldsymbol{x} \to Dom$ such that $\mu_j = (\sigma(t_{j,1}),\ldots,\sigma(t_{j,n}))$ and, accordingly, $\mathcal{I}_r^{\sigma} \not\models_M \neg R(t_{j,1},\ldots,t_{j,n})$.

Since $\mathcal{I}_r \models_M \Gamma$, there exists a variable substitution $\tau : \boldsymbol{y} \to Dom$ such that $\mathcal{I}_r^{\sigma|\tau} \models_M R(t_{p+1,1},\ldots,t_{p+1,n})$, i.e., $\mu_{p+1} := (\sigma|\tau(t_{p+1,1}),\ldots,\sigma|\tau(t_{p+1,n})) \in r$. By the definition of r', we have $\mu_{p+1}' := \varphi^*[\mu_{p+1}] \in r'$.

Exploiting the properties of M and using τ, we can construct a variable substitution $\tau' : \boldsymbol{y} \to Dom$ such that $\mu_{p+1}' = (\sigma'|\tau'(t_{p+1,1}),\ldots,\sigma'|\tau'(t_{p+1,n}))$. Hence, $\mathcal{I}_{r'}^{\sigma'|\tau'} \models_M R(t_{p+1,1},\ldots,t_{p+1,n})$ and thus $\mathcal{I}_{r'}^{\sigma'|\tau'} \models_M \Gamma$. □

Theorem 1 provides a sufficient condition for inference-proofness on schema level, i.e., for *each* relational instance satisfying the a priori knowledge *prior*. In some situations, however, a security officer might aim at only achieving inference-proofness of a fixed particular relational instance r. Such a situation could be captured by a corollary. Essentially, if we know r and thus also f_1 in advance, we can inspect the usefulness of each implicational sentence $\Gamma \in prior$ of form $(\forall \boldsymbol{x})(\exists \boldsymbol{y})[\bigvee_{j=1,\ldots,p} \neg A_j \vee A_{p+1}]$ to derive harmful information for the specific situation. If r already satisfies $(\forall \boldsymbol{x})[\bigvee_{j=1,\ldots,p} \neg A_j]$, then we can completely discard Γ from the considerations. More generally, we could only consider the effects of Γ for those variable substitutions σ of $\boldsymbol{x}$ that make $[\bigvee_{j=1,\ldots,p} \neg A_j]$ *false* for r.

5 Creation of an Appropriate Fragmentation

If an attacker is supposed to have a priori knowledge, a fragmentation has to comply with this knowledge to guarantee inference-proofness in terms of Theorem 1.

F_1	tid	SSN	Illness	HurtBy	Doctor
	1	e_S^1	Borderline	e_H^1	White
	2	e_S^2	Laceration	e_H^2	Warren
	3	e_S^3	Laceration	e_H^3	Warren
	4	e_S^4	Concussion	e_H^4	Warren

F_2	tid	SSN	HurtBy	Name
	1	κ_S^1	κ_H^1	Hellmann
	2	κ_S^2	κ_H^2	Dooley
	3	κ_S^3	κ_H^3	McKinley
	4	κ_S^4	κ_H^4	McKinley

Fig. 3. Inference-proof fragmentation w.r.t. a priori knowledge of Example 8

Hence, an algorithm computing a fragmentation should not only determine an arbitrary fragmentation being confidential in terms of Def. 3. The algorithm should rather consider *all* of these fragmentations and select one complying with the user's a priori knowledge (if such a fragmentation exists).

Example 9. Reconsidering the a priori knowledge presented in Example 8, this knowledge does *not* compromise confidentiality if the fragmentation known from Fig. 1 is modified as depicted in Fig. 3. In terms of Theorem 1, for an attacker knowing f_1 the set M can be chosen to contain the indices of `SSN`, `HurtBy` and `Name` and for both sentences Γ_1 and Γ_2 of Example 8 the set of variables can be partitioned s.t., for both $i \in \{1,2\}$, $X_I, X_D \in \mathcal{X}_1^{\Gamma_i}$ and $X_S, X_N, X_H \in \mathcal{X}_2^{\Gamma_i}$.

In the following, an Integer Linear Program (ILP) (see [11]) computing a confidential fragmentation complying with an attacker's a priori knowledge is developed to solve this problem with the help of generic algorithms solving ILPs.[2] As the optimization goal the set of "encrypted attributes" is chosen to be minimized to reduce the costs for processing queries over the fragmented database as proposed in [2,9]. Other optimization goals are conceivable, too.

Given the attribute set A_R of an original schema $\langle R|A_R|SC_R\rangle$, a set $\mathcal{C}$ of confidentiality constraints and a set *prior* in terms of Theorem 1, the ILP presented in the following computes the attribute sets $\bar{A}_{F_1}$ and $\bar{A}_{F_2}$ as well as the set $\mathcal{E}$ of "encrypted attributes" of a fragmentation being confidential w.r.t to $\mathcal{C}$ and complying with *prior*. The ILP contains the following binary decision variables:

- A variable a_j^i, for both $i \in \{1,2\}$ and for each $a_j \in A_R$. If $a_j^i = 1$, attribute $a_j \in A_R$ is in $\bar{A}_{F_i}$; if $a_j^i = 0$, attribute $a_j \in A_R$ is *not* in $\bar{A}_{F_i}$.
- A variable a_j^e for each $a_j \in A_R$. If $a_j^e = 1$, attribute $a_j \in A_R$ is an "encrypted attribute"; if $a_j^e = 0$, attribute $a_j \in A_R$ is a "cleartext attribute".
- A variable m_j for each $a_j \in A_R$. If $m_j = 1$, the index of attribute a_j is in M; if $m_j = 0$, the index of attribute a_j is *not* in M.
- A variable X^Γ for each variable X contained in a sentence $\Gamma \in$ *prior*. If $X^\Gamma = 1$, variable X is in $\mathcal{X}_1^\Gamma$; if $X^\Gamma = 0$, variable X is in $\mathcal{X}_2^\Gamma$.

For each $\Gamma \in$ *prior* the set Var_j^Γ is assumed to contain X^Γ if Γ is built over an atom $R(t_1,\ldots,t_n)$ with t_j being the variable X (note that each variable might occur in different columns). Moreover, the set $const(\Gamma)$ is assumed to contain the index j, if Γ is built over an atom $R(t_1,\ldots,t_n)$ with t_j being a constant. Then, the ILP computing an appropriate fragmentation is defined as follows:

[2] For our prototype implementation "lp_solve" turned out to be an appropriate and fast ILP solver (see `http://lpsolve.sourceforge.net/`).

Minimize the number of "encrypted attributes", i.e., min: $\sum_{j=1}^{n} a_j^e$ s.t. the following constraints are fulfilled:

- "Cleartext attributes" in exactly one fragment, "encrypted ones" in both:
  $a_j^1 + a_j^2 = 1 + a_j^e$ for each $a_j \in A_R$
- For $i \in \{1,2\}$, fragment $\langle F_i | A_{F_i} | SC_{F_i} \rangle$ fulfills all confidentiality constraints:
  $\sum_{a_j \in c} a_j^i \leq |c| - 1 + \sum_{a_j \in c} a_j^e$ for each $c \in \mathcal{C}$ and each $i \in \{1,2\}$
- $M \subseteq \{h+1, \ldots, n\}$, i.e., M is a subset of attributes in A_{F_2}:
  $m_j \leq a_j^2$ for each $a_j \in A_R$
- M overlaps with the indices of the attributes of each $c \in \mathcal{C}$:
  $\sum_{a_j \in c} m_j \geq 1$ for each $c \in \mathcal{C}$
- For each formula $\Gamma \in \mathit{prior}$:
  - In each $R(t_1, \ldots, t_n)$ of Γ: for each t_j being a constant with $j \notin M$:
    $m_j = 0$ for each $j \in \mathit{const}(\Gamma)$
  - Partitioning of variables into $\mathcal{X}_1^\Gamma$ and $\mathcal{X}_2^\Gamma$:
    $X^\Gamma = 1 - m_j$ for $j \in \{1, \ldots, n\}$ with $\mathit{Var}_j^\Gamma \neq \emptyset$ and each $X^\Gamma \in \mathit{Var}_j^\Gamma$
  - In each atom $(X_i \equiv X_j)$: variables X_i, X_j belong to the same partition:
    $X_i^\Gamma = X_j^\Gamma$ for each atom $(X_i \equiv X_j)$
  - In each atom $(X \equiv v)$: variable X belongs to partition $\mathcal{X}_1^\Gamma$:
    $X^\Gamma = 1$ for each atom $(X \equiv v)$
- Each decision variable of this ILP is binary:
  $0 \leq x \leq 1$ for each integer decision variable x of this ILP

If the ILP solver outputs a feasible solution, an inference-proof fragmentation can be determined by constructing the sets $\bar{A}_{F_1}$, $\bar{A}_{F_2}$ and $\mathcal{E}$ of Def. 1 according to the allocation of the corresponding decision variables of the ILP.

Note that availability requirements such as storing a particular subset of attributes within the same (or even a particular) fragment or keeping the values of a particular attribute as cleartext values can be simply modelled by adding appropriate constraints, i.e., (in-)equations, to the ILP.

6 Conclusion and Future Work

Motivated by the question, whether splitting of data vertically over two semi-honest servers guarantees confidentiality, the fragmentation model introduced in [2,9] is formalized, then modelled logic-orientedly and subsequently analyzed w.r.t. its inference-proofness. This analysis considers an attacker employing his a priori knowledge to draw harmful inferences and provides a sufficient condition to decide whether a given combination of a fragmentation and a priori knowledge is inference-proof w.r.t. a given confidentiality policy. Additionally, a generic ILP formulation computing such an inference-proof fragmentation is developed.

As Theorem 1 only states a sufficient condition for inference-proofness, there might be a more relaxed, most desirably even necessary definition of a priori knowledge still guaranteeing inference-proofness. A full characterization of inference-proofness could also provide a basis for deciding on the existence of a secure fragmentation for a given setting.

Theorem 1 might be also enhanced in the spirit of k-anonymity by a more sophisticated definition of confidentiality guaranteeing that an "invisible value" cannot be narrowed down to a set of possible values of a certain cardinality. A further analysis of confidentiality assuming that commonly used encryption functions such as AES or RSA (which do *not* satisfy the group properties) come into operation is desirable, too. Although a formal analysis based on probability theory and complexity theory is indispensable to guarantee profound statements, we expect these encryption functions to be "sufficiently secure" in practice.

In this article and previously in [5] each one of two existing approaches to achieve confidentiality by vertical fragmentation is analyzed. As a third approach – using an arbitrary number of fragments which are *all* supposed to be known to an attacker – is presented in [8], a formal analysis of this approach in the spirit of Theorem 1 might be another challenging task for future work.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: A distributed architecture for secure database services. In: CIDR 2005, pp. 186–199 (2005)
3. Biskup, J.: Inference-usability confinement by maintaining inference-proof views of an information system. International Journal of Computational Science and Engineering 7(1), 17–37 (2012)
4. Biskup, J., Bonatti, P.A.: Controlled query evaluation with open queries for a decidable relational submodel. Annals of Mathematics and Artificial Intelligence 50(1-2), 39–77 (2007)
5. Biskup, J., Preuß, M., Wiese, L.: On the Inference-Proofness of Database Fragmentation Satisfying Confidentiality Constraints. In: Lai, X., Zhou, J., Li, H. (eds.) ISC 2011. LNCS, vol. 7001, pp. 246–261. Springer, Heidelberg (2011)
6. Biskup, J., Wiese, L.: A sound and complete model-generation procedure for consistent and confidentiality-preserving databases. Theoretical Computer Science 412(31), 4044–4072 (2011)
7. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Keep a few: Outsourcing data while maintaining confidentiality. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 440–455. Springer, Heidelberg (2009)
8. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Combining fragmentation and encryption to protect privacy in data storage. ACM Transactions on Information and System Security 13(3) (2010)
9. Ganapathy, V., Thomas, D., Feder, T., Garcia-Molina, H., Motwani, R.: Distributing data for secure database services. Transactions on Data Privacy 5(1), 253–272 (2012)
10. Hacigümüs, H., Mehrotra, S., Iyer, B.R.: Providing database as a service. In: ICDE 2002, pp. 29–40. IEEE Computer Society, Los Alamitos (2002)
11. Korte, B., Vygen, J.: Combinatorial Optimization: Theory and Algorithms, 5th edn. Algorithms and Combinatorics. Springer, Heidelberg (2012)

Differentially Private Multi-dimensional Time Series Release for Traffic Monitoring

Liyue Fan, Li Xiong, and Vaidy Sunderam

Emory University
Atlanta GA 30322, USA
{lfan3,lxiong,vss}@mathcs.emory.edu

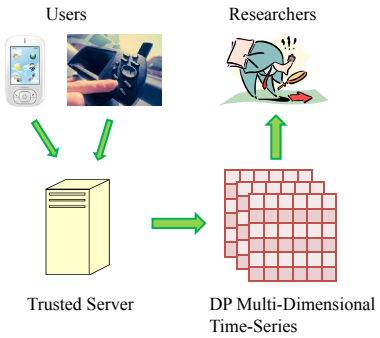
Abstract. Sharing real-time traffic data can be of great value to understanding many important phenomena, such as congestion patterns or popular places. To this end, private user data must be aggregated and shared continuously over time with data privacy guarantee. However, releasing time series data with standard differential privacy mechanism can lead to high perturbation error due to the correlation between time stamps. In addition, data sparsity in the spatial domain imposes another challenge to user privacy as well as utility. To address the challenges, we propose a real-time framework that guarantees differential privacy for individual users and releases accurate data for research purposes. We present two estimation algorithms designed to utilize domain knowledge in order to mitigate the effect of perturbation error. Evaluations with simulated traffic data show our solutions outperform existing methods in both utility and computation efficiency, enabling real-time data sharing with strong privacy guarantee.

Keywords: Traffic Monitoring, Multi-Dimensional Time-Series, Differential Privacy.

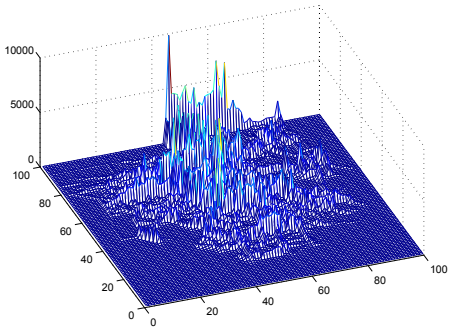
1 Introduction

Sharing real-time traffic data is essential to discovering useful and previously unknown knowledge. As illustrated in Figure 1(a): a wireless service provider gathers data from individual users about their locations, speeds, mobility, etc. The aggregated data, e.g. the number of users present at certain locations during a given time period, can be shared with third party researchers to be mined for commercial interest, such as popular places, as well as public interests, such as congestion trends. Figure 1(b) provides a snapshot of aggregated traffic data at a single time stamp. As is shown, the two-dimensional space is partitioned by a 100×100 grid. For each cell in the 2D space, Figure 1(b) plots the number of users within its extent at the given time stamp. Since the spatial distribution of wireless users could change over time due to movement, such a snapshot is needed at every time stamp in order to perform real-time data mining tasks. However, the privacy of individual users may be affected if their private data is shared with untrusted third parties. The goal of our work is to enable the server/data

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 33–48, 2013.



(a) Privacy-Preserving Data Sharing



(b) A Snapshot of 2D Time Series

Fig. 1. Traffic Data Monitoring

holder to share useful multi-location aggregates continuously (multi-dimensional time series) while preserving individual privacy.

The current state-of-the-art paradigm for privacy-preserving data publishing is *differential privacy* [1], denoted as "DP" in Figure 1(a). Differential privacy requires that the aggregate statistics reported by a data publisher be perturbed by a randomized algorithm $\mathcal{A}$, so that the output of $\mathcal{A}$ remains roughly the same even if any single tuple in the input data is arbitrarily modified. This ensures that given the output of $\mathcal{A}$, an adversary will not be able to infer much about any single tuple in the input, and thus privacy is protected.

Despite the large number of methods on differentially private data publication [4–6, 10, 11, 15–17], there does not currently exist an approach to sharing multi-dimensional time series data. We summarize our challenges below:

- Due to the data correlation between time stamps, a straightforward application of the standard differential privacy mechanism at every time stamp leads to an overall perturbation error of $\Theta(T)$ by composition theorem [12], where T is the length of the time series, which severely limits the utility of the published data when T is large.
- Another challenge is data sparsity in the spatial domain. As shown in Figure 1(b), the majority of cells in the 2D space have very low to zero frequency. In reality, the total number of cells can be very large with respect to the total number of users. The data sparsity poses great challenge for privacy-preserving techniques since the perturbation noise is likely to dominate the released value in presence of a small set of users.
- Furthermore, the monitoring application requires that private, released data is provided in real-time. Therefore, existing techniques that require time-series transformation or prohibitive computation time are not applicable to performing real-time tasks.

Our Contributions. In this paper, we propose a real-time framework and two estimation algorithms to address the above challenges in multi-location traffic monitoring with differential privacy. Domain knowledge, such as road network and density, is utilized by our solutions to model the auto-correlation of individual cells over time as well as correlation between neighboring cells. The temporal estimation algorithm establishes an internal time series model for each individual cell and performs posterior estimation to improve the utility of shared aggregate per time stamp. The spatial estimation algorithm builds a spatial indexing structure based on Quadtree to group similar cells together and to reduce the impact of data sparsity. Our solutions provide a strong privacy guarantee. Both algorithms outperform baseline solution as well as state-of-the-art methods in sharing time series or static multi-dimensional data, providing real-time data release without compromising the utility of shared data.

The rest of the paper is organized as follows: Section 2 provides the problem definition, preliminaries on differential privacy, and the baseline solution. Section 3 presents the technical details of our proposed solutions, i.e. temporal estimation and spatial estimation. Section 4 presents a set of empirical results. Section 5 reviews previous works related to data sharing methods with differential privacy. Section 6 concludes the paper and states possible directions for future work.

2 Problem Statement and Preliminaries

2.1 Problem

In the traffic monitoring application we consider, a set of objects are moving in a two-dimensional space and a central server is collecting information about their locations over time. We adopt a fine-grained 2D grid that partitions the space G into $w \times w$ cells, where w is a constant number called *resolution*. We further assume the expected collection time span is T and denote k as the discrete time index where $0 \leq k < T$. For each cell c in G, we define the *frequency series* of c as $\mathbf{X}^c = \{x_k^c |\ 0 \leq k < T\}$, where x_k^c represents the number of objects within its extent at time stamp k. A multi-dimensional time series $\mathbf{X}^G$ can be defined as the set of frequency series of every cell c in G, i.e. $\mathbf{X}^G = \{\mathbf{X}^c |\ c \in G\}$. A *snapshot* of the spatio-temporal database $\mathbf{X}_k^G$ is defined as the set of cell frequencies at time k, i.e. $\mathbf{X}_k^G = \{x_k^c |\ c \in G\}$. The same terms for the released data set $\mathbf{R}^G$ can be defined similarly.

Problem 1. *Given a multi-dimensional time series $\mathbf{X}^G$ where $G = w \times w$ cells, for each snapshot $\mathbf{X}_k^G$, release in real-time a sanitized version $\mathbf{R}_k^G$ such that the overall release $\mathbf{R}^G$ satisfies α-differential privacy, where α is a user-specified privacy level.*

Note that sharing $\mathbf{R}^G$ will enable a variety of data mining tasks. Therefore we use a generic utility metric, i.e. relative error, to measure the usefulness of the released series for each cell c:

Definition 1 (Utility Metric). The utility of a published series $\mathbf{R}^c = \{r_k^c\}$ can be measured by the *average relative error*, denoted as E^c, against the original time-series $\mathbf{X}^c = \{x_k^c\}$.

$$E^c = \frac{1}{T}\sum_{k=0}^{T-1}\frac{|r_k^c - x_k^c|}{max\{x_k^c, \delta\}} \tag{1}$$

where δ is a user-specified constant (also referred to as *sanitary bound* as in [14]) to mitigate the effect of excessively small query results, e.g. $0's$. Here we set $\delta = 1$ throughout the entire time-series for all cells.

2.2 Differential Privacy

The privacy guarantee provided by our solutions is *differential privacy* [1]. Simply put, a mechanism is differentially private if its outcome is not significantly affected by the removal or addition of a single user. An adversary thus learns approximately the same information about any individual user, irrespective of his/her presence or absence in the original database.

Definition 2 (α-Differential Privacy [1]). A non-interactive privacy mechanism $\mathcal{A}$ gives α-differential privacy if for any dataset D_1 and D_2 differing on at most one record, and for any possible anonymized dataset $\widetilde{D} \in Range(\mathcal{A})$,

$$Pr[\mathcal{A}(D_1) = \widetilde{D}] \leq e^{\alpha} \times Pr[\mathcal{A}(D_2) = \widetilde{D}] \tag{2}$$

where the probability is taken over the randomness of $\mathcal{A}$.

The privacy parameter α, also called the *privacy budget* [12], specifies the degree of privacy offered. Intuitively, a lower value of α implies stronger privacy guarantee and a larger perturbation noise, and a higher value of α implies a weaker guarantee while possibly achieving higher accuracy. We will examine the privacy-utility tradeoff in the experiment section.

Laplace Mechanism. Dwork et al. [5] show that α-differential privacy can be achieved by adding i.i.d. noise $\tilde{N}$ to each query result $q(D)$:

$$\tilde{q}(D) = q(D) + \tilde{N} \tag{3}$$

$$p(\tilde{N} = x) = \frac{1}{2\lambda}e^{-|x|/\lambda}, \quad \lambda = GS(q)/\alpha \tag{4}$$

The magnitude of $\tilde{N}$ conforms to a Laplace distribution in Equation (4) where $GS(q)$ represents the *global sensitivity* [5] of a query q. In the traffic monitoring application, each aggregate value is a *count* query and $GS(count) = 1$. Later on in this paper, we denote the Laplace distribution with 0 mean and λ scale as $Lap(0, \lambda)$.

Composition. The composition properties of differential privacy provide privacy guarantees for a sequence of computations, e.g. a sequence of *count* queries.

Theorem 1 (Sequential Composition [12]). Let $\mathcal{A}_i$ each provide α_i-differential privacy. A sequence of $\mathcal{A}_i(D)$ over the dataset D provides $(\sum_i \alpha_i)$-differential privacy.

Algorithm 1. Laplace Perturbation Algorithm(LPA)

Input: Raw data series $\mathbf{X}^G$, privacy budget α
Output: Released data series $\mathbf{R}^G$

1: **for** each cell $c \in G$ **do**
2: **for** each time stamp k **do**
3: $r_k^c \leftarrow$ perturb x_k^c by $Lap(0, \frac{T}{\alpha})$;

2.3 Baseline Solution

A baseline solution to sharing differentially private multi-dimensional time series is to apply the standard Laplace perturbation at each time stamp to every frequency series. For any c, if every released aggregate satisfies α/T-differential privacy, by Theorem 1 the released frequency series guarantees α-differential privacy. We summarize the baseline algorithm in Algorithm 1 and Line 3 represents the Laplace mechanism to guarantee α/T-differential privacy for each released aggregate. Empirical studies of the LPA algorithm against our proposed solutions are included in Section 4.

3 Proposed Solutions

In this section, we present our proposed solutions for privacy-preserving traffic monitoring. Figure 2 provides a high-level overview of the system framework. At every time stamp, the input multi-dimensional data is perturbed by the *Laplace Perturbation* mechanism to guarantee differential privacy. Then the perturbed data can be post-processed by the *Estimation* module to produce a more accurate, released version. Domain knowledge, such as road network and population density, is utilized by *Modeling/Aggregation*, which in return interacts with the perturbation component as well as the estimation method in use. Below we describe in detail two separate estimation algorithms: one is to perform time-wise estimation for each individual cell, while the other is to perform spatial aggregation and estimation over the entire 2D space.

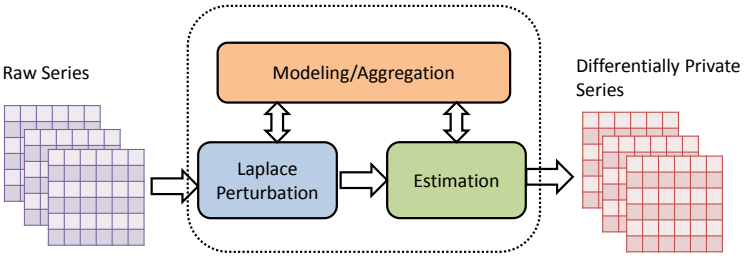


Fig. 2. Differentially Private Traffic Monitoring Framework

3.1 Temporal Estimation

For each cell c in space G, we can apply our recently proposed filtering-based posterior estimation technique [9] to the cell frequency series $\mathbf{X}^c$. The key idea is to utilize an internal time series model for the frequency series and to estimate the true aggregate values based on the Laplace perturbed values. The additional innovation in this paper is that we model different types of cells according to the domain knowledge on the road networks. Below we briefly show how to model the cell frequency series and refer interested readers to our work [9] for further implementation details.

Note that the internal model of cell frequencies depends on many factors, such as location, overall population, road network, etc. Here we simply classify each cell as *sparse* or *dense* based on road network connections and assume the same internal model for cells within each category. For each cell c, its frequency series $\mathbf{X^c}$ can be represented by the following process model:

$$x_{k+1}^c = x_k^c + \omega^c \ , \quad p(\omega_c) \sim \mathbb{N}(0, Q^c) \tag{5}$$

which states that the count values of consecutive time stamps should be consistent except for a white, Gaussian noise ω^c. In particular, ω^c is called the process noise and it follows a normal distribution. Q^c value indicates the level of variation between adjacent time stamps. Intuitively, sparse cells exhibit little variation since very few objects travel within them, therefore we should specify a small Q^c value for such cells. On the other hand, higher Q^c should be assigned for dense cells since they are visited more frequently in reality.

The noisy observation, which is obtained from the Laplace Perturbation mechanism, can be modeled as follows:

$$z_k^c = x_k^c + \nu \ , \quad \nu \sim Lap(0, 1/\alpha_0) \tag{6}$$

where ν, called the measurement noise, corresponds to the Laplace noise and is independent of c. The differential privacy budget for each traffic count is $\alpha_0 = \alpha/T$, since the overall privacy budget α is uniformly allocated to each time stamp.

For posterior estimation purpose, it is sufficient and computationally attractive to approximate ν by a white Gaussian error according to [9]:

$$\nu \sim \mathbb{N}(0, R) \ . \tag{7}$$

Therefore here we adopt the above Gaussian approximation for every cell and use the Kalman filter based filtering technique [9] for posterior estimation.

The outline of the temporal estimation algorithm is presented in Algorithm 2. For every time stamp k and each cell c, we derive a predicted frequency with the *Predict* procedure. Upon receiving the noisy observation, we can derive a posterior estimate with the *Correct* procedure, by linear combination of prediction and observation. The derivation of posterior estimate as well as *Predict* and *Correct* steps can be found in [9] and therefore omitted here for brevity.

Algorithm 2. Temporal Estimation Algorithm

Input: Raw data series $\mathbf{X}^G$, privacy budget α
Output: Released data series $\mathbf{R}^G$

1: **for** each timestamp k **do**
2: **for** each cell $c \in G$ **do**
3: $prior \leftarrow c.\mathbf{Predict}(k)$;
4: $z_k^c \leftarrow$ perturb x_k^c by $Lap(0, \frac{T}{\alpha})$;
5: $posterior \leftarrow c.\mathbf{Correct}(k, prior, z_k^c)$;
6: $r_k^c \leftarrow posterior$;

The advantage of temporal estimation approach is that it utilizes the internal time series model and the observations to form an educated guess, which is shown in [9] to greatly improve the accuracy of released data per time stamp. As for complexity, we can see that the computation time requirement is $O(w^2)$ for every time stamp where w is the spatial resolution, since only $O(1)$ operations are performed for each cell.

3.2 Spatial Estimation

When every cell is perturbed individually, data sparsity imposes great utility challenge, i.e. high relative error due to perturbation. We thus are motivated to group similar cells to overcome the data sparsity issue. Considering the spatial correlation among cells, it is very likely that neighboring cells are connected by the same roads therefore are more similar to each other. To utilize this heuristic, we propose to aggregate similar cells into partitions according to spatial vicinity and perform estimation within each partition assuming uniformly distributed objects within the partitions.

We propose a top-down space partitioning approach based on Quadtree due to several considerations. One advantage of Quadtree is its efficiency: it recursively partitions a 2D space into 4 quadrants disregard the actual object distribution in the space. Another advantage of Quadtree is that it doesn't incur any extra privacy cost due to its independence from data. In contrast, the kdTree structure proposed by Cormode et al [4] does require extra privacy budget spent on finding the "private median". Since the privacy budget for each time stamp is very limited, we believe that Quadtree is more suitable in the multi-dimensional time series scenario.

We outline the spatial aggregation algorithm based on Quadtree in Algorithm 3. Line 5 checks every node/partition for the splitting condition. Line 6 splits a partition into four equal quadrants. The *node*.homogeneous() method returns true if all the cells within the partition belong to the same category. Again, each cell is pre-classified as *sparse* or *dense* based on domain knowledge. We stop splitting a partition if it is homogeneous. Otherwise, as long as the predefined depth threshold d is not violated, we further split the partition in the hope of reducing the class impurity in each child partition. The value of d represents

Algorithm 3. QuadTreeAgg Algorithm

Input: 2D grid G, depth threshold d
Output: QuadTree index structure QT

```
1: QT.root ← G;
2: queue.add(QT.root) ;
3: while ! queue.empty() do
4:     node ← queue.remove() ;
5:     if ! node.homogeneous() and node.depth < d
6:         node.split() ;
7:         queue.add(node.children) ;
```

Algorithm 4. Spatial Estimation Algorithm

Input: Raw data series $\mathbf{X}^G$, depth threshold d, privacy budget α
Output: Released data series $\mathbf{R}^G$

```
1: QT ← QuadTreeAgg(G, d); # initialize the quadtree index
2: for each timestamp k do
3:     for each partition p ∈ QT do
4:         p_k ← Σ_{c∈p} x_k^c ;
5:         p̃_k ← perturb p_k by Lap(0, T/α);
6:         r_k^c ← p̃_k/p.size(), c ∈ p ;
```

the aggregation level. Setting $d = 0$ implies that all cells are aggregated in one partition. Since the uniform assumption within the partition does not hold, high estimation error will be incurred. On the other hand, a higher value of d implies that many partitions will be further split to produce homogeneous regions so as to reduce estimation error. However, due to data sparsity, very few moving objects will fall into each partition when it is small. Therefore, the perturbation error will dominate the released data in that case. Clearly the optimal d value depends on the spatial distribution of cells. We will examine the impact of d in the experiment section.

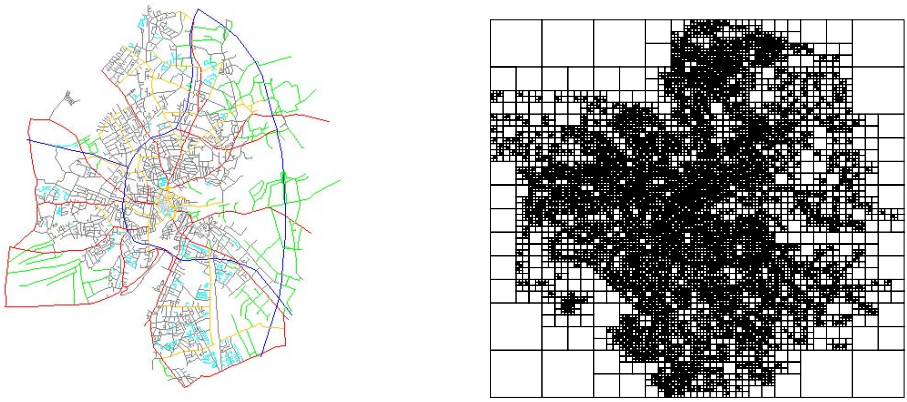
Once the Quadtree index structure of the space G is established, we assume uniform data distribution within each partition and estimate each cell frequency with average partition frequency. The spatial estimate algorithm is described in Algorithm 4. For each time stamp k, a partition count is aggregated from cells for every partition (Line 4). It is then perturbed by the Laplace mechanism to guarantee differential privacy (Line 5) and the average noisy count is used to estimate the frequency of each cell within the partition (Line 6). The intuition is that the cells within each partition have similar density. Therefore by uniformly distributing the noisy partition count to each cell, we reduce the magnitude of perturbation error applied to each cell without compromising the accuracy.

One advantage of the spatial estimation algorithm is that it relies on simple and practical assumptions. The complexity is also $O(w^2)$ for each time stamp

since every cell is visited $O(1)$ times. Although it takes extra time to build the spatial index for initialization, we see it as a one-time cost which can be done off-line. The runtime of the spatial estimation is reduced because only one perturbation noise is needed for every partition at every k (Line 5). In contrast, both the baseline LPA algorithm and the temporal estimation algorithm will generate one perturbation noise for each cell at every k. We will study their runtime performance in the next section.

4 Evaluation

We implemented the proposed algorithms as well as alternative methods in Java with JSC[1] for simulating the statistical distributions. All experiments were conducted using a 2.90GHz Intel Core i7 PC with 8GB RAM.



(a) Map of Oldenburg (b) Partitions by Quadtree

Fig. 3. Overview of Data Set

Data Set. We generated synthetic traffic data with the Brinkhoff generator [2]. The input of the generator is the road map of Oldenburg in Germany[2] (Figure 3(a)),which contains 6,105 nodes and 7,035 edges, and the output is a set of moving objects on the road network. We created the data set with 100 discrete timestamps, with 500,000 objects at the beginning and 25,000 new objects introduced at every time stamp. The starting positions and destinations of the moving objects are selected randomly by the generator (see [2] for detailed network-based techniques). Once an object reaches its destination, it disappears from the map. At the server side, we use a 2D grid with 1024×1024 cells to record the locations of the moving objects, with each cell representing approximately 20×20 square meters' range in reality. We assign each cell a class label,

[1] `http://www.jsc.nildram.co.uk`
[2] `http://iapg.jade-hs.de/personen/brinkhoff/generator/`

i.e. *sparse* or *dense*, based on the presence of roads within its extent. Roughly 95% cells are labeled *sparse*, indicating only the rest 5% have been visited by the moving objects. Figure 3(b) visualizes the partition result achieved by the Quadtree-based algorithm with the depth threshold $d = 8$. It can be seen that spares regions around map edge are contained in larger partitions and densely connected regions in map center are further split into smaller partitions. We will evaluate the set of *sparse* cells and the set of *dense* cells separately since they exhibit very different dynamics over time.

Comparison. We compare our proposed solutions against the state-of-the-art methods which are summarized below:

- **DFT** is the Fourier Perturbation Algorithm recently proposed by Rastogi and Nath [13] for sharing single time series. It first performs the Discrete Fourier Transform on an input time series and retains only the first l DFT coefficients. Those coefficients are then perturbed by the Laplace mechanism to guarantee differential privacy. Finally, the Inverse Discrete Fourier Transform is performed on the perturbed coefficients to produce a released series. The number of coefficients to preserve, i.e. l, is a user-specified parameter. In our empirical study, we set $l = 20$ according to their recommendation [13].
- **kd-hybrid** is proposed by Cormode et al [4] as their best method to achieve differentially private space decomposition with static data. Without the help of a grid, *kd-hybrid* builds a mixture index over the 2D data space that begins with kd-tree and switches to quad-tree at a certain level. They slightly modified the kd-tree algorithm, changing the fanout rate to 4 in order to reduce the privacy budget consumption. According to their studies, *kd-hybrid* is most reliable among several representative differentially private space partitioning methods. They reported the optimal parameter setting empirically with the height set to 8 and the switch level set to 4.

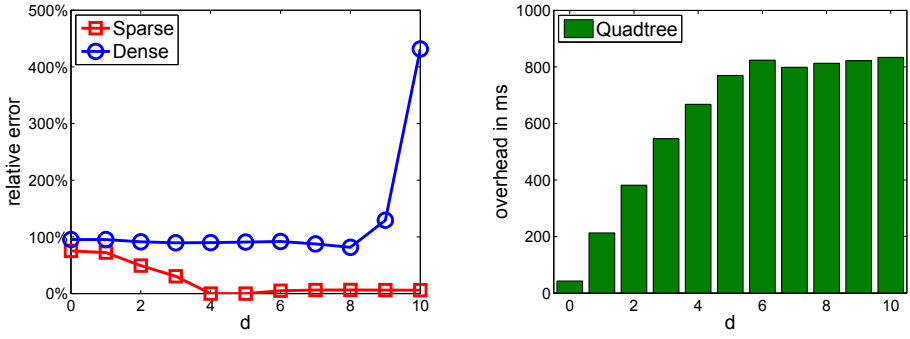
Since the DFT algorithm can be only performed with the complete series, it is not compatible to real-time applications. However, we include it in our evaluation since it serves as a good, off-line reference for utility. As for the *kd-hybrid* algorithm, there are two limitations. One is its high privacy cost since the algorithm iteratively spends budget on finding "private medians" for every data snapshot. The other limitation is its high computation cost: application of the *kd-hybrid* method requires constructing the index structure at every time stamp. Experiments with the author's provided implementation take hours for each iteration, since the domain size and the number of objects in our data set are extremely large. We conclude that the *kd-hybrid* method is too expensive for the continuous, real-time applications and therefore do not include the results in the remaining section.

Table 1. Parameter Settings

Symbol	Description	Default Value
α	Total Privacy Budget	1
w	Resolution for Each Dimension	1024
T	Length of Multidimensional Time Series	100
Q_{sparse}	Process Noise for *Sparse* Cells	10^{-2}
Q_{dense}	Process Noise for *Dense* Cells	10^{3}
R	Gaussian Measurement Noise	10^{6}
d	Depth Threshold for Quadtree	8

4.1 Parameter Impacts

The default parameter setting, unless otherwise noted, is summarized in Table 1. Note that Q_{sparse} and Q_{dense}, which correspond to Q^c in Equation (5) for *sparse* and *dense* cells, can be chosen by domain users and our default setting may not be optimal. As for R from Equation (7), we set its value according to our previous studies [8], which shows that the optimally R is proportional to T^2/α^2.

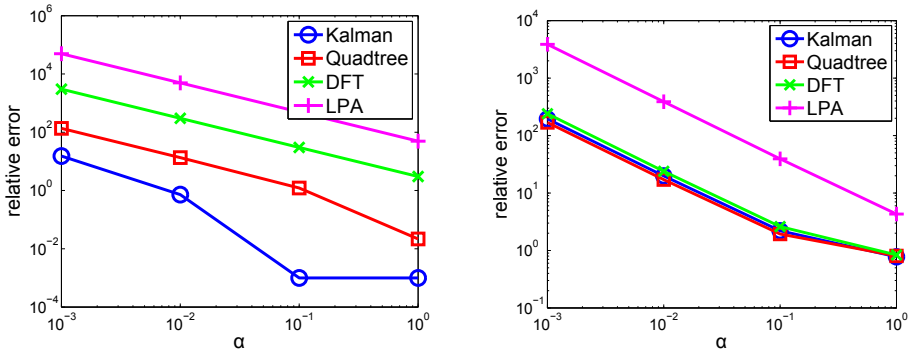


(a) Utility vs. Depth Threshold (b) Overhead vs. Depth Threshold

Fig. 4. Impact of Depth Threshold on Quadtree-based Spatial Estimation

We study the impact of the depth threshold d used in Algorithm 3 in terms of utility as defined in Equation (1) and runtime. Intuitively, the larger value d takes, the finer partitions the algorithm results in, especially along the border of sparse and dense regions. However, it also incurs a higher overhead to construct the index as we can expect. Figure 4(a) plots separately the utility of released series for sparse cells and dense cells when varying the depth threshold d. For each class of cells, we plot the median relative error to avoid the extremely small or large values. As we increase the threshold value, the error for sparse cells gradually drops to 0 between $d = 0$ and $d = 4$ and remains stable when d value is further increased. This is due to the fact that majority sparse cells are located together (on map edge) and will not take too many splits to be separate

from dense cells (in map center). Increasing the d value can help separating those sparse cells on the boarder line. However, the utility of majority sparse cells is not affected since those on the boarder line only count for a very small percentage. On the other hand, dense cells require more splits to achieve optimal separation ($d = 8$). When further split ($d > 8$), the perturbation noise greatly impacts their utility due to data sparsity. Figure 4(b) shows the overhead for constructing the aggregation index when varying the d value. It takes at most 0.9 second and we note that it is a one-time cost. As we expect, a higher depth threshold requires more construction time (from $d = 0$ to $d = 6$). However, when $d > 6$ the overhead does not grow since there are only very few partitions that do not meet the homogeneous requirement at depth 6. As can be seen in Figure 3(b), the densely connected areas in the map are split into finer partitions compared to less populous areas on the map edge.



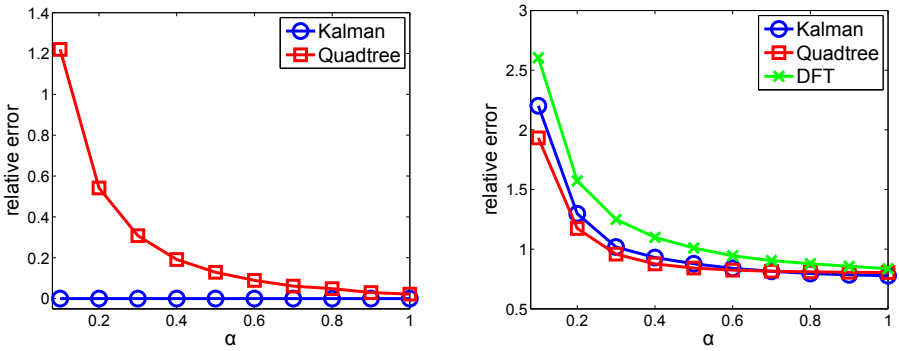
(a) Utility vs. Privacy with *Sparse* Cells (b) Utility vs. Privacy with *Dense* Cells

Fig. 5. Utility of Individual Cells: Comparison of All Methods

4.2 Utility Performance

Utility vs. Privacy. Here we examine the trade-off between utility and privacy. Our proposed solutions, i.e. `Kalman` and `Quadtree`, are compared against the baseline LPA and state-of-the-art DFT algorithm, in terms of utility of individual cells. Figure 5(a) and Figure 5(b) plot the utility of sparse cells and dense cells respectively when varying the overall privacy budget, i.e. α value, from $[10^{-3}, 10^{0}]$. As we can see, the baseline LPA algorithm results in highest relative error in both figures. The DFT algorithm results in high relative error with sparse cells even with high privacy cost ($\alpha = 1$), due to the perturbation and reconstruction error. Our solutions `Kalman` and `Quadtree` outperform both LPA and DFT especially with sparse cells, as `Quadtree` only results in 10% error and `Kalman` produces 0% error when $\alpha = 1$. As for the dense cells, both `Kalman` and `Quadtree` slightly outperforms DFT, which is supposed to be optimal. When $\alpha = 1$, DFT results in 83% error due to lack of smoothness in the original frequency series, while our solutions provide comparable utility to DFT

and real-time data release. Figure 6(a) and Figure 6(b) provide a closer look at the utility curves within a more practical range of privacy budget $\alpha \in [0.1, 1]$. DFT and LPA are not plotted in one or both figures because the errors they result in are prohibitive. For sparse cells, `Kalman` provides optimal performance even under small privacy budget ($\alpha = 0.1$), thanks to the accurate modeling. `Quadtree` is able to approach 0% error as α value increases. For dense cells, we observe that `Quadtree` provides the best utility in the same privacy budget range. We conclude that both our proposed solutions outperforms existing methods, allowing for real-time data sharing without compromising the utility.



(a) Utility vs. Privacy with *Sparse* Cells (b) Utility vs. Privacy with *Dense* Cells

Fig. 6. Closer Look at $\alpha \in [0.1, 1]$

Utility of Range Queries. Here we evaluate our solutions with range queries, where each query is a square window that covers a neighborhood of $m \times m$ cells. For each m value, we randomly generate 100 queries of size $m \times m$, evaluate each method with the same set of queries, and plot the average relative error. Note that when $m = 1$, each set query consists of one cell only and therefore the set query error is equivalent to individual cell error. Our findings are summarized in Figure 7. Our temporal estimation algorithm based on the Kalman filter clearly outperforms `Quadtree` and `LPA` with smaller query windows ($m \leq 100$). For all three methods, the relative error shows a growing trend to different extent as the query set size increases, mainly due to the data sparsity in the space. When $m = 500$, we observe that the error of `Kalman` keeps accumulating while `Quadtree` and `LPA` show reduced relative error. We believe that it is because `Kalman` does not explicitly utilize the spatial correlation between cells. When querying the entire space ($m = 1024$), both `Quadtree` and `LPA` provide good utility because the Laplace noise added to each cell is from a zero-mean distribution and the sum of a large set of such noises is likely to be small. Overall, `Quadtree` outperforms `LPA` by making sound estimation within close-to-uniform partitions.

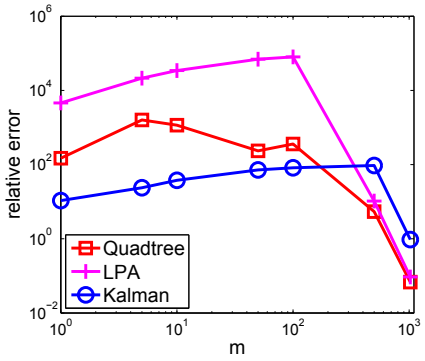


Fig. 7. Utility of Range Queries

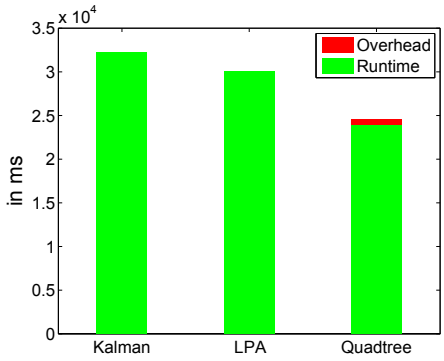


Fig. 8. Runtime Performance

4.3 Runtime Performance

Lastly we compare the runtime performance of our solutions against the baseline since computation time is critical to real-time applications. We measure and plot the runtime for releasing the two-dimensional aggregates for 100 timestamps in order to mitigate random disturbance from the operating system. The results are summarized in Figure 8. As we can see, all three methods take less than 35 seconds to release 100 snapshots of 1024×1024 cell frequencies with differential privacy guarantee. Note that the state-of-the-art *kd-hybrid* takes hours to release/evaluate one time stamp. Compared to `LPA`, our solution `Kalman` takes roughly 2 more seconds in total to perform prediction and correction at every time stamp. `Quadtree` turns out to be the most time efficient, even though it has a small overhead in building the spatial index. This is because less perturbation is performed by `Quadtree`, as at every time stamp we only generate one perturbation noise for each partition rather than for each cell as in `LPA`.

5 Related Works

Here we briefly review the most relevant, recent works on differential privacy and time-series data sharing. Dwork et al. [5] established the guideline to guarantee differential privacy for individual aggregate queries by calibrating the Laplacian noise to the global sensitivity of each query. Since then, various mechanisms have been proposed to enhance the accuracy of differentially private data release. Blum et al. [1] proved the possibility of non-interactive data release satisfying differential privacy for queries with polynomial VC-dimension, such as predicate queries. Dwork et al. [7] further proposed more efficient algorithms to release private sanitization of a data set with hardness results obtained.

Several recent works [4, 10, 11, 15–17] study the counting queries on multi-dimensional data, also referred to as histograms or contingency tables, where the multi-dimensional data can be indexed by a tree structure and each level

in the tree is an increasingly fine-grained summary/count. Cormode et al [4] propose the class of "private spatial decompositions" and conclude that the hybrid structure *kd-hybrid* provides an accurate yet efficient solution compared to alternatives. When applied to highly self-correlated time-series data, all the above methods, designed to perturb static data, become problematic because of highly compound Laplace perturbation error.

Rastogi and Nath [13] proposed a Discrete Fourier Transform (DFT) based algorithm which implements differential privacy by perturbing the discrete Fourier coefficients. However, this algorithm cannot provide real-time private release in a streaming environment. The recent works [3] [6] on continuous data streams defined the *event-level* privacy to protect an event, i.e. one user's presence at a particular time point, rather than the presence of a user. Our previous work [9] studies the problem of sharing single time-series with user-level differential privacy and we proposed an algorithm with filtering and adaptive sampling to improve the utility of the shared series.

6 Conclusion

We have proposed a real-time framework and two estimation algorithms to address the challenges of differentially private multi-dimensional time series release with application in traffic monitoring. The temporal estimation algorithm establishes a single time-series model for each cell in the space and performs posterior estimation to improve the utility of each released aggregate. The spatial estimation algorithm builds a spatial index by Quadtree and group similar cells together to overcome data sparsity. Domain knowledge is exploited by both estimation methods and is shown beneficial. We observe that the temporal estimation algorithm is highly accurate especially with sparse cells but requires modeling and slightly more running time. On the other hand, the spatial estimation algorithm relies on practical assumptions, demands less computation time, and provides better utility for dense cells and larger range queries. Compared to alternative methods, our solutions outperform the baseline LPA algorithm as well as state-of-the-art methods in both utility and computation efficiency. Future work may include in-depth study of complex spatial-temporal correlation between locations and timestamps.

Acknowledgments. This research is supported by NSF under grant CNS-1117763 and AFOSR under grant FA9550-12-1-0240.

References

1. Blum, A., Ligett, K., Roth, A.: A learning theory approach to non-interactive database privacy. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 609–618. ACM, New York (2008)
2. Brinkhoff, T.: A framework for generating network-based moving objects. Geoinformatica 6(2), 153–180 (2002)

3. Hubert Chan, T.-H., Shi, E., Song, D.: Private and continual release of statistics. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6199, pp. 405–417. Springer, Heidelberg (2010)
4. Cormode, G., Procopiuc, C., Srivastava, D., Shen, E., Yu, T.: Differentially private spatial decompositions. In: Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, pp. 20–31. IEEE Computer Society, Washington, DC (2012)
5. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
6. Dwork, C., Naor, M., Pitassi, T., Rothblum, G.N.: Differential privacy under continual observation. In: Proceedings of the 42nd ACM Symposium on Theory of Computing, pp. 715–724. ACM, New York (2010)
7. Dwork, C., Naor, M., Reingold, O., Rothblum, G.N., Vadhan, S.: On the complexity of differentially private data release: efficient algorithms and hardness results, pp. 381–390. ACM, New York (2009)
8. Fan, L., Xiong, L.: Adaptively sharing time-series with differential privacy. CoRR, abs/1202.3461 (2012)
9. Fan, L., Xiong, L.: Real-time aggregate monitoring with differential privacy. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, pp. 2169–2173. ACM, New York (2012)
10. Hay, M., Rastogi, V., Miklau, G., Suciu, D.: Boosting the accuracy of differentially private histograms through consistency. Proc. VLDB Endow. 3(1-2), 1021–1032 (2010)
11. Li, C., Miklau, G.: An adaptive mechanism for accurate query answering under differential privacy. Proc. VLDB Endow. 5(6), 514–525 (2012)
12. McSherry, F.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis, vol. 53, pp. 89–97. ACM, New York (2010)
13. Rastogi, V., Nath, S.: Differentially private aggregation of distributed time-series with transformation and encryption. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 735–746. ACM, New York (2010)
14. Xiao, X., Bender, G., Hay, M., Gehrke, J.: ireduct: differential privacy with reduced relative errors. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, pp. 229–240. ACM, New York (2011)
15. Xiao, X., Wang, G., Gehrke, J.: Differential privacy via wavelet transforms. IEEE Trans. on Knowl. and Data Eng. 23(8), 1200–1214 (2011)
16. Xiao, Y., Xiong, L., Yuan, C.: Differentially private data release through multidimensional partitioning. In: Jonker, W., Petković, M. (eds.) SDM 2010. LNCS, vol. 6358, pp. 150–168. Springer, Heidelberg (2010)
17. Xu, J., Zhang, Z., Xiao, X., Yang, Y., Yu, G.: Differentially private histogram publication. In: Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, pp. 32–43. IEEE Computer Society, Washington, DC (2012)

Policy Analysis for Administrative Role Based Access Control without Separate Administration

Ping Yang[1], Mikhail Gofman[2], and Zijiang Yang[3]

[1] Dept. of Computer Science, State University of New York at Binghamton, NY, USA
[2] Dept. of Computer Science, California State University at Fullerton, CA, USA
[3] School of Computer Science and Engineering, Xi'an University of Technology, China
Dept. of Computer Science, Western Michigan University, MI, USA

Abstract. Access control is widely used in large systems for restricting resource access to authorized users. In particular, role based access control (RBAC) is a generalized approach to access control and is well recognized for its many advantages in managing authorization policies.

This paper considers user-role reachability analysis of administrative role based access control (ARBAC), which defines administrative roles and specifies how members of each administrative role can change the RBAC policy. Most existing works on user-role reachability analysis assume the separate administration restriction in ARBAC policies. While this restriction greatly simplifies the user-role reachability analysis, it also limits the expressiveness and applicability of ARBAC. In this paper, we consider analysis of ARBAC without the separate administration restriction and present new techniques to reduce the number of ARBAC rules and users considered during analysis. We also present a number of parallel algorithms that speed up the analysis on multi-core systems. The experimental results show that our techniques significantly reduce the analysis time, making it practical to analyze ARBAC without separate administration.

1 Introduction

Access control is widely used for restricting resource access to authorized users. In particular, role based access control (RBAC) [2] is broadly recognized as a generalized approach to access control that has many advantages in performing authorization management. An RBAC policy is a tuple $\langle U, R, P, UA, PA\rangle$ where U, R and P are finite sets of users, roles, and permissions, respectively. $UA \subseteq U \times R$ represents the user-role assignment relation and $PA \subseteq P \times R$ represents the permission-role assignment relation. RBAC also supports role hierarchy: $r_1 \succeq r_2$ specifies that r_1 is senior to r_2 (or r_2 is junior to r_1), which implies that every member of r_1 is also a member of r_2, and every permission assigned to r_2 is also available to members of r_1.

Administrative role-based access control (ARBAC'97) [17] defines administrative roles and specifies how members of each administrative role can change the RBAC policy. ARBAC specifies *user-role administration* which controls the way changes are made to the user-role assignments. This control is enforced by two types of rules: (1) $can\_assign(r_a, c, r_t)$ that grants an administrative role r_a permission to assign a target role r_t to any user who satisfies the precondition c, and (2) $can\_revoke(r_a, r_t)$

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 49–64, 2013.

that grants an administrative role r_a permission to revoke a target role r_t from a user. The precondition c is a conjunction of literals, where each literal is either r (positive precondition) or $\neg r$ (negative precondition) for some role r. ARBAC'97 requires *separate administration* [22], *i.e.*, administrative roles cannot be target roles in *can_assign* and *can_revoke* rules or appear in the preconditions. In the rest of this paper, we represent the precondition c as $P \wedge \neg N$ where P contains all positive preconditions and N contains all negative preconditions in c.

The correctness of ARBAC policies is critical to system security because any design flaws and human specification errors in ARBAC may result in the leak of confidential data to unauthorized users. Large organizations may have large ARBAC policies. In such organizations, manual inspection of ARBAC policies for correctness can be impractical because actions performed by different administrators may interfere with each other in subtle ways. Thus, automated analysis algorithms are essential to ensure that an ARBAC policy conforms to the desirable correctness properties.

This paper considers the user-role reachability analysis of ARBAC [22], which asks "given an RBAC policy ϕ, an ARBAC policy ψ, a set of users U, a target user u_t, and a set of roles (called the "goal"), is it possible for users in $U \cup \{u_t\}$ to assign u_t to roles in the goal"? Since many security analysis problems, such as user-role availability [16], role containment [16], and weakest precondition [22], can be reduced to this problem, user-role reachability is crucial for ARBAC analysis.

Researchers have shown that user-role reachability analysis is intractable even under various restrictions on the ARBAC policy [16,18]. Most existing research on user-role reachability analysis [9,8,14] follows the definition of ARBAC'97 that assumes separate administration. By disallowing an administrative role to serve as the target role in any of the ARBAC rules, it is sufficient to consider the user-role assignments of only the target user. However, in practice, the separate administration restriction does not always hold. For example, a university ARBAC policy may specify that the role $DeptChair$ can assign a member of role $Faculty$ to role $AdmissionComittee$, which can in turn assign any user to role $Student$. Formally, this specification translates to the rules $can\_assign(DeptChair,\ Faculty,\ AdmissionComittee)$ and $can\_assign(AdmissionCommittee,\ true,\ Student)$, which do not satisfy the separate administration restriction.

Analysis of ARBAC without separate administration is significantly more challenging because we need to consider administrative actions that change the role memberships of all users, not only the target user. For example, a non-target user u may assign another non-target user u_1 to an administrative role, which can in turn change the role assignments of the target user. Stoller et al. [22] tackled this problem by developing an algorithm that is fixed parameter tractable with respect to the number of users and mixed roles. That is, the algorithm is exponential to the number of users and mixed roles, but is polynomial to the size of the policy when the number of users and mixed roles is fixed. However, since the number of users is usually large in large organizations, the algorithm does not scale well when analyzing ARBAC policies in such organizations. For example, we have applied this algorithm to analyze a university ARBAC policy containing 150 users and the program failed to terminate within 12 hours for 3 out of the 10 randomly generated queries.

Contributions: This paper presents a number of reduction techniques that improve the scalability of the algorithm in [22]. Our main contributions are summarized below.

- We propose two static reduction techniques – optimized slicing (Section 3.1) and hierarchical rule reduction (Section 3.4) – to reduce the number of ARBAC rules considered during analysis.
- We develop a user equivalent set reduction technique that reduces the number of users considered during analysis (Section 3.2).
- We propose a lazy reduction technique that delays performing unnecessary transitions (Section 3.3).
- We present several parallel algorithms, which speed up the analysis on multi-core or multi-processor platforms (Section 4).
- We evaluate the effectiveness of our reduction techniques and our parallel algorithms on an ARBAC policy representing a university administration. The experimental results show that our techniques significantly reduce the analysis time.

Organization: The rest of the paper is organized as follows. Section 2 describes the user-role reachability analysis algorithm for ARBAC without separate administration developed in [22]. Sections 3 and 4 present our reduction techniques and our parallel algorithms, respectively. The experimental results are given in Section 5, followed by a discussion of related research in Section 6. Section 7 concludes the paper.

2 Preliminaries: User-Role Reachability Analysis of ARBAC

User-role reachability analysis of ARBAC [22] asks: "given an RBAC policy ϕ, an ARBAC policy ψ, a set of users U, a target user u_t, and a set of roles (called the "goal"), is it possible for users in $U \cup \{u_t\}$ to assign u_t to all roles in the goal"? Let UA_0 be a set of all user-role assignments in ϕ. The user-role reachability analysis instance is represented as a tuple $I = \langle UA_0, u_t, \psi, goal \rangle$.

Stoller et al. [22] presented an algorithm for analyzing ARBAC without separate administration, which is formalized in Algorithm 1. The algorithm is fixed parameter tractable with respect to the number of users and mixed roles. A role is *negative* if it appears negatively in some precondition in the policy; other roles are *non-negative*. A role is *positive* if it appears in the goal, appears positively in some precondition in the policy, or is an administrative role; other roles are *non-positive*. A role that is both negative and positive is a *mixed* role. Note that their algorithm is applied to ARBAC without role hierarchy; ARBAC with role hierarchy can be converted to the corresponding non-hierarchical policy using the algorithm in [18]. Let $I = \langle UA_0, u_t, \psi, goal \rangle$ be a user-role reachability analysis problem instance. The algorithm works as follows.

First, the algorithm performs a slicing transformation (function $slicing$ in Line 3), which back-chains along the ARBAC rules to identify roles and rules relevant to the goal, and then eliminates the irrelevant ones. Function $slicing$ takes into account whether a role appears positively or negatively in the policy, and computes a set Rel_+ of positive roles and a set Rel_- of negative roles that are relevant to the goal. A set $RelRule$ of relevant rules is computed as a collection of all can_assign rules whose

Algorithm 1. The User-Role Reachability Analysis Algorithm in [22]

1: $Processed = Rel_+ = Rel_- = \emptyset$; $RelRule = \emptyset$;
2: **procedure** $analysis(UA_0, u_t, \psi, goal)$
3: $(Rel_+, Rel_-, RelRule) = slicing(UA_0, \psi, goal)$; $W = Reached = \{closure(UA_0)\}$;
4: **if** $goal \subseteq \{r \mid (u_t, r) \in closure(UA_0)\}$ **then** return true; **end if**
5: **while** $W \neq \emptyset$ **do**
6: remove a state s from W;
7: **for all** $can\_assign(r_a, P \wedge \neg N, r) \in RelRule$ **do**
8: **for all** (user $u \in U$) **do**
9: **if** ($r \in (Rel_+ \cap Rel_-)$, $(u, r) \notin s$, $P \subseteq \{r \mid (u, r) \in s\}$, $N \cap \{r \mid (u, r) \in s\} = \emptyset$, and $(u', r_a) \in s$ for some user u')
10: **then** $s' = \text{closure}(s \cup \{(u, r)\})$; add transition $s \overset{ua(r_a,u,r)}{\rightarrow} s'$ to G;
11: **if** $goal \subseteq \{r | (u_t, r) \in s'\}$ **then** return true; **end if**
12: **if** $s' \notin Reached$ **then** $W = W \cup \{s'\}$; $Reached = Reached \cup \{s'\}$ **end if**
13: **end if** **end for** **end for**
14: **for all** $(can\_revoke(r_a, r) \in RelRule)$
15: **for all** (user $u \in U$)
16: **if** ($(u, r) \in s$ and $(u', r_a) \in s$ for some user u')
17: **then** $s' = \text{closure}(s \setminus \{(u, r)\})$; add transition $s \overset{ur(r_a,u,r)}{\rightarrow} s'$ to G;
18: **if** $goal \subseteq \{r \mid (u_t, r) \in s'\}$ **then** return true; **end if**
19: **if** $s' \notin Reached$ **then** $W = W \cup \{s'\}$; $Reached = Reached \cup \{s'\}$ **end if**
20: **end if** **end for** **end for**
21: **end while**
22: return false;
23: **procedure** $slicing(UA_0, \psi, goal)$
24: **if** $goal = \emptyset$ **then** return $(\emptyset, \emptyset, \emptyset)$ **end if**
25: $Processed = Processed \cup goal$; $R_+ = goal$; $R_- = \emptyset$; $Rule = \emptyset$;
26: **for all** $can\_assign(r_a, P \wedge \neg N, r) \in \psi$ where $r \in goal$ **do**
27: $(R_1, R_2, R_3) = slicing(UA_0, \psi, (\{r_a\} \cup P) \setminus Processed)$; $R_+ = R_+ \cup R_1$;
28: $R_- = R_- \cup N \cup R_2$; $Rule = Rule \cup \{can\_assign(r_a, P \wedge N, r)\} \cup R_3$;
29: **end for**
30: $RelRev = \{can\_revoke(r_a, r) \in \psi \mid r \in R_-\}$; $Rule = Rule \cup RelRev$;
31: **for all** $can\_revoke(r_a, r) \in RelRev$ where $r_a \notin Processed$ **do**
32: $(R_4, R_5, R_6) = slicing(UA_0, \psi, \{r_a\})$; $R_+ = R_+ \cup R_4$;
33: $R_- = R_- \cup R_5$; $Rule = Rule \cup R_6$
34: **end for**
35: return $(R_+, R_-, Rule)$
36: **procedure** $closure(s)$
37: $s_1 = s$;
38: **for all** $can\_assign(r_a, P \wedge \neg N, r) \in RelRule$ **do**
39: **for all** user $u \in U$ **do**
40: **if** ($r \in (Rel_+ \setminus Rel_-)$, $(u, r) \notin s$, $P \subseteq \{r \mid (u, r) \in s\}$, $N \cap \{r \mid (u, r) \in s\} = \emptyset$, and $(u', r_a) \in s$ for some user u')
41: **then** $s_1 = s_1 \cup (u, r)$; **end if** **end for** **end for**
42: **if** $s == s_1$ **then** return s_1; **else** return $closure(s_1)$;

targets are in Rel_+ and all can_revoke rules whose targets are in Rel_-; only rules in $RelRule$ need to be applied during analysis.

Next, the algorithm constructs a *reduced transition graph* G using rules in $RelRule$. Each state in G is a set of user-role assignments and each transition describes an allowed change to the state defined by the ARBAC policy ψ. A transition is either $ua(r_a, u, r)$ which specifies that an administrative role r_a adds user u to role r, or $ur(r_a, u, r)$ which specifies that an administrative role r_a revokes user u from role r. The following reductions are applied: (1) Transitions that revoke non-negative roles (i.e., roles in $Rel_+ \setminus Rel_-$) or add non-positive roles (i.e., $Rel_- \setminus Rel+$) are prohibited because they do not enable any other transitions; (2) Transitions that add non-negative roles or revoke non-positive roles are *invisible*; such transitions will not disable any other transitions. Transitions that add or revoke mixed roles are *visible*. The invisible transitions together with a visible transition form a single composite transition.

The graph G is constructed as follows. First, the algorithm computes $\mathrm{closure}(UA_0)$, which is the largest state that is reachable from UA_0 by performing all invisible transitions enabled from UA_0 (function $closure$ in Line 3). The algorithm then computes a set of all states reachable from $\mathrm{closure}(UA_0)$ (Lines 5–21), and returns true iff there exists a state s in G such that $goal \subseteq \{r \mid (u_t, r) \in s\}$ (Lines 4, 11, and 18).

In [22], they have also identified a condition called the *hierarchical role assignment (HRA)*, under which analysis of ARBAC without separate administration can be reduced to analysis of ARBAC with separate administration. An ARBAC policy satisfies HRA if, for all $can\_assign(r_a, P \wedge \neg N, r)$ where r is an administrative role, $r_a \succeq r$.

Example 1. Consider the following ARBAC policy ψ and the reachability analysis problem for this policy with the initial RBAC policy $UA_0 = \{(u_1, r_1), (u_1, r_3), (u_2, r_2), (u_2, r_8), (u_3, r_2), (u_3, r_8), (u_t, r_6)\}$, the target user u_t, and the goal $\{r_5\}$.

1. $can\_assign(r_1, \{r_2\} \wedge \neg\emptyset, r_3)$
2. $can\_assign(r_6, \{r_4, r_3\} \wedge \neg\emptyset, r_5)$
3. $can\_assign(r_1, \{r_6\} \wedge \neg\{r_3\}, r_4)$
4. $can\_assign(r_2, \{r_8, r_1\} \wedge \neg\emptyset, r_6)$
5. $can\_assign(r_2, \{r_6\} \wedge \neg\emptyset, r_7)$
6. $can\_revoke(r_1, r_2)$
7. $can\_revoke(r_1, r_3)$
8. $can\_revoke(r_1, r_4)$

This policy does not satisfy the separate administration restriction, because role r_6 is both an administrative role in rule 2 and a target role in rule 4.

First, the algorithm performs slicing to compute a set Rel_+ of positive relevant roles and a set Rel_- of negative relevant roles as follows. Initially, Rel_+ contains all roles in the goal, i.e. r_5. Since the target role of rule 2 is r_5, the algorithm adds positive preconditions and administrative role of rule 2, i.e. r_4, r_3, and r_6, to Rel_+. The algorithm then processes rules 1 and 3, whose target roles are r_4 and r_3, respectively, adds their positive preconditions and administrative roles, i.e. r_2, r_6, and r_1, to Rel_+, and adds their negative preconditions, i.e. r_3, to Rel_-. Repeat this process until all roles in Rel_+ are processed, which results in $Rel_+ = \{r_1, r_2, r_3, r_4, r_5, r_6, r_8\}$ and $Rel_- = \{r_3\}$. The set of mixed roles is $Rel_+ \cap Rel_- = \{r_3\}$; other roles are both positive and non-negative. $RelRule$ contains rules 1, 2, 3, 4, and 7.

Next, the algorithm computes the initial state $closure(UA_0)$. Since rule 3 is enabled from UA_0 and r_4 is a non-negative role, (u_t, r_4) is added to UA_0 through an invisible transition. The algorithm then computes all states reachable from $closure(UA_0)$ using

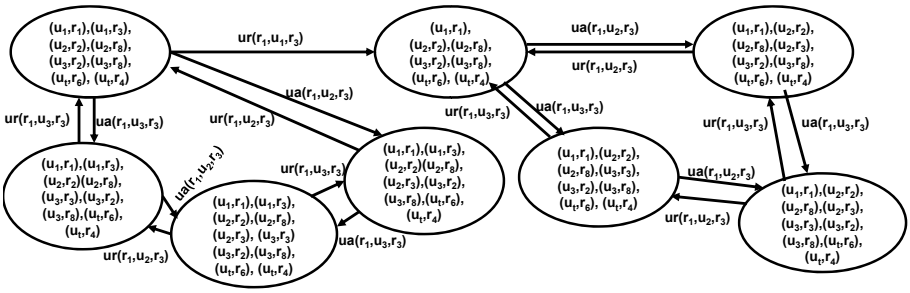


Fig. 1. Graph constructed in Example 1 using the algorithm in [22]

rules in $RelRule$. The resulting graph is given in Figure 1. Because the graph does not contain (u_t, r_5), the goal is not reachable. □

3 Reduction Techniques

The analysis algorithm described in Section 2, although simple, does not scale well for policies containing a large number of users. Let $I = \langle UA_0, u_t, \psi, goal \rangle$ be a user-role reachability analysis problem instance. In this section, we present a number of techniques for reducing the number of users and ARBAC rules considered during analysis.

3.1 Optimized Slicing

In this section, we present an approach to reduce the number of roles processed during slicing, and hence reduce the number of relevant rules computed.

We say that a role is irrevocable if there does not exist a can_revoke rule that revokes the role. For the target user u_t, we apply function $slicing$ defined in Algorithm 1 to perform slicing, except that Line 27 in the algorithm is replaced with the following:

$S = \{r \mid r \in (P \cup \{r_a\}) \wedge (r \text{ is nonnegative or irrevocable}) \wedge (u_t, r) \in UA_0\}$;
$(R_1, R_2, R_3) = slicing(UA_0, \psi, (\{r_a\} \cup P) \setminus S) \setminus Processed)$;

Similarly, Line 32 of Algorithm 1 is replaced with the following:

$S = \{r_a \mid (r_a \text{ is nonnegative or irrevocable}) \wedge (u_t, r_a) \in UA_0\}$;
$(R_4, R_5, R_6) = slicing(UA_0, \psi, (\{r_a\} \setminus S))$; $R_+ = R_+ \cup R_4$;

Basically, prior to slicing, we collect a set of nonnegative and irrevocable roles in the ARBAC policy. During slicing, we do not slice nonnegative or irrevocable roles assigned to the target user in the initial policy UA_0. This is safe because such roles will

Algorithm 2. An Optimized Slicing Algorithm for Non-Target Users

1: $Processed = \emptyset$;
2: **procedure** $optslicing(UA_0, \psi, goal)$
3: **if** $(goal == \emptyset)$ **then** return $(\emptyset, \emptyset, \emptyset)$; **end if**
4: $Processed = Processed \cup goal$; $R_+ = goal$; $R_- = \emptyset$; $Rule = \emptyset$;
5: **for all** $can\_assign(r_a, P \wedge \neg N, r)$ where $(u_t, r) \in goal$ **do**
6: **if** $((u, r_a) \in UA_0$ for some user u and (r_a is non-negative or irrevocable)) **then**
7: $R_1 = R_2 = R_3 = \emptyset$;
8: **else** $(R_1, R_2, R_3) = slicing(UA_0, \psi, \{r_a\} \setminus Processed)$; **end if**
9: $S = \{r \mid r \in P \wedge (r$ is non-negative or irrevocable$) \wedge (u_t, r) \in UA_0\}$;
10: $(R'_1, R'_2, R'_3) = optslicing(UA_0, \psi, (P \setminus S) \setminus Processed)$;
11: $R_+ = R_+ \cup S \cup R_1 \cup R'_1$; $R_- = R_- \cup N \cup R_2 \cup R'_2$; $Rule = Rule \cup R_3 \cup R'_3$;
12: **end for**
13: $RelRev = \{can\_revoke(r_a, r) \mid r \in R_-\}$; $Rule = Rule \cup RelRev$;
14: **for all** $can\_revoke(r_a, r) \in RelRev$ **do**
15: **if** $r_a \notin Processed \wedge (r_a$ is negative $\vee r_a$ is a non-negative role not assigned to any user in $UA_0)$ **then**
16: $(R_4, R_5, R_6) = slicing(\psi, \{r_a\})$;
17: $R_+ = R_+ \cup R_4$; $R_- = R_- \cup R_5$; $Rule = Rule \cup R_6$
18: **end if**
19: **end for**
20: return $(R_+, R_-, Rule)$;

not be revoked during the analysis and hence we do not need to reassign such roles to the target user. In addition, since a negative role may become non-negative after slicing, to further reduce the number of relevant rules computed, we perform slicing multiple times until the set of negative roles remains unchanged.

For non-target users, it is sufficient to apply only rules that assign such users to administrative roles, which have permission to assign the target user u_t to the goal. The pseudocode is given in Algorithm 2.

The reduction is given in Lines 6–10 and 15–16 of Algorithm 2. For every $can\_assign(r_a, P \wedge \neg N, r)$ where $r \in goal$, we check if r_a is a nonnegative or irrevocable role assigned to a user in UA_0. If so, we do not slice r_a; otherwise, we apply function $slicing$ defined in Algorithm 1 to slice r_a (Lines 6–8). This is different from Algorithm 1, in which r_a is always sliced. Next, we compute a set S of all nonnegative or irrevocable roles in P that are assigned to the target user in UA_0, and for every rule whose target role is in $P \setminus S$, we recursively call function $optslicing$ to slice the administrative roles of such rules (Lines 9–10). Note that we do not slice roles in P for non-target users, while Algorithm 1 does. Finally, for every $can\_revoke(r_a, r)$, if r_a is a nonnegative or irrevocable role assigned to some user in UA_0, we do not slice r_a (Lines 15–16).

Example: Consider the ARBAC policy and the query in Example 1. First, we compute a set of relevant roles and rules for the target user u_t using our optimized slicing mechanism. Since r_6 is a non-negative role assigned to u_t in UA_0, we do not slice r_6. Therefore, for the target user, $Rel_+ = \{r_1, r_2, r_3, r_4, r_5, r_6\}$, $Rel_- = \{r_3\}$, and

$RelRule = \{1, 2, 3, 7\}$. Next, we compute a set of relevant roles and rules for non-target users using our optimized slicing mechanism. Only administrative roles that have permissions to assign the target user to the goal, i.e., r_6 and r_1, need to be sliced. Since r_6 and r_1 are non-negative roles assigned to u_t and u_1 in UA_0, respectively, we do not slice these two roles. As a result, for non-target users, $Rel_+ = \{r_1, r_6\}$, $Rel_- = \{r_3\}$, and $RelRule = \emptyset$. This means that there is no need to assign roles to non-target users. The transition graph constructed with the optimized slicing contains only one state $\{(u_1, r_1), (u_1, r_3), (u_2, r_2), (u_2, r_8), (u_3, r_2), (u_3, r_8), (u_t, r_6), (u_t, r_4)\}$.

3.2 User Equivalent Set Reduction

In this section, we show that from each state it is sufficient to perform visible transitions for the target user and non-target users assigned distinct sets of roles. Our technique is based on a notion of user equivalent set defined below.

Definition 1. *The user equivalent set w.r.t a state s is defined as $ue(s) = \{(Uset_1, Rset_1), \ldots, (Uset_n, Rset_n)\}$ where $Rset_1 \neq \ldots \neq Rset_n$, $Uset_1 \cup \ldots \cup Uset_n = \{u | (u, r) \in s\}$, and for every $u \in Uset_i$, $Rset_i = \{r | (u, r) \in s\}$.*

The user equivalent set w.r.t a state s is basically an alternative representation of s, in which all users assigned the same set of roles are grouped together. Let G_{ue} be the transition graph constructed using the user equivalent set representation. There is a transition $ue(s) \xrightarrow{\alpha} ue(s')$ in G_{ue} if and only if there is a transition $s \xrightarrow{\alpha} s'$ in G. The goal is reachable in G_{ue} if and only if there exists a state $s_g \in G_{ue}$ and $(Uset, Rset) \in s_g$ such that $u_t \in Uset$, and $goal \subseteq Rset$.

Our *user equivalent set reduction* works as follows. For every state s and every $(Uset, Rset) \in s$, we compute only transitions for the target user and transitions for **one** randomly selected non-target user in $Uset$, if $Uset$ contains such users. This is different from Algorithm 1, which computes transitions for **all** users in $Uset$. Intuitively, the user equivalent set reduction is correct because transitions performed on all users in $Uset$ are the same, and transitions performed on one user in $Uset$ do not disable transitions performed on other users in $Uset$. We use G_{redue} to represent the transition graph constructed with the user equivalent set reduction.

The correctness of the reduction is formalized in Theorem 1. Given two states s_1 and s_2, we say that $s_1 \equiv s_2$ if there exists a substitution $\delta = \{u_1/u'_1, \ldots, u_n/u'_n\}$, where $u_1 \neq \ldots \neq u_n \neq u_t$ and $u'_1 \neq \ldots \neq u'_n \neq u_t$, such that $s_1\delta = s_2$. For example, $\{(\{u_1, u_t\}, \{r_1, r_2\}), (\{u_2\}, \{r_2\})\} \equiv \{(\{u_2, u_t\}, \{r_1, r_2\}), (\{u_1\}, \{r_2\})\}$ holds because there exists a substitution $\delta = \{u_1/u_2, u_2/u_1\}$ such that $\{(\{u_1, u_t\}, \{r_1, r_2\}), (\{u_2\}, \{r_2\})\}\delta = \{(\{u_2, u_t\}, \{r_1, r_2\}), (\{u_1\}, \{r_2\})\}$.

Theorem 1. *Let $I = \langle UA_0, u_t, \psi, goal\rangle$ be a user-role reachability analysis instance, and G_{redue} and G_{ue} be transition graphs constructed for I with and without using the user equivalent set reduction. The goal is reachable in G_{ue} iff the goal is reachable in G_{redue}.*

Example: Consider the user-role reachability analysis instance in Example 1. Since non-target users u_2 and u_3 are assigned the same set of roles in the initial state, we

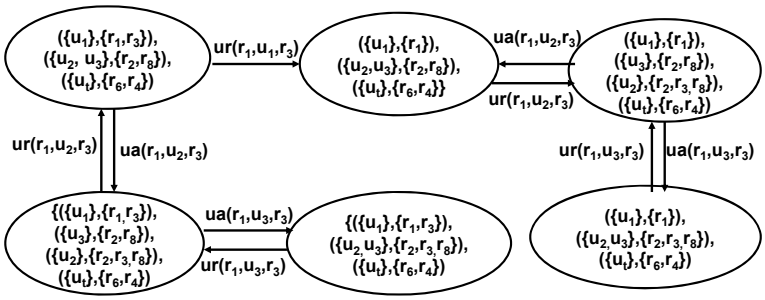


Fig. 2. The transition graph constructed with the user equivalent set reduction

need to perform only transitions for u_2 or u_3, but not both, from the initial state. This is different from Algorithm 1, which performs transitions for both u_2 and u_3 from the initial state. The graph constructed with the user equivalent set reduction is given in Figure 2, which contains 6 states and 9 transitions, i.e., 25% reduction on states and 47% on transitions.

Optimization: We can reduce the size of the state by replacing $Uset$ in $(Uset, Rset)$ with a pair $(counter, target)$, where $counter$ records the number of non-target users in $Uset$, and $target$ is either 1 ($u_t \in Uset$) or 0 ($u_t \notin Uset$).

3.3 Delayed Revocation

In this section, we propose to reduce the size of the transition graph by delaying transitions that can neither enable new transitions in s nor be disabled by any transitions. Formally, a transition $s \overset{ur(r_a,u,r)}{\rightarrow} s'$ is not performed from s (i.e. is delayed) if

1. $trans(s) = trans(s') \cup \{ur(r_a, u, r)\}$ where $trans(s)$ and $trans(s')$ are sets of all visible transitions enabled from s and s', respectively,
2. $s \setminus s' = \{(u, r)\}$,
3. r_a is non-negative or irrevocable.

Rules 1 and 2 specify that $s \overset{ur(r_a,u,r)}{\rightarrow} s'$ does not enable new visible and invisible transitions, respectively. Rule 3 specifies that $s \overset{ur(r_a,u,r)}{\rightarrow} s'$ cannot be disabled by other transitions.

Given a state s, we compute transitions that can be delayed in s as follows. First, we perform all ua transitions that assign users in s to roles. Next, for every ur transition that is enabled in s, we check if the transition enables any transition. If so, we perform the transition from s. Otherwise, we add the transition to a set $Delayed$. Since performing ur transitions may enable new ua and ur transitions, after all can_revoke rules are processed, we compute new transitions and check if any transitions in $Delayed$ enable other transitions. If so, such transitions are performed from s and are removed from $Delayed$. Repeat the above process until no new transitions are computed.

The correctness of the delayed revocation reduction is formalized in Theorem 2.

Theorem 2. *Let $I = \langle UA_0, u_t, \psi, goal\rangle$ be a user-role reachability analysis instance, $s_0 = closure(UA_0)$, and G_{dr} and G be transition graphs constructed for I with and without the delayed revocation reduction. The goal is reachable in G iff the goal is reachable in G_{dr}.*

Example: Consider the user-role reachability analysis instance in Example 1. Since transition $ur(r_1, u_1, r_3)$ does not enable new transitions from the initial state and r_1 is non-negative, with delayed revocation reduction, this transition is not performed from the initial state. The transition graph constructed contains 4 states and 8 transitions, i.e., 50% reduction on the number of states and 47% reduction on the number of transitions.

3.4 Hierarchical Rule Reduction

Hierarchical rule reduction avoids considering rules whose administrative preconditions are junior to non-negative or irrevocable administrative roles in UA_0. This is safe because senior roles inherit all administrative permissions of their junior roles, and non-negative/irrevocable roles are never revoked during analysis. This reduction does not reduce the size of the transition graph, but may reduce the analysis time since fewer rules are applied during analysis.

Consider the user-role reachability analysis problem instance in Example 1 and the role hierarchy $r_1 \succeq r_2$. The following three rules are added after the policy is transformed into the non-hierarchical one: $can\_assign(r_1, \{r_8, r_1\} \wedge \neg\emptyset, r6)$, $can\_assign(r_1, \{r_6\} \wedge \neg\emptyset, r_7)$, and $can\_assign(r_1, \{r_1\} \wedge \neg\emptyset, r_3)$. Since r_1 is a non-negative role, r_1 will never be revoked during analysis. As a result, rules 4 and 5 in Example 1 are not useful for reaching the goal (since administrative roles of these two rules are r_2, which is junior to r_1), and hence will not be applied during analysis.

4 Parallel Analysis Algorithm

Multi-core processors are becoming pervasive. In order for software applications to benefit from the continued exponential throughput advances in new computer systems, it is important to parallelize the applications. In this section, we extend Algorithm 1 to perform analysis in parallel. The pseudocode of our parallel algorithm is given in Algorithm 3.

First, we perform slicing to eliminate irrelevant roles, as we do in Algorithm 1. We then compute the initial state $init$ of the transition graph and add $init$ to a workset W (Line 5). Next, we create n threads $t_0, \ldots, t_n$ (Line 6; $||$ represents the concurrent execution of threads). Finally, each thread t_i removes one state from W, computes transitions enabled from the state using Lines 7 –10 and 14–17 of Algorithm 1, and adds the target states to W and the set of reachable state $Reached$ if the target states are not already in $Reached$ (Lines 11–20). Since multiple threads may access $Reached$ at the same time, $Reached$ needs to be protected by locks in order to ensure the correct execution of the program. Obviously, locking and unlocking $Reached$ every time a thread accesses $Reached$ imposes high overhead. To reduce

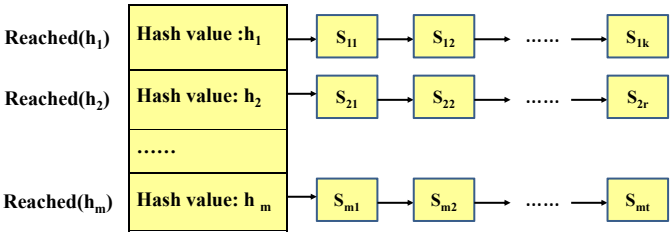


Fig. 3. Implementation of the set of reachable states $Reached$

Algorithm 3. User-Role Reachability Analysis Algorithm in [22].

1: $Reached = W = Rel_+ = Rel_- = \emptyset$; $RelRule = \emptyset$; $done = 0$;
2: **procedure** $mcanalysis(UA_0, u_t, \psi, goal)$
3: $(Rel_+, Rel_-, RelRule) = slicing(UA_0, \psi, goal)$; $init = closure(UA_0)$;
4: **if** $goal \subseteq \{r \mid (u_t, r) \in init\}$ **then** return true; **end if**
5: $W = Reached(h(init)) = \{init\}$;
6: $start(t_1) \mid\mid \ldots \mid\mid start(t_n)$;
7: **procedure** $start(t_i)$
8: **while** !done **do**
9: **if**($W == \emptyset$ and all threads are idle) **then** done = 1; **end if**
10: **while** ($W \neq \emptyset$)
11: $lock(W)$; remove a state s from W; $unlock(W)$;
12: **for all** transitions $s \overset{ua(r_a, u, r)}{\rightarrow} s'$
13: **if** $goal \subseteq \{r \mid (u_t, r) \in s'\}$ **then** return true; **end if**
14: $lock(Reached(h(s'))$;
15: **if** ($Reached(h(s'))$ does not exist)
16: $Reached(h(s')) = \{s'\}$; $unlock(Reached(h(s')))$;
17: $lock(W)$; $W = W \cup \{s'\}$; $unlock(W)$;
18: **else if** ($s' \notin Reached(h(s'))$)
19: $Reached(h(s')) = Reached(h(s')) \cup \{s'\}$; $unlock(Reached(h(s')))$;
20: $lock(W)$; $W = W \cup \{s'\}$; $unlock(W)$;
21: **else** $unlock(Reached(h(s')))$; **end if**
22: **end if** **end for** **end while**
23: **end while**
24: return false;

the time spent on waiting for locks to access $Reached$, we implemented $Reached$ as a hashtable shown in Figure 3. The hashtable is partitioned into multiple regions $Reached(h_1), \ldots, Reached(h_m)$; $Reached(h_i)$ stores a set of states whose hash values are h_i. Once a thread computes a transition $s \overset{\alpha}{\rightarrow} s'$, it computes the hash value $h(s')$ of s', locks $Reached(h(s'))$, adds s' to $Reached(h(s'))$ if s' is not already in $Reached(h(s'))$, and unlocks $Reached(h(s'))$. The above approach enables two threads to access two different regions in $Reached$ simultaneously. Our experimental results show that locking $Reached(h(s))$ instead of $Reached$ significantly improves

the performance. This is because threads access $Reached$ very frequently and checking if a state is in $Reached$ is relatively expensive. The algorithm terminates if the goal is reached, or if W is empty and all threads are not performing any computation.

It is also possible to reduce the time spent on waiting for locks to access the workset W by having each thread to have its own workset. Below, we present three approaches to minimizing (or completely removing) the number of operations performed on locking/unlocking W.

- **NoLock:** In this approach, each thread is not allowed to access other threads' worksets. Every time a thread computes a transition, it stores the target state in its own workset, if the target state is not already in $Reached$. This approach eliminates the requirement for locking, but may result in idle threads (due to empty workset).
- **FullLock:** In this approach, a thread is allowed to access other threads' workset to retrieve a state to process, if the thread's workset is empty. This approach ensures that all threads will be approximately equally busy, but it requires to lock the workset every time the workset is accessed.
- **PartialLock:** In this approach, whenever a thread t_i computes a new transition, it checks if thread $t_{(i-1) \ mod \ n}$ is idle. If so, it locks the workset of $t_{(i-1) \ mod \ n}$, adds the target state to the workset, unlocks the workset, and starts $t_{(i-1) \ mod \ n}$. The advantage of this approach is that locking is only needed when t_i adds a state to $t_{(i-1) \ mod \ n}$'s workset. This approach has limitation that each thread t_i has to frequently check if $t_{(i-1)modn}$ is sleeping.

Discussion: Two threads can safely access the same region in $Reached$ simultaneously if neither thread adds a state to or removes a state from the same region. Thus, in some cases, it may be possible to improve the performance by replacing mutual exclusion locks on $Reached$ with *reader-writer locks*. Unlike a mutual exclusion lock, which prevents all concurrent accesses to a critical region, a reader-writer lock allows multiple threads performing read operations to enter critical region. Our experiments, however, show that such optimization does not yield performance improvement (in fact, it often causes performance degradation). This is because multiple threads rarely access the same region in $Reached$ simultaneously during analysis, and reader-writer locks, due to their complexity, impose greater overheads than mutual exclusion locks.

5 Performance Results

This section evaluates the effectiveness of our reduction techniques and our parallel algorithms using the university ARBAC policy developed in [22] and the university RBAC policy developed in [7]. All reported data were obtained on a 2.4GHz 2 Quad-Core AMD Opteron Processor with 16GB RAM running Ubuntu 3.2.0.

The university RBAC and ARBAC policies contain 845 users, 32 roles, 329 can_assign rules, and 78 can_revoke rules, after being converted to the corresponding non-hierarchical policies. The policies include rules for assignment of users to various student and employee roles. Student roles include undergraduate student, graduate student, teaching assistant, research assistant, honors student, etc. Employee roles include president, provost, dean, department chair, faculty, honor program director, etc. A

Table 1. Performance of analysis algorithms without reduction, with a single reduction, and with all reductions

50 non-target users					
	NoReduct	OptSlice	DelayedRev	UserEquivSet	AllReduct
State	111	45	54	15	4
Transition	620	264	278	61	9
Time	0.97	0.41	0.57	0.13	0.09
75 non-target users					
	NoReduct	OptSlice	DelayedRev	UserEquivSet	AllReduct
State	24909	245	6393	273	5
Transition	214165	2168	42718	3222	10
Time	34.30	6.18	10.99	0.15	0.09
100 non-target users					
	NoReduct	OptSlice	DelayedRev	UserEquivSet	AllReduct
State	12706	7552	7855	39	6
Transition	145115	99520	107323	225	11
Time	2363	2029.26	2166.81	0.81	0.1

sample can_assign rule is: *the honors program director can assign an undergraduate student to the honors student role.* A sample user-role reachability problem instance is: *can a user who is a member of the department chair role and a user who is a member of the undergraduate student role assign the latter user to the honor student role?*

The university ARBAC policy does not satisfy the separate administration restriction. In addition, the policy has hierarchical role assignment w.r.t all administrative roles except those for assigning users to roles honor student and graduate student. This means that if the goal contains these two roles, then we cannot directly apply the algorithm for analyzing ARBAC with separate administration to carry out analysis. In our experiments, we randomly select one target user u_t, one role r, and n non-target users $\{u_1, \ldots, u_n\}$. We then apply analysis algorithms to check if users in $\{u_1, \ldots, u_n, u_t\}$ together can assign u_t to both honor student role and role r.

Effectiveness of Reduction Techniques. Table 1 gives the the size of the transition graph and the execution time for three sets of experiments with different numbers of randomly chosen non-target users (50, 75 or 100). Each data point reported in the table is an average over 8 randomly generated queries. The five columns represent reduction techniques applied during the experiments: with no reduction (NoReduct), with optimized slicing (OptSlice), with user equivalent set (UserEquivSet), with delayed revocation (DelayedRev), and with all reductions (AllReduct). Note that we do not include the hierarchical rule reduction in the table as it is not effective in our experiments. This is because all administrative roles in the university policy that have junior roles are mixed roles and remain mixed after applying all reductions.

We observe that, while all reduction techniques improve the performance, their effectiveness varies under different queries. UserEquivSet performs the best for all three sets of experiments and DelayedRev is the least effective. Integrating all reductions leads to a very effective solution. When the problem becomes difficult for the baseline algorithm to solve, AllReduct achieves an improvement of four orders of magnitude in execution time. In addition, when the number of non-target users is 150, NoReduct fails to complete 3 of the 8 queries within 12 hours, whereas the average analysis time of AllReduct is only 0.1 seconds.

Table 2. Performance of the parallel algorithm without reduction

50 non-target users									
		15 threads				30 threads			
	NoReduct	SharedWorkset	NoLock	FullLock	PartialLock	SharedWorkset	NoLock	FullLock	PartialLock
Time	0.97	0.33	0.40	0.32	0.52	0.32	0.43	0.34	0.45
75 non-target users									
		15 threads				30 threads			
	NoReduct	SharedWorkset	NoLock	FullLock	PartialLock	SharedWorkset	NoLock	FullLock	PartialLock
Time	34.30	6.58	6.60	5.85	6.74	5.82	7.04	5.80	6.54
100 non-target users									
		15 threads				30 threads			
	NoReduct	SharedWorkset	NoLock	FullLock	PartialLock	SharedWorkset	NoLock	FullLock	PartialLock
Time	2363	1059.73	517.09	436.87	513.46	776.34	537.68	407.53	531.83

Performance Results of Parallel Algorithms. Table 2 gives the execution time of four parallel analysis algorithms without reductions – SharedWorkset (Algorithm 3), NoLock, PartialLock, and FullLock – with 15 and 30 threads. The results show that, on average, FullLock performs the best, followed by PartialLock, NoLock, and SharedWorkset. FullLock and SharedWorkset with 30 threads outperform those with 15 threads, because the threads often wait for locks to access the worksets in FullLock and SharedWorkset, and hence more CPU cores are utilized with 30 threads than 15 threads. NoLock and PartialLock with 15 threads outperform those with 30 threads, because the threads do not or only occasionally wait for locks in NoLock and PartialLock, and hence the CPU cores are mostly utilized with 15 threads.

6 Related Work

A number of researchers have studied user-role reachability analysis of ARBAC. Schaad et al. [20] applied the Alloy analyzer [12] to check the separation of duty properties for ARBAC97; they did not consider preconditions for any operations. Li et al. [16] presented algorithms and complexity results for various analysis problems for two restricted versions of ARBAC97, called AATU and AAR; they did not consider negative preconditions. Jayaraman et al. [14] presented an abstraction refinement mechanism for detecting errors in ARBAC policies. Alberti et. al [1] developed a symbolic backward algorithm for analyzing Administrative Attribute-based RBAC policies, in which the policy and the query are encoded into a Bernays-Shonfinkel-Ramsey first order logic formulas. Becker [3] proposed a language DYNPAL for specifying dynamic authorization policies, which is more expressive than ARBAC, and presented techniques for analyzing DYNPAL. Sasturkar et al. [19] showed that user-role reachability analysis of ARBAC is PSPACE-complete, and presented algorithms and complexity results for ARBAC analysis subject to a variety of restrictions. Stoller et al. [21] presented algorithms for analyzing parameterized ARBAC. Gofman et al. [9] presented algorithms for analyzing evolving ARBAC. Uzun et al. [23] developed algorithms for analyzing temporal role-based access control models. However, none of the above works consider analysis of ARBAC without separate administration.

Several researchers have considered analysis of ARBAC without separate administration. Stoller et al. [22] provided fixed-parameter tractable algorithms for ARBAC with and without the separate administrative restriction. Their algorithm for analyzing

ARBAC without separate administration is exponential to the number of users in the policy, which is usually large in practice. Our work significantly improved the scalability of their algorithm by reducing the number of ARBAC rules and users considered during analysis. Ferrara et al [4] converted ARBAC policies to imperative programs and applied abstract-interpretation techniques to analyze the converted programs. However, if the goal is reachable, their approach cannot produce a trace which shows how the goal is reachable. Later, the same authors showed that if the goal is reachable in an ARBAC policy, then there exists a run of S with at most $|\text{administrative roles}| + 1$ users in which the goal is reachable [5]. However, their algorithm and reduction techniques are different from ours. Their techniques can be combined with ours to further reduce the analysis time. In addition, none of the above works present parallel analysis algorithms.

A number of researchers have considered analysis of fixed security policy [13,15,10,11], analysis of a single change to a fixed policy, or analysis of differences between two fixed policies [15,6]. However, none of them consider analysis of ARBAC.

7 Conclusion and Future Work

This paper considers the user-role reachability analysis without the separate administration restriction, which was shown to be PSPACE-complete in general. We present new analysis techniques with the goal of finding a practical solution to the problem. Our techniques focus on reducing the number of ARBAC rules and users considered during analysis and delaying unnecessary computations. We have also presented a number of parallel algorithms that speed up the analysis on multi-core systems. The experimental results on a university ARBAC policy show that our techniques significantly reduce the analysis time. In the future, we plan to develop symbolic analysis algorithms to implicitly search the state space with a potential to further improve the performance of the user-role reachability analysis.

Acknowledgement. This work was supported in part by NSF Grant CNS-0855204. We thank Kyoung-Don Kang for providing feedbacks on parallel algorithms and Dulcinea Chau for her contribution to the implementation of parallel algorithms.

References

1. Alberti, F., Armando, A., Ranise, S.: Efficient symbolic automated analysis of administrative attribute-based rbac-policies. In: ACM Symposium on Information, Computer and Communications Security, pp. 165–175 (2011)
2. A.N.S.I. (ANSI) Role-based access control. ANSI INCITS Standard 359-2004 (February 2004)
3. Becker, M.Y.: Specification and analysis of dynamic authorisation policies. In: 22nd IEEE Computer Security Foundations Symposium (CSF) (2009)
4. Ferrara, A.L., Madhusudan, P., Parlato, G.: Security analysis of role-based access control through program verification. In: Computer Security Foundations Symposium, pp. 113–125 (2012)
5. Ferrara, A.L., Madhusudan, P., Parlato, G.: Policy analysis for self-administrated role-based access control. In: Piterman, N., Smolka, S.A. (eds.) TACAS 2013. LNCS, vol. 7795, pp. 432–447. Springer, Heidelberg (2013)

6. Fisler, K., Krishnamurthi, S., Meyerovich, L.A., Tschantz, M.C.: Verification and change-impact analysis of access-control policies. In: International Conference on Software Engineering (ICSE), pp. 196–205 (2005)
7. Gofman, M., Luo, R., He, J., Zhang, Y., Yang, P.: Incremental information flow analysis of role based access control. In: International Conference on Security and Management, pp. 397–403 (2009)
8. Gofman, M.I., Luo, R., Solomon, A.C., Zhang, Y., Yang, P., Stoller, S.D.: RBAC-PAT: A policy analysis tool for role based access control. In: Kowalewski, S., Philippou, A. (eds.) TACAS 2009. LNCS, vol. 5505, pp. 46–49. Springer, Heidelberg (2009)
9. Gofman, M.I., Luo, R., Yang, P.: User-role reachability analysis of evolving administrative role based access control. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 455–471. Springer, Heidelberg (2010)
10. Guttman, J.D., Herzog, A.L., Ramsdell, J.D., Skorupka, C.W.: Verifying information flow goals in Security-Enhanced Linux. Journal of Computer Security 13(1), 115–134 (2005)
11. Irwin, K., Yu, T., Winsborough, W.H.: On the modeling and analysis of obligations. In: ACM Conference on Computer and Communications Security, pp. 134–143 (2006)
12. Jackson, D., Schechter, I., Shlyakhter, I.: Alcoa: the alloy constraint analyzer, pp. 730–733 (June 2000)
13. Jajodia, S., Samarati, P., Subrahmanian, V.S.: A logical language for expressing authorizations. In: Symposium on Security and Privacy, pp. 31–42 (1997)
14. Jayaraman, K., Ganesh, V., Tripunitara, M., Rinard, M., Chapin, S.: Automatic error finding for access control policies. In: Proceedings of 18th ACM Conference on Computer and Communications Security (CCS) (2011)
15. Jha, S., Reps, T.: Model-checking SPKI-SDSI. Journal of Computer Security 12, 317–353 (2004)
16. Li, N., Tripunitara, M.V.: Security analysis in role-based access control. ACM Transactions on Information and System Security 9(4), 391–420 (2006)
17. Sandhu, R., Bhamidipati, V., Munawer, Q.: The ARBAC97 model for role-based administration of roles. ACM Transactions on Information and Systems Security (TISSEC) 2(1), 105–135 (1999)
18. Sasturkar, A., Yang, P., Stoller, S.D., Ramakrishnan, C.: Policy analysis for administrative role based access control. In: IEEE Computer Security Foundations Workshop (2006)
19. Sasturkar, A., Yang, P., Stoller, S.D., Ramakrishnan, C.: Policy analysis for administrative role based access control. Theoretical Computer Science 412(44), 6208–6234 (2011)
20. Schaad, A., Moffett, J.D.: A lightweight approach to specification and analysis of role-based access control extensions. In: ACM Symposium on Access Control Models and Technologies, pp. 13–22 (2002)
21. Stoller, S.D., Yang, P., Gofman, M.I., Ramakrishnan, C.: Symbolic reachability analysis for parameterized administrative role-based access control. Journal of Computers & Security, 148–164 (2011)
22. Stoller, S.D., Yang, P., Ramakrishnan, C.R., Gofman, M.I.: Efficient policy analysis for administrative role based access control. In: 14th ACM Conference on Computer and Communications Security (CCS), pp. 445–455 (2007)
23. Uzun, E., Atluri, V., Sural, S., Vaidya, J., Parlato, G., Ferrara, A.L., Parthasarathy, M.: Analyzing temporal role based access control models. In: ACM Symposium on Access Control Models and Technologies, pp. 177–186 (2012)

Toward Mining of Temporal Roles

Barsha Mitra[1], Shamik Sural[1], Vijayalakshmi Atluri[2], and Jaideep Vaidya[3]

[1] School of Information Technology, IIT Kharagpur, India
[2] National Science Foundation and MSIS Department, Rutgers University, USA
[3] MSIS Department, Rutgers University, USA
{barsha.mitra,shamik}@sit.iitkgp.ernet.in,
atluri@rutgers.edu, jsvaidya@business.rutgers.edu

Abstract. In Role-Based Access Control (RBAC), users acquire permissions through their assigned roles. Role mining, the process of finding a set of roles from direct user-permission assignments, is essential for successful implementation of RBAC. In many organizations it is often required that users are given permissions that can vary with time. To handle such requirements, temporal extensions of RBAC like Temporal-RBAC (TRBAC) and Generalized Temporal Role-Based Access Control (GTRBAC) have been proposed. Existing role mining techniques, however, cannot be used to process the temporal element associated with roles in these models. In this paper, we propose a method for mining roles in the context of TRBAC. First we formally define the *Temporal Role Mining Problem* (TRMP), and then show that the TRMP problem is NP-complete and present a heuristic approach for solving it.

Keywords: TRBAC, Role Enabling Base, Temporal role mining, NP-complete, Greedy heuristic.

1 Introduction

Role-Based Access Control (RBAC) [14] has emerged as the *de-facto* standard for enforcing authorized access to data and resources. *Roles* play a pivotal part in the working of RBAC. In order to implement RBAC, one of the key challenges is to identify a correct set of roles. The process of defining the set of roles is known as *Role Engineering* [5]. It can be of two types: *top-down* and *bottom-up*. The top-down approach [13] analyzes and decomposes the business processes into smaller units in order to identify the permissions required to carry out the specific tasks. The bottom-up [15] approach uses the existing user-permission assignments to determine the roles. Role mining is a bottom-up role engineering technique. It assumes that the user-permission assignments are available in the form a boolean matrix called the UPA matrix. A 1 in cell (i, j) of the UPA denotes that user i is assigned permission j. Role mining takes the UPA matrix as input and produces as output a set of roles, a UA matrix representing which roles are assigned to each user and a PA matrix representing which permissions are included in each role.

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 65–80, 2013.

In many organizations, there is a need for restricting permissions to users only for a specified period of time. In such cases, the available user-permission assignments have temporal information associated with them. The roles that are derived from these temporal user-permission assignments will also have limited temporal duration. The traditional RBAC model is incapable of supporting such temporal constraints associated with roles. In order to capture this temporal aspect of roles, several extensions of RBAC have been proposed like Temporal-RBAC (TRBAC) [1] and Generalized Temporal Role-Based Access Control (GTRBAC) [8]. The TRBAC model supports periodic enabling and disabling of roles. This implies that a role can be enabled during a certain set of time intervals and remains disabled for the rest of the time. The set of time intervals during which each role can be enabled is specified in a *Role Enabling Base* (REB). To the best of our knowledge, none of the existing role mining techniques take into consideration such temporal information while computing the set of roles, and hence cannot be applied for mining roles in TRBAC or GTRBAC models.

In this paper, we propose an approach for role mining in the context of the TRBAC model. The problem of finding an optimal and correct set of roles from an existing set of temporal user-permission assignments has been named as the *Temporal Role Mining Problem* (TRMP), and the process of finding such a set is termed as *Temporal Role Mining*. We first formally define TRMP and analyze its complexity. We then propose an approach for solving TRMP that works in two phases: i) enumerating a candidate set of roles and ii) selecting a minimal set of roles using a greedy heuristic from the candidate role set and assigning them to the appropriate users so that each user gets his required set of permissions for only the set of time intervals specified in the original temporal user-permission assignments. Our experimental results show how the number of the final set of roles obtained using our approach varies with the number of permissions and also the number of distinct time intervals present.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents some preliminaries related to RBAC and TRBAC. Section 4 defines the problem and analyzes its complexity. Section 5 describes our heuristic approach to solve TRMP. In Section 6, we present experimental results and finally conclude in Section 7 along with directions for future work.

2 Related Work

The problem of finding an optimal set of roles from a given set of user-permission assignments is known as the *Role Mining Problem* (RMP). In [15] and [16], the authors formally define RMP and show the problem to be NP-complete. They also map RMP to the Minimum Tiling Problem of databases and use a known heuristic algorithm for finding the minimum tiling of a database to solve RMP.

Vaidya et al. [17] present an unsupervised approach called *RoleMiner* which is based on clustering users having similar permissions into groups. The authors present two algorithms: *CompleteMiner* and *FastMiner* to enumerate the set

of candidate roles. Lu et al. [9] model the problem of *Optimal Boolean Matrix Decomposition* (OBMD) using binary integer programming. This enables them to directly apply a wide range of heuristics for binary integer programming to solve the OBMD problem. Since solving RMP essentially involves optimally decomposing the boolean matrix UPA into two boolean matrices, UA and PA, RMP is modeled as a boolean matrix decomposition problem. In [6], it is shown that the role minimization problem is equivalent to the Minimum Biclique Cover (MBC) problem. MBC being an NP-hard problem, the authors present a greedy heuristic to find the minimum biclique cover of a bipartite graph which can be used to solve RMP. In [2], [3] the authors propose a three-step methodology to reduce the complexity of role mining as well as the administration cost by restricting the process of role mining only to stable user-permission assignments, i.e., user-permission assignments that belong to roles having weight above a predefined threshold value. The unstable assignments are used to create single-permission roles.

Other role mining approaches include role mining with noisy data [12], where the input data is first cleansed to remove the noise before generating candidate roles, role mining based on weights [10] in which a certain weight is associated with each permission depending on its importance, mining roles having low structural complexity and semantic meaning [11], and Visual Role Mining (VRM) [4], which enumerates roles based on a visual analysis of the graphical representation of the user-permission assignments. Xu and Stoller [19] propose algorithms for role mining which optimize a number of policy quality metrics. Verde et al. present an approach in [18] to make role mining applicable to large datasets and hence scalable.

None of the above-mentioned approaches consider the presence of temporal elements in user-permission assignments. We study the problem of role mining in the context of a temporal extension of RBAC, namely, TRBAC [1].

3 Preliminaries

In this section, we present some preliminaries related to RBAC and TRBAC.

3.1 Role-Based Access Control

According to the NIST standard [7], the RBAC model consists of the following components:

Definition 1. *RBAC*

- $USERS$, $ROLES$, OPS, *and* $OBJS$ *are respectively the set of users, roles, operations, and objects.*
- $UA \subseteq USERS \times ROLES$, *a many-to-many mapping of user-to-role assignment.*
- *The set of permissions,* $PRMS$.
  $PRMS \subseteq \{(op, obj) | op \in OPS \wedge obj \in OBJS\}$.

- $PA \subseteq ROLES \times PRMS$, *a many-to-many mapping of role-to-permission assignment.*
- $assigned\_users(R) = \{u \in USERS | (u, R) \in UA\}$, *the mapping of role* $R \in ROLES$ *onto a set of users.*
- $assigned\_permissions(R) = \{p \in PRMS | (p, R) \in PA\}$, *the mapping of role* $R \in ROLES$ *onto a set of permissions.*

3.2 Temporal Role-Based Access Control

The TRBAC model allows periodic enabling and disabling of roles. Temporal dependencies among such actions are expressed using role triggers. The enabling or disabling of roles is expressed using simple event expressions or prioritized event expressions. Role status expressions, having the form *enabled* R or $\neg$*enabled* R, describe whether a role R is currently enabled or not. Event expressions, role status expressions and role triggers together build up the *Role Enabling Base* (REB), which contains various temporal constraints related to the enabling and disabling of roles. The model also allows runtime requests to be issued by an administrator to dynamically change the status of a role, so as to be able to react to emergency situations.

In order to represent the set of time intervals for which a role can be enabled, the TRBAC model uses the notion of *periodic expressions* [1]. Periodic time can be represented as $\langle [begin, end], P \rangle$, where P is a periodic expression representing an infinite set of periodic time instants and $[begin, end]$ is a time interval that imposes an upper and a lower bound on the instants of P. The representation of periodic expression is based on the notion of *Calender*. A calender is a finite set of consecutive time intervals. Let $C_d, C_1, ..., C_n$ be a set of calenders. A periodic expression P is defined as:

$$P = \sum_{i=1}^{n} O_i \cdot C_i \triangleright r \cdot C_d \tag{1}$$

where $O_1 = all, O_i \in 2^{\mathbb{N}} \cup \{all\}, C_i \sqsubseteq C_{i-1}$ for $i = 2, ..., n, C_d \sqsubseteq C_n$, and $r \in \mathbb{N}$. The symbol $\sqsubseteq$ denotes sub-calender relationship. The first part of the periodic expression P before the symbol $\triangleright$ represents the starting points of the set of time intervals denoted by P and the second part denotes the duration of each time interval in terms of a natural number r and calender C_d.

4 Mining Roles Having Temporal Constraints

In this section, we discuss how the user-permission assignments having associated temporal information can be represented, then formally define the temporal role mining problem, and finally present an analysis of its complexity.

4.1 Temporal UPA Matrix

The temporal role mining process takes as input a temporal user-permission assignment relation, which describes the sets of time intervals for which one or more permissions are assigned to each user. Such a user-permission assignment relation can be directly available in an organization or can be derived from the access logs. We represent these temporal user-permission assignments using a *Temporal UPA* (TUPA) matrix. The rows of the matrix represent the users and the columns represent the permissions. Each cell (u_i, p_j) of the matrix contains either a zero or a set T_{ij} of time intervals for which user u_i is assigned permission p_j. Each set of time intervals T_{ij} is represented using a *periodic expression* of the form of Eqn. 1. Table 1 shows an example TUPA matrix.

Table 1. An Example TUPA Matrix

	p_1	p_2	p_3
u_1	$\langle[1/1/2010, \infty], all.Days+ \{8\}.Hours \triangleright 1.Hours\rangle$, $\langle[1/1/2010, \infty], all.Days+ \{10\}.Hours \triangleright 1.Hours\rangle$	0	$\langle[1/1/2010, \infty], all.Days+ \{8\}.Hours \triangleright 1.Hours\rangle$
u_2	0	$\langle[1/1/2010, \infty], all.Days+ \{6\}.Hours \triangleright 1.Hours\rangle$, $\langle[1/1/2010, \infty], all.Days+ \{8\}.Hours \triangleright 2.Hours\rangle$	$\langle[1/1/2010, \infty], all.Days+ \{8\}.Hours \triangleright 1.Hours\rangle$
u_3	0	$\langle[1/1/2010, \infty], all.Days+ \{9\}.Hours \triangleright 1.Hours\rangle$	0

In this matrix, user u_1 is assigned permission p_1 for two different sets of time intervals: everyday from 8 am to 9 am and from 10 am to 11 am. u_1 is also assigned p_3 for a single set of time intervals: everyday from 8 am to 9 am. Similarly, u_2 is assigned p_2 everyday from 6 am to 7 am and from 8 am to 10 am. u_2 is also assigned p_3 everyday from 8 am to 9 am. Finally, u_3 is assigned only p_2 for a single set of time intervals: everyday from 9 am to 10 am. It may be observed that, in the TUPA matrix, a user can be assigned different permissions for the same or different sets of time intervals and also the same permission can be assigned to different users for the same or different sets of time intervals. In general, the sets of time intervals that are not equal can be either overlapping or disjoint.

4.2 Problem Definition

The TUPA matrix is given as input to the temporal role mining process. The output of the process is a set of roles ROLES, a UA matrix, which is a boolean matrix denoting the roles assigned to each user, a PA matrix, which is a boolean matrix denoting the permissions included in each role and a Role Enabling Base (REB), containing, for each role, a set of time intervals during which the role can be enabled. The permissions are thus available to the users through the assigned roles only during the sets of time intervals during which the corresponding roles are enabled as specified in the REB.

We say that the output is *consistent* with the input TUPA matrix if each user, on being assigned a subset of the set of mined roles, gets a set of permissions for a set of time intervals as he was originally assigned in the given TUPA matrix. Thus, the *Temporal Role Mining Problem* (TRMP) can be formally defined as:

Definition 2. *TRMP*
Given a set of users USERS, a set of permissions PRMS and a temporal user-permission assignment TUPA, find a set of roles ROLES, a user-role assignment UA, a role-permission assignment PA, and an REB consistent with the TUPA, such that the total number of roles is minimized.

4.3 Complexity Analysis

In this subsection, we provide a formal analysis of the complexity of TRMP. Before proceeding with the formal analysis, we first formulate the decision version of TRMP. The Decision-TRMP problem (DTRMP) can be defined as:

Definition 3. *DTRMP*
Given a set of users USERS, a set of permissions PRMS, a temporal user-permission assignment TUPA and a positive integer k, is there a set of roles ROLES, a user-role assignment UA, a role-permission assignment PA and an REB, consistent with the TUPA, such that $|ROLES| \leq k$ *?*

Given a certificate consisting of a set ROLES, a UA, a PA and an REB, it can be verified in polynomial time whether |ROLES|$\leq k$ and whether the ROLES, UA, PA and REB are consistent with the TUPA by finding out the set of time intervals during which each user gets a particular permission through one or more roles assigned to him and comparing with the TUPA. Thus DTRMP is in NP.

Next we show that a known NP-complete (or NP-hard) problem is polynomial time reducible to DTRMP. For this, we select RMP. The Decision RMP, which has been shown to be NP-complete [15], can be stated as:

Definition 4. *Decision RMP*
Given a set of users U_{RMP}*, a set of permissions* P_{RMP}*, a user-permission assignment UPA and a positive integer k, is there a set of roles R, a user-role assignment* UA_{RMP} *and a role-permission assignment* PA_{RMP} *consistent with the UPA, such that* $|R| \leq k$ *?*

Given an instance of the Decision RMP, U_{RMP} and P_{RMP} are respectively mapped to USERS and PRMS using identity transformations. UPA is mapped to TUPA where each zero entry of UPA is mapped to a zero entry of TUPA and each non-zero entry of UPA is assigned a fixed set of time intervals, say T_0 in the TUPA. This reduction is in polynomial time.

To complete the proof, it is to be shown that the output instance of Decision RMP (consisting of R, UA_{RMP} and PA_{RMP}) is such that |R | is less than or equal to k if and only if the output instance of DTRMP (consisting of ROLES, UA, PA and REB) is such that |ROLES| has a value less than or equal to k.

Given an instance of the Decision RMP, let R, UA_{RMP} and PA_{RMP} constitute the output instance such that $|\text{R}| \leq k$. Now, the output instance of DTRMP can be constructed from the output instance of the Decision RMP as follows: R denotes the set of roles ROLES, UA_{RMP} denotes the UA and PA_{RMP} denotes the PA. The REB is constructed by associating the same set of time intervals corresponding to each of the roles in ROLES. Since the set of time intervals during which each role in ROLES can be enabled are the same, so if the output instance of the Decision RMP is consistent with the given UPA, then the output instance of the DTRMP is also consistent with the given TUPA. Therefore, the output instance of DTRMP constructed from the output instance of Decision RMP is a valid solution of DTRMP. Similarly, it can be shown that given an output instance of DTRMP, a valid solution to Decision RMP can be constructed. Thus, DTRMP produces as output a set of roles ROLES having size k or less, a UA, a PA and an REB if and only if Decision RMP gives a set of roles R of size k or less, a UA_{RMP} and a PA_{RMP}. Therefore, DTRMP is NP-complete.

5 Heuristic Approach for Solving TRMP

Since TRMP has been shown to be NP-complete in Section 4, we present a heuristic approach for solving it in this section. It works in two phases:

- *Candidate Role Generation*: This phase enumerates the set of candidate roles from an input TUPA matrix.
- *Role Selection*: This phase selects the least possible number of roles from the candidate roles using a greedy heuristic so that the generated UA, PA and REB together is consistent with the TUPA matrix.

5.1 Candidate Role Generation

A TUPA matrix is given as input to the candidate role generation phase. Each non-zero entry of TUPA represents a triple $\langle u_i, p_j, T_{ij} \rangle$. This implies that user u_i is assigned permission p_j for the set of time intervals T_{ij}. Let us denote the set of all such triples by U_T. A role is a collection of permissions which is enabled during a certain set of time intervals and can be assigned to a specific set of users. So, each triple of U_T can be considered as a role consisting of a single user, a single permission and a set of time intervals. We call such roles as unit roles. In the first phase, a set *UnitRoles* of all such unit roles is initially constructed.

Before going into the details of the successive steps, we show how the creation of roles depends on the interrelationships among the sets of time intervals for which permissions are assigned to users. Let a user u_1 be assigned two permissions, namely, p_1 for the set of time intervals T_{11} and p_2 for the set of time intervals T_{12}. Now, three scenarios may arise.

- If $T_{11} = T_{12}$, then a role r will be created containing p_1 and p_2. r will be enabled during T_{11}.

- If $T_{11} \cap T_{12} = \phi$, i.e., T_{11} and T_{12} are disjoint, then two roles: r_1 containing p_1, and r_2 containing p_2 will be created. r_1 will be enabled during T_{11} and r_2 will be enabled during T_{12}.
- If $T_{11} \cap T_{12} \neq \phi$, i.e., T_{11} and T_{12} have a non-empty intersection, then three roles will be created: r_1 containing p_1 and p_2 which will be enabled during $T_{11} \cap T_{12}$, r_2 containing only p_1 which will be enabled during $T_{11} - (T_{11} \cap T_{12})$, and r_3 containing only p_2 which will be enabled during $T_{12} - (T_{11} \cap T_{12})$. Either one of r_2 and r_3 might be superfluous depending on whether $T_{11} \subset T_{12}$ or $T_{12} \subset T_{11}$.

Thus, it is seen that depending upon how the various sets of time intervals are related to one another, they may have to be split differently while creating roles. Moreover, since $(T_{11} \cap T_{12}) \subseteq T_{11}$, $(T_{11} \cap T_{12}) \subseteq T_{12}$ and each of T_{11} and T_{12} is a subset of itself, the set of time intervals during which a role can be enabled is a subset of the common sets of time intervals associated with the permissions included in that role. Since the sets of time intervals associated with the permissions are the non-zero TUPA matrix entries, the set of time intervals during which a role can be enabled can be considered to be a subset of the non-zero TUPA matrix entries. Therefore, each role can be considered to be a subset of one or more triples of U_T.

After constructing the set of unit roles, a set of initial roles is next constructed as follows: For each user, if a common set of time intervals exists among all the permissions assigned to him, then the user, the permissions and the common set of time intervals are put together to form a role. If after creation of this role, corresponding to any one or more permissions, there remains any set of time intervals that is not included in the role, separate roles are created for each of those permissions by including the corresponding user and the remaining set of time intervals. If no common set of time intervals exists among all the permissions assigned to the user, then separate roles are created by combining the user, each of the permissions and the corresponding sets of time intervals. We call the set of all the initial roles as $InitialRoles$.

In the final step of phase 1, a generated set of roles is constructed by performing pairwise intersection between the members of $InitialRoles$. We call this set as $GeneratedRoles$. For any two roles i and j of $InitialRoles$, let u be the user associated with i and u' be the user associated with j. If both i and j have one or more common permissions and the associated sets of time intervals have a non-empty intersection, then the following roles are created:

- a role r_1 containing the users u and u', the permissions common to both i and j, and the common set of time intervals between i and j.
- a role r_2 containing user u, the permissions common to both i and j, and the remaining set of time intervals (if any, after creating r_1) associated with role i.
- a role r_3 containing user u', the permissions common to both i and j, and the remaining set of time intervals (if any, after creating r_1) associated with role j.

- a role r_4 containing user u, permissions of i which are not present in j, and the set of time intervals associated with i.
- a role r_5 containing user u', permissions of j which are not present in i, and the set of time intervals associated with j.

Roles r_2 and r_3 are created by combining the sets of time intervals that get split as a result of creating role r_1 with the appropriate users and permissions. After creating the generated set of roles, the candidate set of roles $CandidateRoles$ is created by taking the union of $UnitRoles$, $InitialRoles$ and $GeneratedRoles$. If the sets of time intervals associated with any two roles in $CandidateRoles$ are the same and either their permission sets are identical or the user sets associated with them are identical, then the two roles are merged to create a single role by either taking union of their user sets or that of their permission sets respectively. This completes phase 1.

5.2 Role Selection

The set of candidate roles created in the Candidate Role Generation phase is given as input to the Role Selection phase. In this phase, a minimal cardinality subset of $CandidateRoles$ is selected so that each user after being assigned a subset of the set of selected roles, gets each of the permissions assigned to him for only those sets of time intervals as specified in the TUPA matrix. As already mentioned, each role can be considered to be a subset of the set U_T. A set of such roles can be considered to be a set of subsets of U_T. We say that a role r covers a triple $\langle u_i, p_j, T_{ij} \rangle$ of U_T if r contains the user u_i, permission p_j and either a proper or an improper subset of the set of time intervals T_{ij}. Each role covers one or more triples of the set U_T and each triple of U_T can be covered by more than one role (if a set of time intervals corresponding to a non-zero TUPA entry gets split into two or more sets of intervals during role creation). So, there arises a need to distinguish between fully covered and partially covered triples.

Definition 5. *Fully Covered, Partially Covered*
Let T_{im} be a set of time intervals corresponding to the triple $t = \langle u_i, p_m, T_{im} \rangle$ and r_k be a role such that $r_k = (\{u_i, u_j\}, \{p_m, p_n\}, \{T_k\})$. If $T_{im} \subseteq T_k$, then r_k fully covers the triple t. If $T_k \subset T_{im}$, then r_k partially covers t.

The task of the role selection phase is to select the minimum number of roles to fully cover all the triples of U_T. It uses the following greedy heuristic: at each stage, the role that fully covers the maximum number of uncovered triples is selected. If more than one role fully covers the maximum number of triples, the tie is broken by selecting the role that partially covers the maximum number of triples. This is done because the sets of time intervals remaining after a number of sets of time intervals get partially covered may be covered by a single role if the corresponding user and permission sets are the same. At each stage, after selecting a role, the fully covered triples of U_T are marked appropriately and the partially covered ones are updated.

After all the triples are fully covered, it is checked whether any two or more of the selected roles can be merged. If so, the appropriate roles are merged into a single role. In the merging step at the end of the candidate role generation phase, the merged roles are not compared with each other for further merging. It may so happen that two or more roles created as a result of merging at the end of the previous phase can be merged further to form a single role. If these roles are selected by the role selection phase, then they are merged in this final merging step. If not, then no merging is required. This final merging step can further reduce the number of roles.

Algorithm 1. Enumerate Candidate Roles

Require: $P(u)$: the set of permissions assigned to user u
Require: $U(i)$: the set of users associated with role i
Require: $T(P(u))$: the common set of time intervals among all the permissions assigned to user u
Require: T_{up}: the set of time intervals for which user u is assigned permission p
Require: $T(x)$: the set of time intervals during which role x can be enabled
1: Initialize $UnitRoles$, $InitialRoles$, $GeneratedRoles$, $CandidateRoles$ as empty sets
2: **for** each triple $(\{u\}, \{p\}, \{T_{up}\})$ corresponding to a non-zero entry of TUPA **do**
3: $UnitRoles \leftarrow UnitRoles \cup (\{u\}, \{p\}, \{T_{up}\})$
4: **end for**
5: **for** each user $u \in TUPA$ **do**
6: **if** $T(P(u)) \neq \phi$ **then**
7: $InitialRoles \leftarrow InitialRoles \cup (\{u\}, \{P(u)\}, \{T(P(u))\})$
8: **for** each $p \in P(u)$ **do**
9: $InitialRoles \leftarrow InitialRoles \cup (\{u\}, \{p\}, \{T_{up} - T(P(u))\})$
10: **end for**
11: **else**
12: $InitialRoles \leftarrow InitialRoles \cup (\{u\}, \{p\}, \{T_{up}\})$
13: **end if**
14: **end for**
15: **for** each role $i \in InitialRoles$ **do**
16: $InitialRoles \leftarrow InitialRoles - i$
17: **for** each role $j \in InitialRoles$ **do**
18: **if** $\{i \cap j\} \neq \phi$ and $T(i) \cap T(j) \neq \phi$ **then**
19: ▷ $\{i \cap j\}$ denotes the common set of permissions of roles i and j
20: $GeneratedRoles \leftarrow GeneratedRoles \cup (\{u, u'\}, \{i \cap j\}, \{T(i) \cap T(j)\}) \cup (\{u\}, \{i \cap j\}, \{T(i) - (T(i) \cap T(j))\}) \cup (\{u'\}, \{i \cap j\}, \{T(j) - (T(i) \cap T(j))\}) \cup (\{u\}, \{i - (i \cap j)\}, \{T(i)\}) \cup (\{u'\}, \{j - (i \cap j)\}, \{T(j)\})$ ▷ u and u' are the users associated with roles i and j respectively
21: **end if**
22: **end for**
23: **end for**
24: $CandidateRoles \leftarrow UnitRoles \cup InitialRoles \cup GeneratedRoles$
25: **for** each $i, j \in CandidateRoles$ **do**
26: **if** $T(i) = T(j)$ **then**
27: **if** $i = j$ **then** ▷ permission sets of i and j are same
28: $CandidateRoles \leftarrow \{CandidateRoles - i - j\} \cup (\{U(i) \cup U(j)\}, \{i\}, \{T(i)\})$
29: **else**
30: **if** $U(i) = U(j)$ **then**
31: $CandidateRoles \leftarrow \{CandidateRoles - i - j\} \cup (\{U(i)\}, \{i \cup j\}, \{T(i)\})$
32: **end if**
33: **end if**
34: **end if**
35: **end for**

The algorithm for enumerating the set of candidate roles is given in Algorithm 1. The sets $UnitRoles, InitialRoles, GeneratedRoles$ and $CandidateRoles$ are initialized as empty sets in line 1. The set $UnitRoles$ is created from the set of triples corresponding to the non-zero entries of the TUPA matrix in lines 2 - 4.

Lines 5 - 14 create the set of initial roles. The set of generated roles is created in lines 15 - 23 by performing pairwise intersection between the members of $InitialRoles$. The set $CandidateRoles$ is created in line 24 by taking union of $UnitRoles$, $InitialRoles$ and $GeneratedRoles$. Finally, for each candidate role, it is checked whether it can be merged with any other candidate role, and if possible, the two roles are merged (lines 25 - 35).

The algorithm for selecting the minimal set of roles from $CandidateRoles$ is shown in Algorithm 2. *Select Final Roles* takes as input the set $CandidateRoles$. It keeps track of the number of triples of U_T that remain uncovered, using $entry\_count$. $entry\_count$ is initialized to the number of triples of U_T (line 1). The *while loop* of lines 2 - 9 selects a role that fully covers the maximum number of uncovered triples in each iteration (line 3) until all the triples are fully covered. If there is a tie, the role that partially covers the maximum number of uncovered triples is selected (line 5). U_T is updated in line 7 by marking appropriately the triples which get fully covered and by modifying the ones which get partially covered. $entry\_count$ is next updated (line 8). Finally, when no uncovered triples are left the set of selected roles is checked to determine if any of the roles can be merged. If so, the appropriate roles are merged (line 10). Lastly, the UA, PA and REB are constructed from the set of selected roles (line 11).

Algorithm 2. Select Final Roles

Require: $entry\_count$: the number of uncovered triples of U_T
Require: $fully\_covered$: the number of triples that are fully covered in a particular iteration
1: $entry\_count \leftarrow |U_T|$
2: **while** $entry\_count > 0$ **do**
3: select a role that fully covers the maximum number uncovered triples
4: **if** there is a tie **then**
5: select the role that partially covers the maximum number of triples
6: **end if**
7: update U_T
8: $entry\_count \leftarrow entry\_count - fully\_covered$
9: **end while**
10: merge roles, if possible, in the set of selected roles
11: create UA, PA and REB from the set of selected roles

5.3 Illustrative Example

We illustrate how our approach works using the TUPA matrix given in Table 2 which is a simplified representation of Table 1. In this table, each non-zero TUPA matrix entry is a set of one or two time intervals. The proposed approach is, however, generic enough to handle complex sets of time intervals represented in the form of periodic expressions as mentioned in Section 3.

A. Candidate Role Generation
Algorithm 1 constructs the set $UnitRoles$ as follows.

Table 2. Simplified Representation of the TUPA Matrix given in Table 1

	p_1	p_2	p_3
u_1	8 am - 9 am 10 am - 11 am	0	8 am - 9 am
u_2	0	6 am - 7 am 8 am - 10 am	8 am - 9 am
u_3	0	9 am - 10 am	0

$UnitRoles = \{r_1 = (\{u_1\}, \{p_1\}, \{8\ am - 9\ am\}), r_2 = (\{u_1\}, \{p_1\}, \{10\ am - 11\ am\}), r_3 = (\{u_1\}, \{p_3\}, \{8\ am{-}9\ am\}), r_4 = (\{u_2\}, \{p_2\}, \{6\ am{-}7\ am\}), r_5 = (\{u_2\}, \{p_2\}, \{8\ am{-}10\ am\}), r_6 = (\{u_2\}, \{p_3\}, \{8\ am{-}9\ am\}), r_7 = (\{u_3\}, \{p_2\}, \{9\ am - 10\ am\})\}$

The set *InitialRoles* is next constructed by considering each user of the TUPA matrix one at a time.

$InitialRoles = \{r_8 = (\{u_1\}, \{p_1, p_3\}, \{8\ am - 9\ am\}), r_9 = (\{u_1\}, \{p_1\}, \{10\ am{-}11\ am\}), r_{10} = (\{u_1\}, \{p_3\}, \{8\ am{-}9\ am\}), r_{11} = (\{u_2\}, \{p_2\}, \{6\ am{-}7\ am\}), r_{12} = (\{u_2\}, \{p_2, p_3\}, \{8\ am{-}9\ am\}), r_{13} = (\{u_2\}, \{p_2\}, \{9\ am{-}10\ am\}), r_{14} = (\{u_2\}, \{p_3\}, \{8\ am - 9\ am\}), r_{15} = (\{u_3\}, \{p_2\}, \{9\ am - 10\ am\})\}$

By performing pairwise intersection between the members of *InitialRoles*, the set of generated roles is obtained.

$GeneratedRoles = \{r_{16} = (\{u_1, u_2\}, \{p_3\}, \{8\ am{-}9\ am\}), r_{17} = (\{u_1\}, \{p_1\}, \{8\ am{-}9\ am\}), r_{18} = (\{u_2\}, \{p_2\}, \{8\ am{-}9\ am\}), r_{19} = (\{u_2, u_3\}, \{p_2\}, \{9\ am{-}10am\})\}$

Finally, after taking union of the three sets of roles created, merging the roles and renaming them, the set *CandidateRoles* is obtained.

$CandidateRoles = \{r_1 = (\{u_1\}, \{p_1, p_3\}, \{8\ am - 9\ am\}), r_2 = (\{u_1\}, \{p_1\}, \{10\ am{-}11\ am\}), r_3 = (\{u_2\}, \{p_2\}, \{6\ am{-}7\ am\}), r_4 = (\{u_2\}, \{p_2, p_3\}, \{8\ am{-}9\ am\}), r_5 = (\{u_1, u_2\}, \{p_3\}, \{8\ am - 9\ am\}), r_6 = (\{u_2, u_3\}, \{p_2\}, \{9\ am - 10\ am\}), r_7 = (\{u_2\}, \{p_2\}, \{8\ am - 10\ am\})\}$

B. Role Selection

The set *CandidateRoles* is given as input to Algorithm 2. In the first iteration, both r_1 and r_5 fully cover the maximum number of uncovered triples, i.e., 2, and neither of them partially covers any of the triples. This tie is broken by selecting r_1. As a result, triples $\langle u_1, p_1, \{8am{-}9am\}\rangle$ and $\langle u_1, p_3, \{8am{-}9am\}\rangle$ are fully covered. In the next iteration, there is a tie among all the remaining candidate roles as each of them fully covers 1 triple. Among these roles, only r_4 and r_6 each covers 1 triple partially. This tie is broken by selecting r_4. Now the triple $\langle u_2, p_3, \{8am{-}9am\}\rangle$ gets fully covered and the triple $\langle u_2, p_2, \{8am{-}10am\}\rangle$ after getting partially covered becomes $\langle u_2, p_2, \{9am - 10am\}\rangle$. Each of

the remaining triples is fully covered by selecting the roles r_6, r_2 and r_3 one by one. After sorting the roles according to their indices and renaming r_6 to r_5, the resulting UA, PA and REB are shown in Tables 3, 4 and 5, respectively.

Table 3. UA Matrix

	r_1	r_2	r_3	r_4	r_5
u_1	1	1	0	0	0
u_2	0	0	1	1	1
u_3	0	0	0	0	1

Table 4. PA Matrix

	p_1	p_2	p_3
r_1	1	0	1
r_2	1	0	0
r_3	0	1	0
r_4	0	1	1
r_5	0	1	0

Table 5. REB

Role	Enabling Time Interval
r_1	$all.Days + \{8\}.Hours \triangleright 1.Hours$
r_2	$all.Days + \{10\}.Hours \triangleright 1.Hours$
r_3	$all.Days + \{6\}.Hours \triangleright 1.Hours$
r_4	$all.Days + \{8\}.Hours \triangleright 1.Hours$
r_5	$all.Days + \{9\}.Hours \triangleright 1.Hours$

6 Experimental Results

We test the performance of the proposed temporal role mining algorithm on a number of synthetically generated TUPA matrices. Instead of directly creating random TUPA matrices, we first create UA, PA and REB randomly and then combine them to obtain the random TUPA matrix. For all the datasets, the number of users is fixed at 100. The REB is created by varying the number of distinct time intervals from 1 to 3. When the number of distinct time intervals is 2, we consider three cases that may arise: (i) one time interval is contained in the other (2C) (ii) the two time intervals overlap, but neither is contained in the other (2O) and (iii) the two time intervals are disjoint (2D). When the number of distinct time intervals is 3, we consider 5 scenarios: (i) two intervals overlap, neither one is contained in the other and the third one is disjoint (2O1D) (ii) all the three intervals are disjoint (3D) (iii) one interval is contained in the other and the third is disjoint (2C1D) (iv) two intervals are contained in the third one (3C) and (v) all the three intervals overlap, but no interval is contained in any of the remaining intervals (3O). For generating the data, the number of roles is taken to be one-tenth of of the number of permissions, except in two cases: (i) when the number of permissions is 10, the number of roles is taken as 2 for both one and two distinct time intervals and (ii) when the number of distinct time intervals is three and the number of permissions is 10 or 20, the number of roles is taken as 3. For achieving high confidence level, we created 20 datasets for each parameter setting and the final number of roles reported is the mean and mode of the output of all the 20 runs.

Table 6 shows the variation of the number of roles with the number of permissions when there is only one distinct time interval for all the user-permission assignments. In this scenario, the temporal role mining problem reduces to non-temporal role mining. The results of Table 6 indicate correctness of our approach since the number of roles obtained in each case is the same as the number of roles with which the dataset was generated.

Table 7 shows the variation of the number of roles with the number of permissions when the number of distinct time intervals is two. This table shows that as the number of permissions increases, the number of roles also increases. The

Table 6. No. of Roles (Mean|Mode) vs. No. of Permissions when the No. of Distinct Time Intervals is 1

Number of Permissions	Number of Roles (Mean|Mode)
10	2.0 |2
20	2.0 |2
30	3.0 |3
40	4.0 |4

Table 7. No. of Roles (Mean|Mode) vs. No. of Permissions when the No. of Distinct Time Intervals is 2

	Number of Roles (Mean|Mode)		
Number of Permissions	2C	2O	2D
10	2.7 |2	2.8 |2	2.8 |2
20	2.9 |2	3.2 |2	2.7 |2
30	5.5 |3	8.4 |12	5.6 |7
40	7.4 |7	21.8 |23	9.9 |12

increase in the number of roles is attributed to the splitting of time intervals during role creation. The increase is relatively less for case 2C because, if a user is assigned a permission for both the time intervals, then actually he is assigned the permission for a single time interval, namely, the one which contains the other. But still the number of roles generated is more than that obtained for one distinct time interval, because a single user can acquire different permissions during either one of the two distinct time intervals, resulting in the splitting of the time intervals during role creation. Case 2O generates a large number of roles as the two time intervals get split during role creation. Finally, for case 2D, the time intervals do not get split during role creation and so the number of roles is comparatively less than case 2O.

Finally, in Table 8, we show the variation of the number of roles with the number of permissions for all the five cases mentioned above for three distinct time intervals. Here also it is seen that with the increase in the number of permissions, the number of roles increases. Case 3O generates the maximum number of roles as overlap among three time intervals results in the maximum amount of time interval splitting. Cases 3C and 3O generate more number of roles than cases 2C and 2O respectively, thus showing that with the increase in the number of distinct time intervals, the number of roles generated also increases. Case 2O1D generates lesser number of roles than case 3O, as overlap between two time intervals results in less splitting than that occurring in case of overlap among three time intervals. The number of roles obtained in case 3D is less than the rest of the cases, since these 4 cases result in time interval splitting during role creation which is completely absent in case 3D. Cases 2O1D and 3O respectively generate more number of roles than cases 2C1D and 3C as overlap among time intervals causes greater amount of splitting than containment of time intervals within one another.

Our results show that, as the number of distinct time intervals increases and there exists some overlap among them, the number of roles finally produced also increases due to the splitting of time intervals during role creation. The effect of overlap is more significant than that of permissions.

Table 8. No. of Roles (Mean|Mode) vs. No. of Permissions when the No. of Distinct Time Intervals is 3

Number of Permissions	Number of Roles (Mean|Mode)									
	2O1D		3D		2C1D		3C		3O	
10	6.2	7	5.6	6	5.7	3	6.6	6	7.8	8
20	7.4	6	5.2	6	5.9	7	7.0	6	7.7	3
30	7.3	7	6.5	7	6.7	7	6.3	6	9.9	15
40	13.3	12	9.0	8	11.6	10	8.5	4	19.9	23

7 Conclusions and Future Directions

Temporal role mining is essential for creating roles in systems that assign permissions to users for varying sets of time intervals. In this paper, we have formally defined the *Temporal Role Mining Problem* (TRMP) and proved it to be NP-complete. We have proposed an approach for mining roles from temporal user-permission assignments. Our approach first creates a candidate set of roles and then selects a minimal subset of the candidate role set using a greedy heuristic to cover all the requisite assignments.

Future work in this area would include designing of other heuristics that can further reduce the number of roles finally obtained. Different optimization metrics besides the number of roles may be defined that would generate more meaningful roles having temporal constraints. An approximate solution approach could be designed that would allow a certain amount of inaccuracy in terms of the time duration for which some users would acquire certain permissions through one or more roles assigned to him.

Acknowledgement. This work is partially supported by the National Science Foundation under grant numbers CNS-0746943 and 1018414.

References

1. Bertino, E., Bonatti, P.A., Ferrari, E.: TRBAC: A temporal role-based access control model. ACM Transactions on Information and System Security 4(3), 191–233 (2001)
2. Colantonio, A., Di Pietro, R., Ocello, A., Verde, N.V.: Mining stable roles in RBAC. In: Gritzalis, D., Lopez, J. (eds.) SEC 2009. IFIP AICT, vol. 297, pp. 259–269. Springer, Heidelberg (2009)
3. Colantonio, A., Pietro, R.D., Ocello, A., Verde, N.V.: Taming role mining complexity in RBAC. Computers and Security Special Issue on Challenges for Security and Privacy and Trust 29(5), 548–564 (2010)
4. Colantonio, A., Pietro, R.D., Ocello, A., Verde, N.V.: Visual role mining: A picture is worth a thousand roles. IEEE Transactions on Knowledge and Data Engineering 24(6), 1120–1133 (2012)
5. Coyne, E.J.: Role engineering. In: Proceedings of 1st ACM Workshop on Role Based Access Control, pp. 15–16 (November 1995)

6. Ene, A., Horne, W., Milosavljevic, N., Rao, P., Schreiber, R., Tarjan, R.E.: Fast exact and heuristic methods for role minimization problems. In: Proceedings of 13th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 1–10 (June 2008)
7. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security (TISSEC) 4(3), 224–274 (2001)
8. Joshi, J., Bertino, E., Latif, U., Ghafoor, A.: A generalized temporal role-based access control model. IEEE Transactions on Knowledge and Data Engineering 17(1), 4–23 (2005)
9. Lu, H., Vaidya, J., Atluri, V.: Optimal boolean matrix decomposition: Application to role engineering. In: Proceedings of 24th IEEE International Conference on Data Engineering (ICDE), pp. 297–306 (April 2008)
10. Ma, X., Li, R., Lu, Z.: Role mining based on weights. In: Proceedings of 15th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 65–74 (June 2010)
11. Molloy, I., Chen, H., Li, T., Wang, Q., Li, N., Bertino, E., Calo, S., Lobo, J.: Mining roles with multiple objectives. ACM Transactions on Information and System Security (TISSEC) 13(4), 36:1–36:35 (2010)
12. Molloy, I., Li, N., Lobo, J., Dickens, L.: Mining roles with noisy data. In: Proceedings of 15th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 45–54 (June 2010)
13. Roeckle, H., Schimpf, G., Weidinger, R.: Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization. In: Proceedings of 5th ACM Workshop on Role-Based Access Control, pp. 103–110 (July 2000)
14. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)
15. Vaidya, J., Atluri, V., Guo, Q.: The role mining problem: Finding a minimal descriptive set of roles. In: Proceedings of 12th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 175–184 (June 2007)
16. Vaidya, J., Atluri, V., Guo, Q.: The role mining problem: A formal perspective. ACM Transactions on Information and System Security (TISSEC) 13(3), 27:1–27:31 (2010)
17. Vaidya, J., Atluri, V., Warner, J.: Role miner: Mining roles using subset enumeration. In: Proceedings of 13th ACM Conference on Computer and Communications Security (CCS), pp. 144–153 (October 2006)
18. Verde, N.V., Vaidya, J., Atluri, V., Colantonio, A.: Role engineering: From theory to practice. In: Proceedings of 2nd ACM Conference on Data and Application Security and Privacy (CODASPY), pp. 181–191 (February 2012)
19. Xu, Z., Stoller, S.D.: Algorithms for mining meaningful roles. In: Proceedings of 17th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 57–66 (June 2012)

Towards User-Oriented RBAC Model

Haibing Lu[1], Yuan Hong[2], Yanjiang Yang[3], Lian Duan[4], and Nazia Badar[2]

[1] Santa Clara University
[2] Rutgers University
[3] I2R Singapore
[4] New Jersey Institute of Technology
hlu@scu.edu, yhong@cimic.rutgers.edu, yyang@i2r.a-star.edu.sg,
lian.duan@njit.edu, nbadar@pegasus.rutgers.edu

Abstract. Role mining recently has attracted much attention from the role-based access control (RBAC) research community as it provides a machine-operated means of discovering roles from existing permission assignments. While there is a rich body of literature on role mining, we find that user experience/perception - one ultimate goal for any information system - is surprisingly ignored by the existing works. This work is the first to study role mining from the end-user perspective. Specifically, based on the observation that end-users prefer simple role assignments, we propose to incorporate to the role mining process a user-role assignment sparseness constraint that mandates the maximum number of roles each user can have. Under this rationale, we formulate user-oriented role mining as two specific problems: one is user-oriented exact role mining problem (RMP), which is obliged to completely reconstruct the given permission assignments, and the other is user-oriented approximate RMP, which tolerates a certain amount of deviation from the complete reconstruction. The extra sparseness constraint poses a great challenge to role mining, which in general is already a hard problem. We examine some typical existing role mining methods to see their applicability to our problems. In light of their insufficiency, we present a new algorithm, which is based on a novel dynamic candidate role generation strategy, tailored to our problems. Experiments on benchmark datasets demonstrate the effectiveness of our proposed algorithm.

Keywords: Access Control, Role Mining, Sparseness, Binary, Optimization.

1 Introduction

Role-based access control (RBAC) restricts system access to authorized users by assigning permissions to roles and then assigning roles to users. RBAC has become a de facto access control model, due to its many advantages, including the convenience of authorization allocation and the reduction of the system administrative workload. Enterprises still employing their old access control systems want to migrate to RBAC. To accomplish the migration, the first phase is to define a good role set. While the role defining problem is seemingly straightforward, it has been recognized as one of the costliest phases in the implementation of RBAC and poses a great challenge to the system engineers. The difficulty comes from the fact that a RBAC system engineer usually has little knowledge on the semantic meanings of user responsibilities and business processes within an enterprise.

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 81–96, 2013.

Role mining has proven to be an effective (machine-operated) means of discovering a good role set. Its key idea is to utilize data mining technologies to extract patterns from existing permission assignments of the old access control system, which are then used to establish roles. This greatly facilitates the implementation of RBAC (by migrating from the old access control system). In the literature, role mining has been extensively studied. In a nutshell, the existing literature investigates role mining with different objectives, including minimization of the number of roles, minimization of the administration cost, minimization of the complexity of the role hierarchy structure, and others. However, we find that none of the existing works has ever considered to improve end-user experience (of the underlying RBAC system), which should be one ultimate goal for any practical information system. Needless to say, users' experience/perception of a system represents system usability and directly affects the eventual success of the system in practice. As such, we argue that user-friendliness should be an essential criterion for evaluating the quality of role mining.

In this paper, we study user-oriented role mining, being the first to explore the role mining problem from the end-user perspective. Our daily experiences tell us that end users often prefer fewer role assignments; as long as a user acquires all the needed permissions, the fewer roles she has to bear, the better usability she may feel upon the system. That is, from the end-user perspective, a good RBAC system should have as sparse user-role assignments as possible. This coincides with an advantage of RBAC: recall that one reason accounting for the wide acceptance of RBAC is that it allows users to carry very few roles while enjoying their (potentially many) access rights. However, on the flip side, if we create a unique role for every user, in which case user-role assignments are trivially the most sparse, then the resultant RBAC system would contain too many roles. This absolutely contradicts to a premise of RBAC, which is to map permission roles with functional roles within an organization. As such, a user-oriented RBAC solution should not compromise other advantages of RBAC. To this end, we propose to limit the maximum number of roles a user can take on top of regular role mining. Such a strategy would well balance user friendliness and other system quality factors such as administrative overhead. While the idea is clear, the added constraint poses extra challenges to role mining, considering that role mining in general has already been a hard problem. Towards tackling the obstacle, we make the following contributions: (1) we formulate user-oriented role mining as two specific problems, i.e., user-oriented exact RMP (Role Mining Problem) and user-oriented approximate RMP; (2) in searching for efficient solutions to the formulated problems, we examine several typical role mining algorithms and reveal that they do not meet our needs; (3) in view of the weaknesses of the existing algorithms, we present an efficient algorithm, tailored to the user-oriented role mining problems; (4) to investigate the effectiveness of our algorithm, we conduct experiments on benchmark datasets and obtain promising experimental results.

The remainder of the paper is organized as follows. Section 2 reviews existing role mining works in the literature. Section 3 presents the user-oriented role mining problem. Section 4 presents optimization models. The heuristic algorithm is provided in Section 5. Experimental results on benchmark access control data sets are reported in Section 6. Section 7 concludes the paper.

2 Related Work

The concept of role engineering was introduced in 1995 by Coyne [1]. It aims to define an architectural structure that is complete, correct and efficient to specify the organization's security policies and business functions. Coyne's approach is a top-down process oriented strategy for role definition. With the top-down approach, one starts from requirements and successively refine the definitions to reflect the business functions [13].

Top-down approaches are only suitable for small size enterprises. For medium or large size cases, the bottom-up approach that utilizes the existing user-permission assignments to formulate roles is preferred. In particular, data mining techniques are often employed in the bottom-up approach to identify promising roles. Kuhlmann et al. [5] coined the concept of role mining using data mining. In [14], an algorithm ORCA is proposed to build a hierarchy of permission clusters using data mining technologies. However, overlap between roles is not allowed in ORCA, which contradicts to normal practice in real applications. Vaidya et al. [18] propose a subset enumeration approach, which can effectively overcome this limitation.

An inherent issue with all of the above approaches is that there is no formal notion for goodness of role. Vaidya et al. [15] propose to use the number of roles to evaluate the goodness of a role set. They [16] also introduce to use the administrative task as an evaluative criterion. Role hierarchy is another important evaluative criterion, as it is closely related to the semantic meanings of roles. Related works on role hierarchy include [10]. Ma et al. [9] also use weights to combine multiple objectives. Our work in this paper strengthens this line of research by being the first to incorporate user experience/perception as an extra evaluative criterion in role mining.

Beside the above works on finding a role set with respect to different criteria, there are other interesting works with different flavors. Lu et al. [6,8] present an optimization framework for role engineering. They even extend their work to incorporate negative authorizations in [7]. Frank et al. [3] provide a probabilistic model to analyze the relevance of different kinds of business information for defining roles that can both explain the given user-permission assignments and describe the meanings from the business perspective. They [4] also introduce a model to take the attributes of users into account. Studies on role mining in the presence of noisy data are presented in [17].

3 Conflict Resolution

In this section, we study and formulate user-oriented role mining. As we have discussed in the introduction, from users' perspective a user-friendly RBAC system should assign as few roles as possible to each user; no user is happy with being overwhelmed by assuming too many roles (titles). Ideally, a user would wish to carry only one role given that the role provides with him all the necessary access privileges for him to work and function smoothly. Indeed, in reality most organization's systems are designed that way. For example, in a school system, the majority of people carry only one role among STUDENT, FACULTY, STAFF, and VISITOR. In a software company, most employees are either ACCOUNTANT, ENGINEER, or MANAGER. Thus, user-oriented role mining is characterized with the fact that user-permission assignments should be sparse. This gives rise to the definition of user-oriented role mining, as stated below.

Definition 1. *(**User-Oriented Role Mining**) User-Oriented Role Mining is to discover a RBAC solution with respect to some evaluation criterions, such that after role assignments users get the same permissions as before and the resultant user-role assignments are sparse.*

To concretize user-oriented role mining, we have two questions to answer: (1) what evaluation criterion should be used to evaluate the goodness of a RBAC solution? (2) what level of spareness of user-role assignments is appropriate? Towards answering these questions, we examine existing role evaluation criteria for some insights.

3.1 Existing Evaluation Criteria

Role mining is typically formulated as certain optimization problems with objectives and constraints. As summarized by Molloy et al. at [10], there are five main factors which can be used to evaluate the goodness of a RBAC solution. They are the number of roles $|R|$, the complexity of user-role assignments $|UA|$, the complexity of role-permission assignments $|PA|$, the number of direct user-permission assignments $|DUPA|$, and the complexity of reduced role hierarchy $|t\_reduce(RH)|$. Among them, $|R|$, $|UA|$, and $|PA|$ are routine notations in RBAC, and no further exposition is needed on them. Direct user-permission assignments $DUPA$ imply that roles of one single user are treated as special roles. $|DUPA|$ is the amount of such direct user-permission assignments in a deployed RBAC solution. Role hierarchy $RH \subseteq R \times R$ represents the partial order over roles R. $t\_reduce(RH)$ denotes the transitive reduction of the role hierarchy.

Almost all role mining evaluative criteria can be generally described with the weighted structural complexity measure introduced in [10], which sums up the above five factors, with possibly different weights for each factor.

Definition 2. *(**Weighted Structural Complexity Measure**) Given a weight vector $W =< w_r, w_u, w_p, w_d, w_h >$, where $w_r, w_u, w_p, w_d, w_h \in \mathcal{Q}^+ \cup \{0\}$[1], the weighted structural complexity of an RBAC state γ, which is denoted as $wsc(\gamma, W)$ is computed as:*

$$\begin{aligned} wsc(\gamma, W) = {} & w_r * |R| + w_u * |UA| + w_p * |PA| \\ & + w_d * |DUPA| + w_h * |t\_reduce(RH)| \end{aligned}$$

Role mining in general involves minimizing $wsc(\gamma, W)$. However, a minimization implicating all factors not only is too complex, but may not lead to a good RBAC system, as they may counteract with each other in the minimization. Depending on the objective to achieve, a specific role mining task often chooses to minimize a subset of factors relevant to the underlying objective. In particular, minimizing the number of roles $|R|$ might be the most studied role mining problem, where in the weighted structural complexity measure, w_r is a positive number while others are all 0. Such a specific role mining problem is referred to as basic RMP (Role Mining Problem), and it has been proven equivalent to the classic set cover problem. It is true that the number of roles can describe the complexity of a RBAC solution in a certain way. However, sometimes a RBAC solution minimizing the number of roles might not be able to capture the internal organizational structure within an company. We shall show this by a toy

[1] $\mathcal{Q}^+$ is the set of all non-negative rational numbers.

example with six users, three permissions, and user-permission assignments such that $u1 : \{p1, p2, p3\}, u2 : \{p1, p2, p3\}, u3 : \{p1, p2, p3\}, u4 : \{p1\}, u5 : \{p2\}, u6 : \{p3\}$.

Under the objective of minimizing $|R|$, it gives a solution such that each individual permission is a role. As a result, there are three roles in total and $u1$, $u2$ and $u3$ are assigned to three roles to cover their permissions. However, even without the semantic information of permissions and the knowledge of user responsibilities, by simply observing the data set, one would conjecture that the permission set of $\{p1, p2, p3\}$ should be one role, as the permission set is shared by three out of six users.

However, if we incorporate the consideration of the size of the total user-role assignments, it can lead us to the right track. With the goal of minimizing $|UA|$, each user will get one role and each unique set of user permissions is treated as a role. As a result, the role of $\{u1, u2, u3\}$ is discovered. This example suggests the importance of incorporating the complexity of $|UA|$ as a part of the role mining goal.

The sum of user-role assignments and role-permission assignments of $|UA| + |PA|$ is commonly viewed as the representation of the system administrative cost. The minimization of $|UA| + |PA|$ is called edge RMP [16]. However, as far as an end-user is concerned, $|UA|$ is the only part that she can experience of a RBAC system. A user would not care how complex PA is. For example, a student would not care about how many permissions her STUDENT role actually contains, and all she cares is the simplicity of executing her role (or roles).

Other evaluative criteria, such as minimizing the complexity of the resultant role hierarchy [10], are also more from the system administrator perspective, rather than the end-user perspective.

By examining existing evaluative criterion, we found that the only factor in the Weighted Structural Complexity Measure that matters to end-users is the size of user-role assignments $|UA|$. However, if the user-oriented RMP is defined as the minimization of $|UA|$, the trivially optimized solution is to create a unique role for every user. That absolutely contradicts to the premise of role mining, which is to map permission roles with functional roles. As such, a user-oriented RMP solution should balance the user-friendliness and the overall quality of the system. Among five factors in the weighted structural complexity measure, the number of roles $|R|$ would be the best representative of the succinctness and goodness of a RBAC solution and is also the most studied criterion. So we propose to define the user-oriented RMP as the minimization of the combination of the number of roles and the size of user-role assignments, $w_r * |R| + w_u * |UA|$.

3.2 User-Oriented RMP

User-Oriented Exact RMP. Given m users, n permissions, user-permission assignments $UPA_{m\times n}$, and the evaluation criteria of $w_r * |R| + w_u * |UA|$, the user-oriented exact RMP is to find $UA_{m\times k}$ and $PA_{k\times n}$ to completely reconstruct UPA while minimizing the evaluation criterion. It can be formulated as below:

$$\begin{aligned} &min \;\; w_r * |R| + w_u * |UA_{m\times k}| \\ s.t. &\Big\{ UPA_{m\times n} = UA_{m\times k} \otimes PA_{k\times n} \end{aligned} \tag{1}$$

where $\otimes$ is the Boolean product operator [6].

However, directly working on this formulation has two difficulties. First, it is not easy to determine weights of w_r and w_u in practice, as a role mining engineer may not have a global sense on the importance of $|R|$ and $|UA|$. Second, it is difficult to solve an optimization problem with a complex objective function. It would be relatively easier to solve an optimization problem with an objective of either $|R|$ or $|UA|$. In light of them, we redefine the user-oriented exact RMP as the following.

Problem 1 (User-Oriented Exact RMP). Given m users, n permissions, user-permission assignments $UPA_{m\times n}$ and a positive number t, it is to discover a role set $PA_{k\times n}$ and the user-role assignments $UA_{m\times k}$ such that: (1) the number of roles k is minimized, (2) the role assignments UA and the permission-role assignments PA accurately and completely reconstruct the existing user-permission assignments UPA, and (3) no user gets more than t roles.

Mathematically, it can be described in an optimization form as follows.

$$min\ \ k$$
$$s.t. \begin{cases} UA_{m\times k} \otimes PA_{k\times n} = UPA_{m\times n} \\ \sum_j UA(i,j) \leq t,\ \forall i \\ UA \in \{0,1\}^{m\times k}, PA \in \{0,1\}^{k\times n} \end{cases} \tag{2}$$

It is not difficult for a role mining engineer to find out the maximum roles a user can have. For example, it could be achieved through discussions with company operators and an investigation of the general organizational structure of the company. When the maximum roles each user can have is limited to a small number, $|UA|$ is naturally enforced to be small.

Another property is that Equation (2) can be easily converted to Equation (1) with the method of Lagrange multipliers. If we move the constraint $\sum_j UA(i,j) \leq t$ to the objective function by adding $\sum_j UA(i,j) - t$ as a penalty component, Equation 2 becomes

$$\begin{aligned} &min\ \ |R| + \sum_i \lambda_i (\sum_j UA(i,j) - t) \\ &s.t. \Big\{ UPA_{m\times n} = UA_{m\times k} \otimes PA_{k\times n} \end{aligned} \tag{3}$$

where λ_i is the Lagrange multiplier for the constraint of $\sum_j UA(i,j) \leq t$. Further, we could assume that all Lagrange multipliers have the same value λ. Then the equation is changed to the following.

$$\begin{aligned} &min\ \ w_r * |R| + \lambda |UA| - \lambda * t \\ &s.t. \Big\{ UPA_{m\times n} = UA_{m\times k} \otimes PA_{k\times n} \end{aligned} \tag{4}$$

Since $\lambda * t$ is a constant, it can be dropped from the objective function. Now the resultant optimization problem is the same as Equation 1. The effect of adjusting the Lagrange multiplier (penalty parameter) λ is equivalent to adjusting k, the maximum roles a user can have.

User-Oriented Approximate RMP. Role mining with exact coverage of permission assignments is only suitable when the given permission assignments contain no error. The recent research results on the role mining on noisy data[12,17]. suggest that when the given user-permission assignments contain noise, it is not necessary to enforce a complete reconstruction, as it causes the over-fitting problem. In such cases, approximate role mining may return better results. So in this paper, we also consider the user-oriented approximate RMP, which is defined as below.

Problem 2 (User-Oriented Approximate RMP). Given m users, n permissions, user-permission assignments $UPA_{m\times n}$, a positive integer number t and a positive fractional number δ, it is to discover a role set $PA_{k\times n}$ and the user-role assignments $UA_{m\times k}$ such that: (1) the number of roles k is minimized, (2) the role assignments UA and the role set PA reconstruct the existing user-permission assignments UPA with the error rate less than δ, and (3) no user gets more than t roles.

The problem can be roughly described in the following optimization form.

$$min\ \ k$$
$$s.t. \begin{cases} ||UA_{m\times k} \otimes PA_{k\times n} - UPA_{m\times n}||_1 \leq \delta \cdot \sum_{ij} UPA_{ij} \\ \sum_j UA(i,j) \leq t,\ \forall i \\ UA \in \{0,1\}^{m\times k}, PA \in \{0,1\}^{k\times n} \end{cases} \tag{5}$$

3.3 NP-hardness

Recall that the basic RMP is to minimize the number of role while the resultant RBAC solution completely reconstructs the given user-permission assignments. The user-oriented exact RMP is a generalization of the basic RMP. If we make the number of the maximum roles each user can have be a large enough number, so that the sparseness constraint does not take effect, then the user-oriented exact RMP becomes the basic RMP. The basic RMP is known to be NP-hard, as it can be reduced to the classic NP-hard set cover problem [15]. Therefore, the user-oriented exact RMP is NP-hard. Similarly, the user-oriented approximate RMP is a generalization of the approximate RMP, which is NP-hard. Thus, it is also NP-hard.

4 Optimization Model

Among many existing role mining approaches, the optimization approach has been favored by researchers, due to the existence of many public and commercial optimization software packages. The user-oriented RMP problems can be formulated by optimization models as well, which enables an engineer to directly adopt an existing software package.

We formulate the user-oriented exact RMP first, which can be viewed as a variant of the basic RMP with a constraint that each user cannot have more than t roles. Suppose we have located a set of q candidate roles, represented by a binary matrix

$CR \in \{0,1\}^{q \times n}$, where $CR_{kj} = 1$ means candidate role k contains permission j. Then the user-oriented exact RMP is reduced to finding the minimum roles from CR to completely reconstruct existing user-permission assignments while no one can have more than t roles. The problem can be formulated as the following ILP.

$$minimize \sum_k d_k$$
$$\begin{cases} \sum_{k=1}^{q} UA_{ik} CR_{kj} \geq 1, if\ UPA_{ij} = 1 \\ \sum_{k=1}^{q} UA_{ik} CR_{kj} = 0, if\ UPA_{ij} = 0 \\ d_k \geq UA_{ij}, \qquad \forall i,j \\ \sum_j UA_{ij} \leq t\ \forall i \\ d_k, UA_{ij} \in \{0,1\} \end{cases} \tag{6}$$

In the model, UA_{ik} and d_k are variables. The detailed description of the model is given as follows:

- Binary variable UA_{ik} determines whether candidate role k is assigned to user i and binary variable d_k determines whether candidate role k is selected. So the objective function $\sum_k d_k$ represents the number of selected roles.
- The first constraint enforces that if user i has permission j, at least one role containing permission j has to be assigned to user i.
- The second constraint enforces that if user i has no permission j, no role containing permission j can be assigned to user i.
- The third constraint $d_k \geq UA_{ij}$ ensures d_k to be 1 as long as one user has role k.
- $\sum_j UA_{ij} \leq t$ enforces that a user cannot have more than t role assignments.

The user-oriented approximate RMP can be viewed as a variant of the approximate RMP with the constraint that no user can have more than t roles. Similarly, we simplify the problem by locating a candidate role set CR. At the basis of the ILP formulation for the approximate RMP, an ILP formulation for the user-oriented approximate RMP is presented as follows.

$$minimize \sum_t d_k$$
$$\begin{cases} \sum_{k=1}^{q} UA_{ik} CR_{kj} + V_{ij} \geq 1, if\ UPA_{ij} = 1 \\ \sum_{k=1}^{q} UA_{ik} CR_{kj} - V_{ij} = 0, if\ UPA_{ij} = 0 \\ MU_{ij} - V_{ij} \geq 0, \forall i,j \\ U_{ij} \leq V_{ij}, \forall i,j \\ \sum_i \sum_j U_{ij} \leq \delta \cdot \sum_{ij} UPA_{ij} \\ d_k \geq UA_{ik}, \qquad \forall i,k \\ \sum_j UA_{ij} \leq t\ \forall i \\ d_j, UA_{ik}, U_{ij} \in \{0,1\}, V_{ij} \geq 0 \end{cases} \tag{7}$$

In the model, UA_{ik}, V_{ij}, U_{ij} and d_k are variables and M is a large enough constant. The detailed descriptions of the model are given as follows:

- In the first two constraints, V_{ij} acts as an auxiliary variable. Without V_{ij}, the constraints would enforce the exact coverage as the ILP model for the user-oriented

exact RMP. With the existence of V_{ij}, the exact coverage constraint is relaxed. The value of V_{ij} indicates whether the constraint for element (i, j) is violated.
- The third and fourth constraints convert V_{ij} to a binary value U_{ij}. If V_{ij} is 1, which means the constraint for element (i, j) is violated, U_{ij} has to be 1; otherwise U_{ij} is 0. The fifth constraint $\sum_{i,j} U_{ij} \leq \delta \cdot \sum_{ij} UPA_{ij}$ enforces the error rate to be less than δ.
- The constraint of $d_k \geq UA_{ik}\ \forall i, k$ enforces d_k to be 1 as long as a user is assigned to role K. So the objective function represents the number of roles being selected.
- $\sum_j UA_{ij} \leq t$ ensures no user gets more than t role assignments.

Although the optimization framework allows us to directly adopt fruitful optimization research results, the ILP in general is NP-hard. Existing algorithms and software packages for general ILP problems only work for small-scale problems. For mid to large size RMP problems, specially designed efficient heuristics are still required.

5 Heuristic Algorithm

In this section, we propose a tailored algorithm for the two user-oriented RMP variants formulated above. It is a heuristic solution, employing an iterative approach to discover roles. The key of our algorithm is a dynamic role generation strategy. Lately, we happened to notice that the idea of dynamic role generation was briefly mentioned in [11], but no further details were seen.

5.1 User-Oriented Exact RMP

The user-oriented exact RMP is to find a minimum set of roles to accurately and completely reconstruct the existing user-permission assignments with the constraint that no user can have more than t roles. Before coming to the details of our algorithm, we start by introducing a preprocessing stage that helps to reduce the problem complexity.

In the preprocessing stage, there are two steps. The first step is to reduce the data size by removing users with the same permission assignments. This step is also employed in other role mining methods, such as [18]. To do so, we group all users who have the exact same set of permissions, which can be done in a single pass over the data by maintaining a hash table of the sets of permissions gradually discovered.

The second step identifies a subset of users U' who have permissions that no other people have. These user-permissions assignments, $\{UA_{i:} | i \in U'\}$, will be included into the final role set. In other words, these users only get one role, which are themselves. Our argument is that if a user has a permission that is exclusively for herself, she must have at least one role containing that permission, and that role is not shared by other people. As such, from the end-user perspective, why not simply package all permissions of that user as one role and assign the only role to her? Therefore, the number of role assignments is significantly reduced while without increasing the total role number. This preprocessing step can significantly reduce the data size as well and simplify the subsequent role mining task.

Note that after the two preprocessing steps, in the remaining data, all user permissions assignments are unique and every permission is assigned to at least two users.

The general structure of our algorithm is to iteratively choose a candidate role and assign it to users until all existing permission assignments are covered while the constraint that no user gets more than t roles is carefully respected. We mention that such an idea of iterative role assignment has also been used in many other role mining methods such as the Lattice Model [2], the FastMinder [18] and the optimization-based heuristic [6]. The distinguishing element of our algorithm is the way of generating candidate roles.

The core of our algorithm is a dynamic role generation strategy. All of the other role mining algorithms generate a static set of candidate roles. Given n permissions, there are 2^n possible roles. If we consider too many candidate roles, the computing time is expensive. Conversely, if we consider only a very limited set of candidate roles, we might not be able to find a good role set. To avoid the extreme cases, our strategy is dynamic candidate roles generation. Specifically, rather than generating a static set of roles at the start of the algorithm, we generate a small set of promising roles at each iteration of the algorithm and the role set is updated according to the remaining user-permission assignments as the algorithm proceeds. There are two advantages: (i) we do not need to maintain and consider a large candidate role set all the time; (ii) the candidate role pool always keeps the potentially interesting roles.

In particular, we always consider the remaining user-permission assignments as potentially interesting roles. For instance, consider Table 1 as the existing user-permission assignments. Our algorithm treats the permission assignments for each user as a candidate role. So in this case, there are three candidate roles: cr1 (0 0 1 1 1 1), cr2 (0 0 1 1 0 0), and cr3 (1 1 1 1 0 0).

Table 1. Existing Assignments

	p1	p2	p3	p4	p5	p6
u1	0	0	1	1	1	1
u2	0	0	1	1	0	0
u3	1	1	1	1	0	0

Table 2. Remaining Assignments

	p1	p2	p3	p4	p5	p6
u1	0	0	0	0	1	1
u2	0	0	0	0	0	0
u3	1	1	0	0	0	0

Suppose now $cr2$ is chosen and it is assigned to all of the three users. Then the remaining permission assignments become Table 2, and they are treated as candidate roles for the next step in the algorithm. So the candidate roles are updated to be the following: cr1 (0 0 0 0 1 1) and cr2 (1 1 0 0 0 0).

With the candidate roles being defined, we need to figure out two things: (1) how to select a candidate role at each step? (2) how to enforce the constraint that no user can have more than t roles. For the first question, there are some studies and discussions in the literature. Here are some well known strategies. Vaidya et al. [18] chooses the candidate role which covers the most remaining permission assignments. Ene et al. [2] selects the candidate role with the least number of permissions. We have tested both of them and found out they do not work well in our case. As such, we use an alternative strategy: we choose the candidate role which covers the most users. In other words, the selected candidate role can be assigned to the most users. In fact, our strategy is justified by that the need for a permission set to become a role comes from the fact that they are shared by many people. Suppose that $\{p1, p2\}$ are possessed by three people, while

Algorithm 1. User-Oriented Exact RMP

Input: UPA, t
Output: UA, PA
1: $UA \leftarrow \emptyset, PA \leftarrow \emptyset, UPA' \leftarrow \emptyset$;
2: $CRoles \leftarrow UPA$;
3: **while** $UPA' \neq UPA$ **do**
4: Call RSelector;
5: Call CGenerator;
6: **end while**

$\{p1, p2, ..., p10\}$ are possed by only one person. It is more reasonable to make $\{p1, p2\}$ as a role than $\{p1, p2, ..., p10\}$. To illustrate this candidate role selection strategy, look at Table 1 again. Among those three candidate roles, (0 0 1 1 0 0) can be assigned to three people, so it is chosen.

To enforce the constraint that no user gets more than t roles, we make some special arrangement, when a user U_i has been covered by $t-1$ roles and still has uncovered permissions. In such a case, we either need to create a role to cover all remaining permissions of U_i or revoke roles that have been assigned to U_i. Suppose we create a new role which consists of all remaining permissions of U_i and assign it to U_i. Then, we may need to check if the new role can be repetitively used by other users, otherwise it is costly. If no one else can take the new role, we make all permissions of U_i as a single role. Thus we can revoke all roles that have been assigned to U_i and assign the sole role to the user. In this way, at the same cost of adding one role, the role assignments for the user are significantly reduced, which is exactly the goal of this work. Based on this idea, the following steps are implemented to enforce that no user gets more than t roles.

When a user U_i has been covered by $t-1$ roles and still has uncovered permissions, we stop choosing roles from the candidate role set. Instead, we first treat the uncovered permissions of U_i as a candidate role, and evaluate its suitability by checking if some other user who has been assigned less than $t-1$ roles will take this candidate role. If so, it means that the role can be repetitively used, then we include the role into the final role set and assign it to users. Otherwise, we discard it, and then make all permissions of the user U_i as a role, assign the role to U_i and delete all of the other role assignments to U_i. The complete algorithm is stated in Algorithms 1-3 ($UA_{i:}$ denotes the ith row of UA, which represents the role assignments to user i; $UPA_{i:}$ denotes the ith row of UPA, which represents the permissions assigned to user i).

5.2 User-Oriented Approximate RMP

The user-oriented approximate RMP is the same as the user-oriented exact RMP, except that the complete reconstruction is not required. The above algorithm for the user-oriented exact RMP can be easily modified for the the user-oriented approximate RMP by changing the termination condition from $UPA \neq UPA'$ to $||UPA - UPA'|| > \delta$. Consequently, the algorithm stops early and avoids covering too much noisy information.

Algorithm 2. RSelector

Input: $UPA, UPA', CRoles, UA, t$
Output: r, UA, PA, UPA'
if $\exists i\ s.t.\ |UA_{i:}| = t - 1$ **then**
  $temp \leftarrow UPA_{i:} \setminus UPA'_{i:}$
  if $\exists j\ s.t.\ UPA_{j:} \supseteq temp, j \neq i$ **then**
    $r \leftarrow temp$
  else
    $r \leftarrow UPA_{i:}$
    $UA_{i:} \leftarrow \emptyset$
  end if
else
  $r \leftarrow argmax_{r \in CRoles} |UPA_{i:}\ s.t.\ UPA_i \supseteq r|$
end if
Update PA by including r;
Update UA by assigning r to all valid users;
Update UPA';

Algorithm 3. CGenerator

Input: UPA, UPA'
Output: $CRoles$
$CRoles \leftarrow \emptyset$;
for $i = 1 \rightarrow |UPA|$ **do**
  if $UPA_i \setminus UPA'_i \neq \emptyset$ **then**
    $CRoles \leftarrow \{CRoles, UPA_i \setminus UPA'_i\}$
  end if
end for

5.3 Computational Complexity Analysis

The key of the above user-oriented role mining algorithm is the continuous updating of candidate roles. At each iteration of the algorithm, a candidate role is chosen and the role coverage is determined. The total computations then depend on the number of iterations. Consider a user-permission dataset with m users and n permissions. According to our algorithm, at each iteration, at least one user's permissions are completely covered. So the maximum required iterations are m. At each iteration step, each candidate role is compared against each remaining user. As the number of candidate roles is less than m, the number of remaining users is less than m and each user (or role) has up to n permission, so the incurred computations at each iteration cannot be over m^2n. Therefore the computation complexity of our algorithm is upper bounded by m^3n.

6 Experiments and Results

Experiments are conducted on benchmark access control datasets. They are **americas_small**, **apj**, **healthcare**, **domino**, **firewall1** and **firewall2**, which can be found at the

HP website [2]. **americas_small** and **apj** are user profiles from Cisco firewalls. **healthcare** was obtained from the US Veteran's Administration. The **domino** graph is from a set of user and access profiles for a Lotus Domino server. **firewall1** and **firewall2** are results of running an analysis algorithm on Checkpoint firewalls. Descriptions on the data sets including the number of users, the number of permissions, and the size of user-permission assignments are given in Table 3. More detailed descriptions can be found in [2].

The first experiment evaluates the user-oriented exact RMP. We want to know whether our **Dynamic** algorithm can effectively enforce the sparseness constraint and whether the output of the algorithm is comparable to the optimal RBAC solution without the sparsity constraint. To find the answers, we run the **Dynamic** algorithm on those real data sets with different sparsity constraints. We compare our results with the benchmark role mining algorithm, **Lattice** [2]. As far as we know, **Lattice** has the best reported result with respect to the minimization of the number of roles and the minimization of the system administrative cost. The experimental results are reported in Tables 4 - 7. In these tables, δ denotes the error rate. The exact RMP requires the error rate to be 0. So we only look at the portion of the results with $\delta = 0$. Other parameters are: t denotes the maximum number of role assignments enforced in our algorithm, $|UA|$ denotes the size of user-permission assignments and $|PA|$ denoting the size of permission-role assignments. Note that δ and t has no effect on the **Lattice** algorithm, as **Lattice** returns an exact RBAC solution and the solution is unique. In the results, the value at the row of **Lattice** and the column of t is the maximum number of roles that a user has in the RBAC solution returned by the **Lattice** algorithm.

Table 3. Data Description

Data Set	$\|U\|$	$\|P\|$	$\|UPA\|$
healthcare	46	46	1,486
domino	79	231	730
firewall1	365	709	31,951
firewall2	325	590	36,428
apj	2,044	1,164	6,841
americas small	3,477	1,587	105,205

In the results, when t decreases, the size of UA decreases accordingly. However, the value of $|UA| + |PA|$ changes in an opposite direction. This matches our expectation. Specifically, when t is a small value, each user gets few role assignments. Thus, we need roles with more permissions, so each user can still get enough permission assignments. When the sparseness constraint becomes more strict, the number of required roles increases. As a result, the value of $|PA|$ increases accordingly.

Furthermore, we are pleased to see that even with the sparseness constraint being enforced, the complexity of the RBAC solution returned by **Dynamic** is still comparable to that of **Lattice**. For example, in Table 8, when t is 2, **Dynamic** returns a RBAC solution with only 18 roles, while **Lattice** returns a solution with 15 roles and the maximum role assignments of 4. Another observation is that the $|UA|$ value of the solutions

[2] `http://www.hpl.hp.com/personal/Robert_Schreiber/`

Table 4. fire1

	δ	t	$\|R\|$	$\|UA\|$	$\|UA\|+\|PA\|$
Dynamic	0.00	2	90	365	7100
		4	85	454	6890
		6	84	600	6897
		8	80	1516	6638
	0.05	2	37	250	5688
		4	48	361	5685
		6	41	529	5523
		8	39	1416	5671
	0.10	2	30	250	4619
		4	27	330	4297
		6	26	439	3868
		8	14	1464	2970
	0.15	2	17	250	2661
		4	9	422	1762
		6	10	563	1810
		8	7	1334	2055
	0.20	2	11	250	1881
		4	8	426	1655
		6	6	1131	1839
		8	6	1131	1839
Lattice	0.00	9	66	874	1953

Table 5. americas small

	δ	t	$\|R\|$	$\|UA\|$	$\|UA\|+\|PA\|$
Dynamic	0.00	2	259	3477	25229
		4	256	3722	23610
		6	249	3890	22194
		8	246	4269	20119
	0.05	2	224	3283	23174
		4	184	3566	21349
		6	183	3780	20109
		8	185	4160	18406
	0.10	2	205	3220	21421
		4	154	3453	19870
		6	157	3715	18451
		8	147	4042	16318
	0.15	2	180	3096	19929
		4	138	3383	17966
		6	137	3604	16630
		8	127	4036	14698
	0.20	2	171	3096	18505
		4	130	3384	16530
		6	123	3673	14971
		8	117	3915	14238
Lattice	0.00	10	192	4782	9830

Table 6. fire2

	δ	t	$\|R\|$	$\|UA\|$	$\|UA\|+\|PA\|$
Dynamic	0.00	2	11	325	1499
	0.05	2	11	325	1499
	0.10	2	7	285	1092
	0.15	2	7	285	1092
	0.20	2	7	285	1092
Lattice	0.00	3	10	434	1110

Table 7. domino

	δ	t	$\|R\|$	$\|UA\|$	$\|UA\|+\|PA\|$
Dynamic	0.00	2	23	79	716
	0.05	2	17	71	695
	0.10	2	14	64	680
	0.20	2	10	53	657
Lattice	0.00	3	20	110	713

returned by **Dynamic** can be much less than that of the solutions returned by **Lattice**. For instance, in Table 5, the value of $|UA|$ for **Dynamic** with δ of 0 and t of 2 is 3477, while that for **Lattice** is 4782.

The second experiment is to study the user-oriented approximate RMP. We want to know how the RBAC solution varies with the error rate. We run the **Dynamic** algorithm by varying the value of δ from 0.05 to 0.20. Results are reported in Tables 4 - 7. We observe that when the complexity of the RBAC solutions decrease drastically when δ increases. For instance, in Table 4, with t of 8, only 39 roles are required to cover the 95 percent of permission assignments (i.e., $\delta = 0.05$), while 80 roles are required for the complete coverage (i.e., $\delta = 0$). In terms of the coverage of permission assignments, those 39 roles appear more promising than the remaining 41 roles. In cases where data noise is believed to exist, the approximate version of **Dynamic** appears to be more useful.

To summarize, the two experiments have demonstrated the effectiveness of our user-oriented RMP approach. We highlight that one primary advantage of **Dynamic** is that it allows a RBAC engineer to tune the sparsity constraint to reflect the real need. This feature is not supported by any existing role mining method. More importantly, the overall system complexity of the resultant solution is comparable to that of the optimal solution without any sparsity constraint.

Table 8. healthcare

	δ	t	$\|R\|$	$\|UA\|$	$\|UA\| + \|PA\|$
Dynamic	0.00	2	18	46	545
		3	18	53	468
	0.05	2	5	46	191
		3	6	54	201
	0.10	2	5	45	191
		3	6	54	201
	0.15	2	5	45	191
		3	6	54	201
	0.20	2	5	45	191
		3	3	72	126
Lattice	0	4	15	106	209

Table 9. apj

	δ	t	$\|R\|$	$\|UA\|$	$\|UA\| + \|PA\|$
Dynamic	0.00	2	564	2044	5565
		3	497	2218	5221
		4	485	2277	5096
	0.05	2	477	1664	4995
		3	394	1538	4362
		4	381	1571	4215
	0.10	2	407	1292	4449
		3	358	1449	4122
		4	352	1501	4043
	0.15	2	348	876	3878
		3	322	1258	3831
		4	317	1313	3766
	0.20	2	315	816	3609
		3	290	1064	3533
		4	289	1169	3550
Lattice	0.00	6	454	2437	4117

7 Conclusion

In this paper, we studied the role mining problem from the end-user perspective. Unlike other existing role mining approaches which primarily aim to reduce the administrative workload, our approach strives to incorporate better user experience into the role decision process. As end-users prefer simple role assignments, we add a sparseness constraint that mandates the maximum number of roles a user can have to the role mining process. The number usually can be determined in practice by a brief study on the general business processes of an organization. Basing on this rationale, we formulated user-oriented role mining as two specific problems. One is the user-oriented exact RMP, which is obliged to completely reconstruct given permission assignments while obeying the sparseness constraint. It is applicable for scenarios where the given dataset has no noise. The other is the user-oriented approximate RMP, which tolerates a certain amount of deviation from the complete reconstruction. It suits for datasets containing noises. We studied existing role mining methods, and found that some of them can be applied to our problems with simple modification. For better efficiency, we also designed new algorithms tailored to our problems, which are based on a dynamic candidate role generation strategy. Experimental results demonstrate the effectiveness of our approach in discovering a user-oriented RBAC solution while without increasing the overall administrative workload too much.

Future work can go along two directions. One is to study the feasibility of employing some statistical measures such as Bayesian information criterion to facilitate the role mining process. The motivation is that sometimes the accurate sparseness constraint (the maximum role that a user can have) is not available. We could employ some statistical criteria to choose the RBAC model with a good balance of model complexity and describability. The other direction is to consider the dynamic sparseness constraint. In this work, we assume that the same sparseness constraint is enforced to everyone. However, it might be the case that some user requires many role assignments due to some need. In such cases, a more practical role mining approach is to minimize the sparsity of the whole user-role assignments rather than enforcing a sparseness constraint for every user.

References

1. Coyne, E.J.: Role engineering. In: RBAC 1995: Proceedings of the first ACM Workshop on Role-based Access Control, p. 4. ACM, New York (1996)
2. Ene, A., Horne, W., Milosavljevic, N., Rao, P., Schreiber, R., Tarjan, R.E.: Fast exact and heuristic methods for role minimization problems. In: SACMAT 2008: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, pp. 1–10. ACM, New York (2008)
3. Frank, M., Basin, D., Buhmann, J.M.: A class of probabilistic models for role engineering. In: Proceedings of the 15th ACM Conference on Computer and Communications Security (2008)
4. Frank, M., Streich, A.P., Basin, D., Buhmann, J.M.: A probabilistic approach to hybrid role mining. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS 2009, pp. 101–111. ACM, New York (2009)
5. Kuhlmann, M., Shohat, D., Schimpf, G.: Role mining - revealing business roles for security administration using data mining technology. In: SACMAT 2003: Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies, pp. 179–186. ACM, New York (2003)
6. Lu, H., Vaidya, J., Atluri, V.: Optimal boolean matrix decomposition: Application to role engineering. In: IEEE 24th International Conference on Data Engineering, pp. 297–306 (2008)
7. Lu, H., Vaidya, J., Atluri, V., Hong, Y.: Extended boolean matrix decomposition. In: IEEE International Conference on Data Mining (2009)
8. Lu, H., Vaidya, J., Atluri, V., Hong, Y.: Constraint-aware role mining via extended boolean matrix decomposition. IEEE Transactions on Dependable and Secure Computing 9(5), 655–669 (2012)
9. Ma, X., Li, R., Lu, Z.: Role mining based on weights. In: SACMAT, pp. 65–74 (2010)
10. Molloy, I., Chen, H., Li, T., Wang, Q., Li, N., Bertino, E., Calo, S., Lobo, J.: Mining roles with semantic meanings. In: SACMAT 2008: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, pp. 21–30. ACM, New York (2008)
11. Molloy, I., Chen, H., Li, T., Wang, Q., Li, N., Bertino, E., Calo, S.B., Lobo, J.: Mining roles with multiple objectives. ACM Trans. Inf. Syst. Secur. 13(4), 36 (2010)
12. Molloy, I., Li, N., Qi, Y.A., Lobo, J., Dickens, L.: Mining roles with noisy data. In: Proceeding of the 15th ACM Symposium on Access Control Models and Technologies, SACMAT 2010, pp. 45–54. ACM, New York (2010)
13. Neumann, G., Strembeck, M.: A scenario-driven role engineering process for functional rbac roles. In: Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies, SACMAT 2002, pp. 33–42. ACM, New York (2002)
14. Schlegelmilch, J., Steffens, U.: Role mining with orca. In: SACMAT 2005: Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies, pp. 168–176 (2005)
15. Vaidya, J., Atluri, V., Guo, Q.: The role mining problem: finding a minimal descriptive set of roles. In: SACMAT, pp. 175–184 (2007)
16. Vaidya, J., Atluri, V., Guo, Q., Lu, H.: Edge-rmp: Minimizing administrative assignments for role-based access control. Journal of Computer Security 17(2), 211–235 (2009)
17. Vaidya, J., Atluri, V., Guo, Q., Lu, H.: Role mining in the presence of noise. In: Foresti, S., Jajodia, S. (eds.) Data and Applications Security and Privacy XXIV. LNCS, vol. 6166, pp. 97–112. Springer, Heidelberg (2010)
18. Vaidya, J., Atluri, V., Warner, J.: Roleminer: mining roles using subset enumeration. In: The 13th ACM Conference on Computer and Communications Security, pp. 144–153 (2006)

Hypervisor Event Logs as a Source of Consistent Virtual Machine Evidence for Forensic Cloud Investigations

Sean Thorpe[1], Indrajit Ray[2], Tyrone Grandison[3], Abbie Barbir[4], and Robert France[2]

[1] Faculty of Engineering & Computing, University of Technology, Kingston, Jamaica
thorpe.sean@gmail.com
[2] Department of Computer Science, Colorado State University, Fort Collins, USA
{Indrajit,france}@cs.colostate.edu
[3] Proficiency Labs Intl, Ashland, Oregon
tgrandison@proficiencylabs.com
[4] Bank of America
abbie.barbir@bankofamerica.com

Abstract. Cloud Computing is an emerging model of computing where users can leverage the computing infrastructure as a service stack or commodity. The security and privacy concerns of this infrastructure arising from the large co-location of tenants are, however, significant and pose considerable challenges in its widespread deployment. The current work addresses one aspect of the security problem by facilitating forensic investigations to determine if these virtual tenant spaces were maliciously violated by other tenants. It presents the design, application and limitations of a software prototype called the Virtual Machine (VM) Log Auditor that helps in detecting inconsistencies within the activity timelines for a VM history. A discussion on modeling a consistent approach is also provided.

1 Introduction

Temporal event logs meticulously record events that have occurred in the history of the computer system, and therefore, constitute a valuable source of digital evidence. Event logs are generated by the operating system as well as by other subsystems and their applications. There has been a significant amount of research about the examination and the auditing of such logs for post incident forensic purposes [1, 2]. With cloud computing service environments, the expectations for post incident forensics is no different. Cloud computing is predicated on the well known service oriented architecture (SOA) and harnesses the power of the virtualization stack. The known services offered across the virtual stack layers are Infrastructure as Service (IAAS), Platform as a Service (PAAS) and Software as a Service (SAAS). In our work the PAAS provision, which handles the hypervisor logs, is our primary concern. By studying the hypervisor logs, this work represents the first body of work in the literature that seeks to explore closing the semantic gap of how eye witness forensic data that can be collected between the lower layers and higher virtualization layers and is motivated by prior work [1, 2, 21].

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 97–112, 2013.

In this work the log categorization used is the hypervisor event logs, which contains a hierarchy of application logs, security logs, error logs and system logs. Some have likened a log file in computer forensics to an eyewitness in a traditional crime scene [1]. The analogy seems befitting when using the hypervisor logs to manage virtual crime scenes for which the multi-tenant VMs are all co-located.

Providing a picture of what happened through the analysis of the hypervisor event logs at a virtual crime scene is no trivial task. Detecting patterns that may be events of interest by hand in a large log file is not feasible. Managing the sheer volume of log data is a well known quantity problem. Logs must be parsed programmatically and even this can take a very long time, with the exact amount of time varying significantly, and is often dependent on the type of algorithm employed to detect the patterns of interest [2].

As log data are explicitly a record of events, establishing their reliability is of particular importance. Log files are written to very frequently and hence may get corrupted or could be difficult to understand; as records may be saved in an unexpected sequence as a result of unusual system behavior, e.g. software bug or power outage. As log files are an obvious record of events, they are also an obvious target for tampering. Suspicious events indicating that something is wrong may be deliberately removed, rendering all or a part of the log potentially fraudulent.

Our work addresses the issue of deliberate tampering, internal contradictions and inconsistent entries with these hypervisor event logs within the storage area network (SAN) data centre environments. We attempt to improve on the state of the art by providing a forensic platform that transparently monitors and records data access events using these PAAS logs as a form of static snapshot state analysis for a post incident VM cloud investigation. This approach complements the traditional statistical trace analysis methods and the VM memory introspection methods established in prior work [21, 22, 25]. As it relates to using the PAAS logs to detect VM attacks, particularly session hijacking, this represents independent ongoing work using several data mining techniques [23, 24] that unearth ground truth forensic evidence based on anomalous patterns detected from such logs. As it is now, the timestamps recorded in the hypervisor event logs may be unreliable, as result of both flaws in the clocks that generate them and the nature of the software that records entries to these data cloud logs.

The rest of this paper is organized as follows. Section 2 introduces our VM profiling model. Section 3 examines approaches for the detection of inconsistency in timelines, dealing both with inconsistencies in VM event timestamps and VM events omitted from the hypervisor kernel system's record. Section 4 describes the experiments with the tool for testing the approaches discussed in Section 3. Section 5 describes the rule base for the experiments and evaluates the detection techniques based on the rule base. Section 6 provides a formal discussion on constructing the consistency approach, and Section 7 provides the discussion of the experimental results. Section 8 presents a discussion of future work, Section 9 provides the related work and we conclude in Section 10.

2 Virtual Machine Profiling Model

This work is based on concepts from the computer-profiling model described by Marrington et al. [9]. The authors' model of a computer system consists of objects representing the various entities that form part of the computer system's operation. These entities include users, data files, system software, hardware devices, and applications. The objects discovered on the computer system under examination, collectively referred to as the set ***O***, are classified according to their type. In their model [9], there are four broad types of objects (Application, Principal, Content and System) with increasingly specific subtypes.

We adopt in our work each of these categories as VM sets. The set of Application objects, ***A***, consists of all the application software on the VM host computer system. The set of Principal objects, ***P***, consists of all the computer system's users and groups, and all of the people and organizations otherwise discovered in the examination of the computer system. Of these objects, some Principal objects are described as *canonical* if they represent definite entities on the computer system that are actors in their own right, such as users and groups. Principal objects may be described as *non-canonical* if they represent people or groups of people who may not be actors on the system, but may be, for instance, people mentioned in documents. The set of Content objects, ***C***, consists of all the documents, images and other data files on the host computer system from which the VMs are running. The set of VM System objects, ***S***, consists of all the VM configuration information, system software and hardware devices on the VM host computer system. ***A***, ***S***, ***C***, and ***P*** are subsets of ***O***, the set of objects on the cloud system under investigation.

We characterize our model with the set of all times in the history of the VM host computer system, $\boldsymbol{T_v}$, and the set of all events, ***Evt***, which have taken place in the history of the VM host computer system. Let t be a time in $\boldsymbol{T_v}$, x be the VM object that triggered the VM event, y be the object that was the target of the event, ε describe the action of the event, and α describe the result of the event (successful, unsuccessful, or unknown). An event evt in the set ***Evt*** consists of the quintuple $evt = (t, x \in O, y \in O, \varepsilon, \alpha)$.

This same finite set ***Evt*** consists of two enumerable subsets, and one subset which cannot be enumerated. The first subset – the recorded events set ***EvtR*** – consists of VM events that are recorded in the VM host computer hypervisor system's logs. The second subset – the inferred events set ***EvtI*** – consists of events that are not recorded in logs, but that can be inferred on the basis of other digital evidence on the system, such as relationships between different objects. These two sets do not necessarily describe the complete history of the cloud system in an exhaustive manner. There may be other events that took place and were unrecorded and left no artifact from which they could be inferred. These VM events are obviously unknown, and comprise the final subset of ***Evt***.

The set ***EvtI*** is particularly vulnerable to inconsistency or incompleteness in the data obtained from the VM target computer's file system. Contradictory, inaccurate or missing information can lead to an incomplete timeline of a user's activity. ***EvtR*** is a

direct representation of the contents of the VM host hypervisor system's logs, and consequently, incorporates any inaccurate event records in the system logs. Further, if a VM event is not logged, and cannot be inferred, it will not be an element of either ***EvtR*** or ***EvtI***. Handling unknown events within the VM history of the computer system is a challenge and hence the less accurate the timeline of the target computer's activity will be. We address this challenge in an independent paper. For this work, we focus on the declared events and making the inferences from these stated events. This work provides a means for the semi-automated detection of inaccuracy or incompleteness leading to chronological inconsistency in timelines of VM computer activity.

Marrington et al. [6], discussed a timestamp-based technique for building a timeline about a given object in the profile of the computer system. However, their approach is not resilient to inaccuracies in timestamps, which may cause VM events to appear out of sequence. Missing events, whether removed manually or simply never recorded, lead to timelines that may present events out of the context they actually occurred. We posit that as a general principle, the failure to detect an inconsistency in a timeline is a greater problem for the purposes of VM activity time-lining than falsely identifying an event as inconsistent. This is simply because false positives can be manually investigated and dismissed, whereas false negatives will never receive further attention. Nevertheless, it is obviously desirable to minimize the rate of false positives in all detection techniques.

A limitation of any time-lining activity based on timestamps provided by a computer's system clock is the inaccuracy inherent in such clocks. This inaccuracy in computer-generated timestamps is normal, and the solution suggested most frequently in the literature is to note the system clock time of a computer under investigation at the time of its examination and to determine the discrepancy between that time and the time of a reference clock [1, 8]. However, this solution does not address the issue of clock skew. A few works, notably [2, 10], propose algebra for the formal expression of falsifiable hypotheses about the discrepancy between a computer's clock and physical time. The term proposed for such a phenomenon is a clock hypothesis. In practice, it would be necessary to form a clock hypothesis for every moment in time throughout the history of the VM host system. Our tool is intended to detect internal inconsistency in timelines. An investigator could potentially be assisted in the formation of VM clock hypotheses using the output of our tool.

3 VM Log Auditor for Timelines

We now describe the approaches employed by our log auditor software to detect inconsistency in timelines. Inconsistency in virtual machine computer activity timelines can arise because hypervisor kernel log events in the timeline are out of sequence, or VM events that should be in the timeline are missing.

3.1 Detecting Out of Sequence VM Timelines

There are some VM events that need to occur before some other event can happen. This sort of relation between events is described as the *happened-before* relation [4]. Gladyshev and Patel [5] discuss the application of the *happened-before* relation to a forensic context. An example of such a relation between two events would be that a VM user x must "login" successfully to the computer system before the user x can "execute" the application y. So the *happened-before* ($\rightarrow$) relation implies that in the VM activity timeline, the time of the VM login event must be before the time of the execution event. We express this as follows. Let $x \in \boldsymbol{P}$, $y \in \boldsymbol{A}$, and $t_m, t_n \in \boldsymbol{T_y}$. Then $((t_m, x, y, \text{login, success}) \rightarrow (t_n, x, \text{VM system, execution, success})) \Rightarrow (t_m > t_n)$, where $\Rightarrow$ is the logical implication operator . Note that the *happened-before* relation is transitive [4, 5]:

After the construction of a VM log timeline (which is a sequence over the set ***Evt***) in the log auditor's execution process, an evaluation can be applied to all VM events ordered by their timestamp. If a VM event $vmevt_a$ has a *happened-before* relation to $vmevt_b$, but the VM kernel log timestamp (t_b) of $vmevt_b$ suggests that $vmevt_b$ occurred before $vmevt_a$ then we can say that t_a and t_b are inconsistent. In order to detect this inconsistency, a rule base must be created which describes the *happened-before* relations for the various types of events [15]. When the VM timeline is evaluated against the rules base, the inconsistent events can be identified and assertions about their time stamps can be made.

Consider the two rules $vmevt_a \rightarrow vmevt_b$ and $vmevt_b \rightarrow vmevt_c$ with

$$vmevt_a = \left(t_a \in T_y,\ x,\ system,\ \text{login, success}\right)$$

$$vmevt_b = \left(t_b \in T_y,\ x,\ a,\ \text{execute},\ \alpha \in \{\text{success, fail, unknown}\}\right)$$

$$vmevt_c = \left(t_c \in T_y,\ x,\ system,\ \text{logout, success}\right)$$

where x is a User VM object, a is an Application VM objects, and *system* is a VM System object representing the target VM computer system itself. Then, by the transitivity property of the *happened-before* relation,

$$\left(\left(vmevt_a \rightarrow vmevt_b\right) \wedge \left(vmevt_b \rightarrow vmevt_c\right)\right) \Rightarrow \left(vmevt_a \rightarrow vmevt_c\right)$$

For the purposes of this example, let the time-lining function $VH(x)$ produce a timeline (where a timeline is an ordered set of discrete time instances) corresponding to a single VM user session of the user x. The first rule states that a user x must be logged in before executing any application. The second rule states that user x cannot have logged out before performing that execution. If the execution event $vmevt_b$ occurs, the login event $vmevt_a$ must happen before it, and $vmevt_b$ must happen before the logout event $vmevt_c$. Therefore, the physical time t_c at which the event $vmevt_c$ must have occurred must be after the physical time t_b at which the event $vmevt_b$ must have occurred, which must in turn be after the physical time t_a at which the event $vmevt_a$ must have occurred.

This is stated as: $\left(VH(x) \supseteq \langle vmevt_a, vmevt_b, vmevt_c \rangle\right) \Rightarrow \left(t_a > t_b > t_c\right)$ where $\langle \circlearrowright \rangle$ denotes an ordered set.

If, given the two rules $vmevt_a \rightarrow vmevt_b$ and $vmevt_b \rightarrow vmevt_c$, it is not the case that $t_c > t_b > t_a$, then the timestamps (t_a, t_b, t_c) do not reflect the physical times at which the VM events must have occurred. The VM timestamps are therefore inaccurate, as they suggest an internally inconsistent chronology. From this example, the utility of the *happened-before* relation as a basis for proposing rules for the detection of inconsistent VM events is apparent. A hypothesized chronology of a VM computer system can be evaluated for internal inconsistencies by testing the hypothesized sequence of events against a set of *happened-before* rules.

3.2 Detecting Missing VM Events

There are some *happened-before* relations where the first VM event is a precondition for the second. In such relations, the presence of the second VM event necessarily implies the presence of the first. In the example in Section 3.1, the VM login event $vmevt_a$ must occur before the VM application execution event $vmevt_b$; in other words, if $vmevt_b$ occurred, then $vmevt_a$ must also have occurred. Note that this does not hold true for all *happened-before* relations. This can be seen in the same example, where although the execution event $vmevt_b$ must happen before the logout event $vmevt_c$ in order for $vmevt_b$ to happen at all, the occurrence of the logout event $vmevt_c$ does not imply that $vmevt_b$ also happened; $vmevt_b$ is not a precondition for $vmevt_c$.

[10] extends the use of the *happened-before* relation of [3, 5] to imply causality. The *happened-before* relation is therefore equivalent to the "precondition" predicate. For the purposes of the log auditor, it is preferable to maintain the *happened-before* relation as described by [3, 5] and to employ the "precondition" predicate to imply a causal relationship. The *happened-before* relation allows for the detection of events that are listed in the timeline out of the sequence in which they must have occurred, whereas the "precondition" predicate allows for the detection of missing events.

If the VM event $vmevt_a$ that "happened" does not exist in either the set of recorded VM events, ***EvtR***, or in the existing set of inferred VM events, ***EvtI***, then it is a missing VM event. It is a missing VM event because it was removed from, or never recorded in the VM's hypervisor kernel logs, and it was not previously inferred on the basis of relationships and object fields. These VM events could also be called *inferred* VM events, but it is convenient to preserve a distinction between events detected using this approach and other VM inferred events.

Precondition VM events which are absent from ***EvtR*** and ***EvtI*** can be added to the set of missing VM events, which we call ***EvtM***.

The VM login event $vmevt_a$ and the VM application execution event $vmevt_b$, and the logout event $vmevt_c$ have the same definitions as in the previous example. The new rule states that if the event $vmevt_b$ occurred in the timeline of the VM User object x, then the event $vmevt_a$ must also have occurred.

Missing VM events are suspicious and hence important. They are important because it is possible to deliberately delete them from the hypervisor kernel system

logs. Detecting that an event is missing allows for the construction of a more complete timeline, hence helping the VM investigator gain a more complete understanding of the VM computer system. By automatically indicating that at a particular point in the timeline an event was either not recorded or its record was deleted, the forensic software could provide a lead for the subsequent manual investigation. This, in turn, may determine why the record is missing. However, it should be noted that there are many instances where an event may be missing as a result of non-suspicious VM host computer activity.

The log auditor infers VM events to describe an action by or on an object with associated VM temporal data. These inferred VM events are combined with VM events recorded in the hypervisor kernel system logs in order to provide as complete a timeline as possible. In the experiments on computers running VMware sessions on Microsoft Windows, our software prototype inferred many VM events that occurred prior to the enabling of many logging options. There were very few recorded VM events from that early time period in the VM's computer history, and thus these inferred VM events were out-of-context. Such inferred VM events may appear to have occurred outside of the VM user sessions, or in an otherwise inconsistent fashion, however, the absence of complete information must obviously be considered in the VM investigator's assessment as to whether or not the event is suspicious.

This scenario is an example of how the normal configuration of the VM computer system may make an event seem inconsistent.

4 VM Log Auditor Software

A summary of the VM Log Auditor architectural components are provided in Fig. 1. As a VM log inconsistency checking profile tool it maps all hypervisor events, as they are extracted from the kernel logs. It uses a set of shell script parsers to provide an associative and transformational mapping [17,18] of these logs into a normalized database with a set of discovered and inferred VM event tables. The prototype software examines the suspect target VM host file system (which is mounted read-only) and enumerates the set of VM objects of applications, files, and users of the target VMware essx3i computer system. We achieve this by performing a Storage Area Network (SAN) disk image of the suspected VM host to our evidence server in our test bed. The parsed hypervisor log events are described as the set of recorded VM events (***EvtR***) in our Oracle database. Finally, a set of events are inferred from the temporal data associated with each log file which may be as a result of modification, access or creation (MAC) events which have occurred. These events are the inferred VM events (***EvtI***), and are saved in a separate table in the database called Inferred VM Events. It is useful to note that the inferred VM events ***EvtI*** is particularly vulnerable to inconsistency or incompleteness of data obtained from the VM host file system. The two sets ***EvtR*** and ***EvtI*** do not exhaustively describe the complete history of the VM hosted computer system. Hence where an event is not logged or it cannot be inferred we described this as an unknown or missing event. It stands to reason that the more missing or unknown events the less complete the

timeline of the suspect VM host computer system being investigated. Due space limitation we omit the formal completeness proof. After conducting this automated process, the software prototype provides a basic interface for the purpose of detecting temporal inconsistency in a given timeline.

The detection techniques described in this paper match the VM events in a timeline against the events in each rule being tested. Programmatically, every rule is implemented by a C# object, and every event is implemented by a C# object. Rule objects have two event objects as fields, one called $vmevt_a$ and another called $vmevt_b$. The objects $vmevt_a$ and $vmevt_b$ are archetype VM events, against which known VM events are compared. A known VM event is compared against the archetypes on the basis of the fields of each. The fields of the archetype events can have a specific value, or be null. If the archetype has a specific value for a particular field, then any known VM event that matches the archetype must have the same value. If the archetype has a null value for a particular field, it can match any value for the known VM event's corresponding field.

The rule object can also be set to match subject and target fields, that is to say, to require that both matching VM events have the same subject or target field. The rule can also specify that the subject field of one event is the target of the other event, or vice versa. This allows for the definition of generic rules.

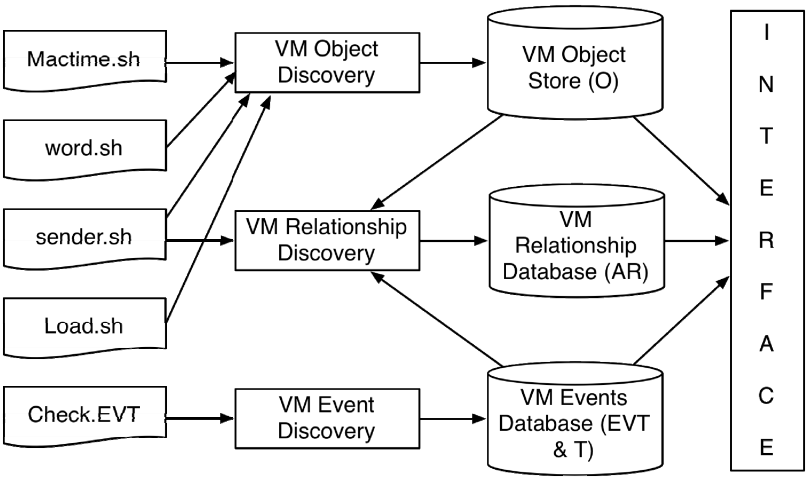


Fig. 1. Architectural components of the VM log auditor prototype

In the object that represented this rule, $vmevt_a$ would represent the "logon" VM event, and $vmevt_b$ would represent the "modified" VM event. A Boolean field of the rule object would be set to true to indicate that the subject of each VM event had to be the same object. Given this, the values of the fields of the objects $vmevt_a$ and $vmevt_b$ would be as follows:

$$vmevt_a = \{\text{null, null, s, logon, success}\}$$
$$vmevt_b = \{\text{null, null, null, modified, success}\}:$$

The prototype log auditor software does not yet implement the concept of a user VM session. A logon or logoff VM event is treated the same as any other event. This means that the user needs to specify which events are to be treated as the beginning and end of the user session timeline. In order to check timelines of a computer system's complete history, the prototype software would need to have a concept of user session built into it. This is an item of future work.

5 Rule Base for Experiments

The VM log auditor software prototype incorporates a small set of rules to check for VM temporal inconsistency. It provides a backend functionality that allows the user to specify a timeline to be checked for inconsistency. It then checks that timeline against the rule base. The rules built into the prototype software for the purposes of these experiments use the following algorithm.

vmevtA = (null, null, **s**, "logon", "success")
vmevtB = (null, null, null, "modified", "success")
rule = **vmevtA** *happened-before* **vmevtB**
where field 2 of **vmevtA** == **x** and where field 2 of **vmevtB** == **x**
for each **vmevt** in V**H(x)**
    if **vmevt** = (*, **x**, **s**, "logon", "success")
        a = index of **vmevt**
    if **vmevt** = (*, **x**, *, "modified", "success")
        b = index of **vmevt**
next **vmevt**
if **a** > **b** then
    rule has been broken

Fig. 2. VM Inconsistency Algorithm

The data structures in our implementation that represented each of the archetype VM events in the rules base, had null values in place of the fields *x*, y.

5.1 Experimental Setup

In order to obtain data for these experiments, we employed a suspect test VMWare essx3i private network hosted computer running on Windows 7 within the University's local area network. This deployment represents our private cloud test environment for the course of the timeline experiments carried out. All system logging options were turned on in order to give us as complete a set of hypervisor event logs which are all stored in a comma delimited format (.csv) on the host oracle databases. We logged onto the VM test host computer twice for the purpose of generating two different VM user sessions: the first, an "innocent" user session, and

the second, a user session in which a document was created with misleading authorship information. The details of these two sessions are described below.

We also tinkered with the detection outcomes of meddling with the hypervisor logs. For this purpose, a copy of the case file and database about the suspicious test VM host computer system inspected by the log auditor tool was provided, and then manually modified the database table containing the discovered VM events. As these discovered VM events are derived from the VMWare essx3i kernel system event logs, the removal or modification of recorded VM events in the set *EvtR* effectively simulates the removal or modification of VM event records in the same. The investigator removed the log-on/log-off VM events from the first user session, and modified the timestamps of these events on the second user session so that they would be presented out of their real sequence if ordered by timestamp.

5.2 Evaluation of VM Detection Techniques

This section describes each of the VM timelines examined in each these experiments and provides the results of the log auditor's analysis of inconsistency. There are four VM log timelines (two unmodified, and two modified) that correspond directly to user sessions. Each of the timelines is a combination of the VM inferred events and the recorded VM events in the history of the VM hosted computer system between two boundary events, ordered by timestamp. Due to space limitations the detailed timeline trace tables that capture the recorded and inferred events have been omitted.

Timeline A - Normal VM User Session

Timeline A was a normal user VM session during which a text document was created. The user "thorpe" logged into the VM host computer system at 9:48 pm on 12 October 2011, and created the file "vmsyslog.doc" at 9:51pm. The VM user then browsed the Internet for a few minutes and logged off at 10:00 pm. Nothing suspicious happened in the user VM session. The timeline consisted of all of the events that took place during the user VM session, both recorded and inferred. Our software inserted these VM events into its Oracle 11g VM event database during its automated examination of the target system.

Most events in timeline A were discovered events (i.e. discovered in the VMWare essx3i hypervisor kernel event logs running under Windows 7), however, the events with "CREATED", "MODIFIED" or "OPENED" as their actions were inferred events (i.e. inferred on the basis of an object, its relationships, or other information about the object).

Timeline B - Deliberate Misattribution of Authorship

Timeline B represents a user VM session during which the user created a text document with misleading authorship information, in an effort to shift responsibility for that document to an innocent third party. The user "VMuser" logged into the computer system at 9:51pm on 13 October 2011, and at 9:55pm a Word document was created with "anoynmous " as the listed author. The user "VMuser " then logged off.

Timeline B was analyzed for inconsistency with our prototype software. The events are all related to the authorship of the word document entitled "WORDOC letter from anonymous". The anonymous user was not logged in at the time the text document was created, and yet the author field listed "anonymous" as the document's author.

There are two sets of "CREATED" events for both the suspect Word document and its template. Hence "anonymous" could not have been the author of the text document. This is so because there are two sources of information that lead the log auditor inferring such an event. The earlier timestamp is obtained from the text document's metadata, and represents the time at which the document was first created. The later timestamp is obtained from the target VM host computer's file system, and is the time at which the document was first saved as a file on the host VM physical disk. Both sets of "CREATED" VM events derive their subject field from the same source, the Word document's author field.

Timeline C - VM user Session with Logon/Logoff Events Deleted

Timeline C was derived from timeline A. The recorded and inferred VM events in the prototype's events database were copied and manually modified. The resulting timeline, timeline C, was identical to timeline A without the logon/ logoff VM events. The removal of these two discovered VM events left user activity outside of a logon/logoff-bound VM user session. This demonstrates that removing VM user session information from the hypervisor event log will draw attention to the inferred VM events that took place during the session.

Timeline D - With VM user Modified Timestamps

Timeline D was derived from timeline A, with the timestamp of the user's logoff VM event deliberately modified so as to appear to have taken place prior to the creation of the text document.

The event was listed as breaking three rules, all of which assert that if a file is modified, accessed or created, it must be modified, accessed or created prior to the user logging out of the VM host computer system. The results of the analysis of timeline D were just as expected.

The detection of this VM event demonstrates the suitability of this approach to detecting events whose timestamps are modified.

6 Constructing Consistent Timelines

The temporal inconsistencies can be handled by the creation of a consistent VM timeline. A consistent VM timeline, in the context of VM computer profiling, is defined as a sequence of VM events ordered by physical time at which they occurred with no obviously missing VM events. The physical time at which the event occurred may or may not correspond to the VM hosted computer generated timestamp of a VM event, which may be missing from the sets ***EVTR*** and ***EVTI***, but which are detected using the techniques that establish VM relationships via hypervisor log object

profiles. ***EVTM*** is the set of all of the missing VM events detected on the basis of a precondition rule. The sequence ***EVTC*** is a sequence over ***EVT*** ordered into a consistent VM timeline. This section describes a technique for constructing such a timeline.

There are some VM events, especially members of ***EVTM***, for which there is no timestamp. There are other VM events for which there is a timestamp, but whose timestamp is provably incorrect (as determined by detecting the out of sequence experiments). Gladyshev and Patel describe the process of determining the time at which a given event takes place by bounding the event's time [4], and we adopt this approach for our VM log auditor tool. The upper and lower bounds for the time of an event can be determined if the VM event must have occurred between two other VM events. The range between these bounds, i.e. the time interval Δt, is the range of possible times at which the VM event could have occurred. The range can be further narrowed if it is known that there is a minimal delay d , which applies to a particular *happened-before* relation [4, 5]. If it is known that there is at least a ten seconds greater than the time of the first event. The range of possible times at which a VM event $vmevt_b$ might have occurred can be calculated.

This approach can only provide a range of times in lieu of a missing or inaccurate timestamp, but such a range is the best possible indication of when the event occurred. As it is impossible to obtain perfect timestamps for every VM event in the history of the VM computer system, the sequence ***EVTC*** cannot be ordered on the basis of the available timestamps. The available timestamps will not be precise enough for ordering the events in and themselves, although they might be useful in determining some other basis for ordering events.

Instead of timestamps, the use of a Lamport logical clock [7] is proposed to provide the basis of ordering the consistent timeline ***EVTC***. The timestamp (or time interval in the case of VM events with indeterminate time) of a VM event will be used as a variable in the clock, but it will be the clock and not the timestamp which will be used as the basis for ordering VM hosted events.

The VM hosted computer BIOS clock C is defined to be a function which assigns a number to every VM event in the consistent timeline ***EVTC***. The number produced by C has no bearing on physical time, but each VM event has a timestamp t for a time interval Δt which can be used to determine the physical time of the VM event. The number produced by C must be lower for VM events which occurred earlier in the VM history of the computer system than the number produced for VM events which occurred later. This will permit events to be sorted by the number produced by the clock C on the ascending order basis.

Given a complete set of rules to detect inconsistent and missing VM events, the number of unknown VM events in the VM computer system's history can be minimized. The proportion of the set of all of the VM events that occurred in the VM history of the VM hosted computer system ***EVT*** that is known can be maximized. A VM event is a known event if it is an element of the sets ***EVTR***, ***EVTI***, or ***EVTM***. Each of the known events will have an associated timestamp, or, in the case of VM events with no timestamps or with provably incorrect timestamps, and the narrowest possible time interval during which the event could have occurred. The clock function

C will combine the timestamp or time interval for each known event with the rules relating that event to the other knowable VM events, and produce a number (i.e. Lamport timestamp) according to which the VM event may be sorted into the consistent timeline ***EVTC***. In the case of potentially concurrent events (which have no *happened-before* relations to other VM events) whose timestamps or time intervals are inconclusive relative to other VM events, the number produced by *C* will be identical. In such cases, some arbitrary mechanism can be used to sort VM events into the consistent timeline ***EVTC***. Once completed, the consistent timeline ***EVTC*** will represent the best sequential ordering of known events that make up the associated VM history profile.

This permits an investigator to view both a consistent timeline, and the original inconsistent timeline along with the reasoning as to why the original hypervisor temporal log data was inconsistent. Then the investigator could then make an informed judgment about the cause of the inconsistencies in the VM computer system timeline.

7 Discussion of Results

The results of the experiments demonstrate that automatically detecting temporal inconsistency in VM hosted computer activity timelines constructed from realistic data is possible using our tool. These experiments applied a simple rule set to a VM hosted computer system's activity timeline, and the results demonstrate that inconsistency can be detected in several basic scenarios. The *happened-before* relation and the precondition predicate can be used together to construct effective rules to draw an investigator's attention to suspicious VM events. Timeline B demonstrated that such rules can be applied to detect an event (in this case, the creation of a document) initiated by a different user than first suggested by the VM file system.

Timeline C showed that the deletion of a hypervisor kernel system log set of entries pertaining to important VM events can be detected. If the deleted events are preconditions for other events, which are recorded or inferred, then they can be detected. Timeline D demonstrated that, by applying a rational set of rules in an automated analysis of a timeline, VM events can be detected that should have occurred in another sequence than their timestamps suggest.

The experiment's use of data from a VM hosted computer system demonstrated that this approach to detecting temporal inconsistency on VM log data is robust enough to be tested in real cases. The ideal next step will be to perform experiments with the log auditor using large scale live incident case data, which will test the robustness and suitability of the approach with respect to real investigations.

The noise in real VM event data is a lesser problem to the VM log auditing tool than it is to a human investigator. By distilling VM event records down to the most important fields that are common to most events, our approach is likely to reduce the complexity and heterogeneity of the various types of VM events.

8 Future Work

We plan to improve the log auditor so that the software can automatically detect user sessions given our ongoing work [16]. At the moment, the prototype software requires the user to specify the bounds (i.e. start and finish) of a user VM session before it is able to check the timeline of that VM session for internal consistency.

We would like to extend the VM log auditor process and software to construct consistency-check timelines of VM hosted computers running non-Windows operating systems.

The basic model presented in this paper doesn't detect hidden nested states that may exist on the VM host as extracted from the hypervisor logs particularly as a remote or distributed logging service. We plan to extend the existing model to now explore and formalize this concurrency problem.

9 Related Work

Although there has not been any formal accepted definition for cloud forensics, its fundamentals are still entrenched in a view that the data provided as case evidence has to be court admissible. The provision of a time-lining technique together with the practical approaches for gathering and inferring VM events comprise a technique for tracing the history of the VM hosted computer system as possible source of such potential forensic evidence is motivated by prior work [20, 21, 22, 25]. In this paper, we attempt to improve on the state of the art by providing a forensic platform that transparently and distinctively monitors and records data access events using the hypervisor kernel event logs. This work adopts a static state snapshot log approach to support post incident off line forensic investigations. Our work complements the live trace analysis and VM introspection methods [21, 22, 25] and the static snapshot finite state hypothesis computational models [5, 6, 11].

10 Conclusion

This work has produced a tool, implementing techniques for detecting contradictory and missing VM events in the history of the computer system. The experiments with this software demonstrate that the techniques that have been proposed can be used successfully to detect temporal inconsistencies in a VM computer activity timeline. The automatic detection of inconsistencies that might indicate deliberate tampering could assist a human investigator in a subsequent manual examination of the VM hosted system running within the data center.

References

1. Rodgers, M.: The role of criminal profiling in the computer forensics process. Computers & Security 22(4), 292–298 (2003)

2. Rodgers, M., Goubalt–Larrecq, J.: Log auditing through model checking. In: Proceedings of the 14th IEEE Computer Security Foundations Workshop, Cape Breton, Nova Scotia (June 2001)
3. Boyd, C., Forster, P.: Time and date issues in forensic computing a case study. Digital Investigation 1(1), 18–23 (2004)
4. Buchholz, F., Tjaden, B.: A brief study of time. In: Proceedings of the 7th Digital Forensics Workshop, Pittsburg, Pennsylvania, USA (August 2007)
5. Fidge, C.: Logical time in distributed computing systems. Computer 24(1), 28–33 (1991)
6. Gladyshev, P., Patel, A.: Formalizing event time bounding in digital investigations. International Journal of Digital Evidence 4(2), 1–14 (2005)
7. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Communications of the ACM 21(1), 558–565 (1978)
8. Marrington, A., Mohay, G., Clark, A., Morarji, H.: Event-based computer profiling for the forensic reconstruction of computer activity. In: Proceedings of the AusCERT Asia Pacific Information Technology Security Conference, Gold Coast, Australia (May 2007)
9. Marrington, A., Mohay, G., Morarji, H., Clark, A.: A Model for Computer Profiling. In: Proceedings of the 5th International Workshop on Digital Forensics at the International Conference on Availability, Reliability and Security, Krakow, Poland (February 2010)
10. Nolan, R., O'Sullivan, C., Branson, J., Waits, C.: First responder's guide to computer forensics. Software Engineering Institute, Carnegie Mellon University, Pittsburg, USA (May 2005)
11. Schatz, B., Mohay, G., Clark, A.: A correlation method for establishing provenance of timestamps in digital evidence. In: Proceedings of the 6th Annual Digital Forensic Research Workshop, West Lafayette, Indiana, USA (August 2006)
12. Willassen, S.Y.: Hypothesis-based investigation of digital timestamps. Advances in Digital Forensics IV 285(1), 75–86 (2008)
13. Willassen, S.Y.: Timestamp evidence correlation by model based clock hypothesis testing. In: Proceedings of the 1st International Conference on Forensic Applications and Techniques in Telecommunications, Information, and Multimedia and Workshop, Adelaide, Australia (January 2008)
14. Willassen, S.Y.: A model based approach to timestamp evidence interpretation. International Journal of Digital Crime and Forensics 1(2), 1–12 (2009)
15. Thorpe, S., Ray, I., Grandison, T.: A Formal Temporal Log Model for the synchronized Virtual Machine Environment. Journal of Information Assurance and Security 6(5), 398–406 (2011)
16. Thorpe, S., Ray, I., Barbir, A., Grandison, T.: Towards a Formal Parameterized Context for a Cloud Computing Forensic Database. In: Proceedings of the 3rd Digital Forensics and Cybercrime Conference, Dublin, Ireland (October 2011)
17. Thorpe, S., Ray, I., Grandison, T.: Associative Mapping Techniques for the synchronized virtual machine environment. In: Proceedings of the 4th Computational Intelligence in Security for Information Systems Conference, Torremolinos, Spain (June 2011)
18. Thorpe, S., Ray, I., Grandison, T.: Enforcing Data Quality Rules for the synchronized virtual machine environment. In: Proceedings of the 4th Computational Intelligence in Security for Information Systems Conference, Torremolinos, Spain (June 2011)
19. Thorpe, S.: PhD Thesis - The Theory of a Cloud Computing Digital Investigation using the Hypervisor kernel logs, University of Technology Jamaica (February 2013)
20. Thorpe, S.: A Virtual Machine History Model Framework for a Data Cloud Investigation. Journal of Convergence 3(4), 9–14 (2012)

21. Srinivas, K., Snow, K., Monrose, F.: Trail of Bytes: Efficient support for Forensic Analysis. In: Proceedings of the ACM Conference on Communication Security, Chicago, Illinois, USA (October 2010)
22. Gidwani, T., Argano, M., Yan, W., Issa, F.: A Comprehensive Survey of Event Analytics. International Journal of Digital Crime and Forensics 4(3), 33–46 (2012)
23. Thorpe, S., Ray, I., Grandison, T., Barbir, A.: A Model for Compiling Truthful Forensic Evidence from the Log Cloud Hypervisor Databases. In: Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC), Work in Progress Session, Orlando, USA (December 2012)
24. Thorpe, S., Ray, I., Grandison, T., Barbir, A.: Log Audit Explanation Templates with Private Data Clouds. In: Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC), Work in Progress Session, Orlando, USA (December 2012)
25. Pauw, W., Heisig, S.: Visual and algorithmic tooling for system trace analysis: A case study. ACMSIGOPS Operating System Review 44(1), 97–102 (2010)

TerraCheck: Verification of Dedicated Cloud Storage

Zhan Wang[1,2], Kun Sun[2], Sushil Jajodia[2], and Jiwu Jing[1]

[1] State Key Laboratory of Information Security
Institute of Information Engineering
Chinese Academy of Sciences, Beijing 100093, China
{zwang,jing}@lois.cn
[2] Center for Secure Information Systems
George Mason University, Fairfax, VA 22030, USA
{ksun3,jajodia}@gmu.edu

Abstract. When hardware resources are shared between mutually distrustful tenants in the cloud, it may cause information leakage and bring difficulties to regulatory control. To address these concerns, cloud providers are starting to offer hardware resources dedicated to a single user. Cloud users have to pay more for such dedicated tenancy; however, they may not be able to detect the unexpected misuse of their dedicated storage due to the abstraction layer of the cloud. In this paper, we propose *TerraCheck* to help cloud users verify if their dedicated storage devices have been misused to store other users' data. *TerraCheck* detects the malicious occupation of the dedicated device by monitoring the change of the shadow data that are residual bits intentionally left on the disk and are invisible by the file system. When the cloud providers share the dedicated disk with other users, such misuses can be detected since the shadow data will be overwritten and become irretrievable. We describe the theoretical framework of *TerraCheck* and show experimentally that *TerraCheck* works well in practice.

Keywords: Dedicated Storage, Cloud Security, Verification.

1 Introduction

Cloud service significantly reduces costs by multiplexing hardware resources among users [12]. The co-resident data belonging to different users may lead to information leakage, which has become a major security concern for cloud users. For instance, a malicious VM is capable of retrieving the encryption keys [22] from a victim VM hosted on the same physical machine. Sensitive information can be compromised through the covert communication channels based on the shared CPU cache [20], memory bus [19], hard disks [15,18] and so on in the cloud.

Cloud providers are starting to offer physically isolated resources in order to lower the entry barrier of cloud adoption for processing sensitive data. For instance, in Amazon cloud [1], *Dedicated Instances* are a form of EC2 instances

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 113–127, 2013.

launched within the Amazon Virtual Private Cloud (Amazon VPC) that runs on hardware dedicated to a single customer. A dedicated instance ensures that the resources, such as CPU, memory, disk storage and network, are isolated physically at the hardware level. Unsurprisingly, the cloud users have to pay more for the dedicated resources than the regular ones.

Although the dedication property is guaranteed by the Service Level Agreement (SLA), a misbehaved cloud provider may fail to meet the isolation requirement due to either accidental configuration error or intentionally reassigning the unallocated resources to other users. As a consequence, the dedicated resource, for example the storage device, will store the data belonging to unexpected users and cause information leakage. Because the cloud users usually can only see a logical view of their resources due to the abstraction layer or the business model of cloud computing [11], they may not be aware of or not be able to detect the violation of the desired dedicated configuration before the security breaches occur.

In this paper, we propose *TerraCheck* to help cloud users verify if the unallocated disk space has been occupied by undesired users without the cooperation of the cloud provider. We assume that the cloud providers are *honest-but-greedy*, i.e., trustworthy for managing user's data without violating the data privacy but *greedy* for allocating the storage resources not being in use by the dedicated user to other tenants. To detect the greedy allocation, *TerraCheck* places shadow data on the unallocated disk space and verifies the dedication by detecting the change of the shadow information.

Shadow data are portions of the residue left behind when files are deleted from the disk. As such, this data cannot be accessed directly by the file system, but can be recovered using forensic techniques. We group the set of residual bits related to the same original file as "shadow chunk". We record the hash value and physical disk address of each shadow chunk as verification metadata. To verify the integrity of each shadow chunk, we utilize disk forensics techniques to retrieve shadow chunks according to the prior recorded disk addresses. If the shadow chunks cannot be recovered entirely, it indicates that the unallocated disk space has been overwritten and the dedication property is violated.

Our shadow chunk method has two advantages, comparing to simply stuffing the unallocated disk space with void files. First, it makes the cheating behavior of the *honest-but-greedy* cloud provider very costly. The retrieval of the shadow chunks relies on the physical disk address of each chunk. If the misbehaved cloud providers move the shadow data to some non-dedicated devices and make the shadow data still retrievable, they must map the prior recorded disk address to the addresses of the new device. Instead, accessing files relies on the file system and can be redirected to another device with less effort. Second, shadow data will not affect the normal use of the dedicated device. The attested disk area filled by the shadow chunks remains available for allocation in the view of the file system. However, if the attested disk area is filled by files, it cannot be occupied by the dedicated user immediately.

We present two schemes for verifying the dedication property of cloud storage. The basic *TerraCheck* scheme can detect the unexpected occupation of a dedicated storage device with high accuracy by checking the retrievability of every chunk. With sampling, our advanced probabilistic *TerraCheck* scheme can discover 10% unexpected occupation of the dedicated storage device with 95% probability, by randomly challenging 29 chunks. Therefore, a smaller chunk achieves low computational cost, but results in the large storage of metadata. Furthermore, with the help of Bloom filter with 1% false positive rate, the size of verification metadata can be reduced 5.5 times.

The rest of the paper is organized as follows. In Section 2, we describe the threat model and general assumptions. Section 3 presents the requirements and operations of the dedication verification. Section 4 describes both the basic and advanced probabilistic *TerraCheck* schemes. Section 5 implements two schemes and evaluates both the computational and storage costs. Section 6 overviews the related work. Section 7 concludes this paper.

2 Threat Model and Assumptions

The dedication property of cloud storage is guaranteed by the terms in SLA. However, a misbehaved cloud provider may fail to meet such dedication requirement due to either accidental configuration errors or intentionally being greedy with the unallocated storage resources: First, configuration error may allocate dedicated storage space to undesired tenants. For instance, in Amazon dedicated instance, the dedication property is enabled by the "Dedicated" attribute configured at the launch time. The "Dedicated" attribute may be silently disabled (e.g., for software update, server migration or testing). Second, a cloud provider may intentionally place the non-frequently accessed data, such as archive data, to the unoccupied disk space where it is supposed to belong to one specific customer.

We consider the misbehaved cloud providers as *honest-but-greedy*. *Honest* means that the cloud providers are not motivated to corrupt user's data or violate the data privacy with respect to the business reputation. However, the cloud providers may be *greedy* for allocating the storage not in used by the dedicated user to other tenants. Although the *honest-but-greedy* cloud providers are only interested in the large amount of unused disk space belonging to a dedicated user, they cannot control the behavior of the co-resident tenants once the cloud provider accidentally allocates the unoccupied space to another tenant. Co-resident tenants may threaten the security and privacy of the existing user data, such as exploiting covert channels to retrieve encryption key [22] and other sensitive information [15] or violating the access control policy [18].

We also consider the cloud providers are economically rational. Misbehaved cloud providers will not defeat our verification mechanism by paying higher storage overhead. For example, the cloud provider can intecept the write call or monitor the process of placing the shadow data. However, by doing these, the cloud provider has to store the shadow data somewhere else in order to be able

to response the verification challenge correctly and occupy the dedicated storage at the same time.

We assume the usage of the dedicated storage is well-planned by the user. For example, the user allocates a determined amount of dedicated disk space to each VM. This is a common practice [11] of resource management in the cloud. When the user launches a small number of VMs, only part of the dedicated storage is allocated. The rest of the dedicated storage should be protected from being exploited by other users due to both the security and performance reasons. We refer this part of the disk space as *attested area*. The disk space being in use by the dedicated user is called *occupied area*. Additionally, the attested area may scale up and down based on the occupation of the dedicated disk. *TerraCheck* requires a small amount of trusted disk space for storing verification metadata on the *occupied area*. We assume the *occupied area* is trusted, since an *honest-but-greedy* cloud provider is trustworthy for managing user data.

3 System Model

In TerraCheck, both *occupied* and *attested* disk spaces are assigned to the user by the cloud provider and under the management of file system. Occupied area is the disk space which has stored user's data; attested area is the empty disk space that is available to be allocated by the user. TerraCheck only focuses on the verification of dedicated storage assigned by the cloud provider rather than the physical disk, since the users of dedicated storage cannot control the disk space that doesn't belong to them. The verification of dedicated storage that solely occupies a physical disk is another research topic, which can be addressed by existing co-residency checking techniques [18,17].

We first formalize the model of *TerraCheck*. Suppose a user C pays and possesses a dedicated disk with the capacity of s in the cloud. The dedicated disk is divided into two areas as shown in Fig. 1. The *occupied area* with the capacity of s_a disk space has been allocated by C for storing the data associated with running VMs or as general purpose storage. We consider occupied area is trusted by C to execute the *TerraCheck* and store the verification metadata. The *attested area* with the capacity of s_u disk space remains unallocated where $s_u = s - s_a$. Attested area is the verification target of *TerraCheck*. When C needs more disk space by increasing the size of occupied area, the size of attested area will shrink accordingly. The goal of *TerraCheck* is to verify if the attested area has been maliciously taken by other users or the cloud provider.

TerraCheck consists of four major procedures, as shown in Fig. 1. First, it places shadow chunks on the attested area of the target disk. The shadow chunks are deleted files which cannot be accessed from the file system directly by system calls. Instead, shadow chunks can be recovered by disk forensics technique as long as they have not been overwritten. Second, it generates metadata, such as the hash value of the shadow chunks, for monitoring the alternation of shadow chunks. The metadata are stored on the occupied area where it has been allocated for storing the data associated with running VMs or as general purpose

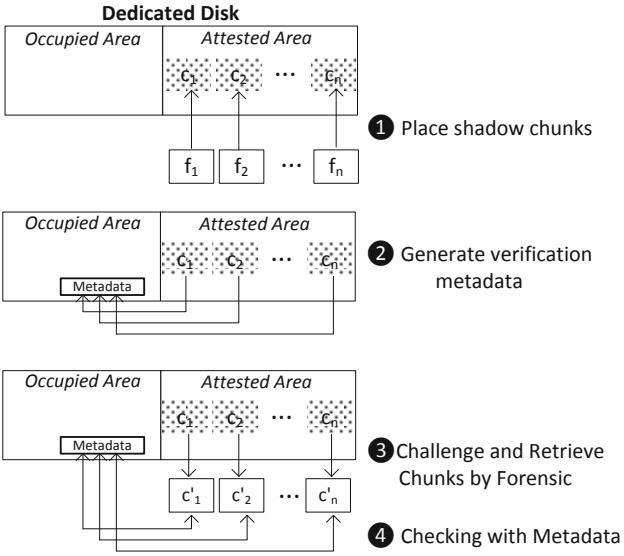


Fig. 1. Overview of *TerraCheck*

storage. Third, *TerraCheck* challenges the shadow chunks by using disk forensic techniques to recover them. Lastly, it compares the forensics results with the verification metadata. If any one of the shadow chunks has been altered and cannot be recovered, a violation of dedication property is detected.

3.1 Verification Requirements

A solution for verifying the dedicated storage should satisfy the following technical requirements.

- *Accuracy.* The verification mechanism should ensure the users to trust the result of the verification. When the misbehaved cloud providers break the dedication property by reassigning the dedicated storage to undesired tenants, the user should be able to detect such violation with a high probability.
- *Efficiency.* The verification procedure should be fast, without obviously interrupting the disk activities against the allocated part of the disk. Moreover, The metadata used for verification should be small; otherwise, it is unacceptable to use the same amount or more disk space to store the original shadow data on the local disk. When the dedicated user occupies or releases more disk space, for example, for running more VMs or shutting down existing VMs, the disk area to be attested varies. Every time the customer needs to scale the disk space up or down, the affected shadow chunks should be as few as possible.

Table 1. Summary of Operation Parameters

Variable	Meaning
C	The cloud user who possesses the dedicated device and executes dedication verification
n	The number of shadow chunks placed on attested disk area
l_k	Length of each shadow chunk
t_h	Header tag of each chunk
t_f	Footer tag of each chunk
K	The set of shadow chunks
s_u	Size of unallocated disk space
id_{k_i}	ID of shadow chunk i
F	The set of files for generating shadow chunks
img_{AA}	Disk image of attested area
$meta_{DB}$	File for storing verification metadata
b_i	Starting disk address of chunk i on attested area
e_i	Ending disk address of chunk i on attested area
id_{AR_x}	ID of attested region x
$meta_{FILTER}$	File for storing Bloom filter

3.2 System Operations

TerraCheck consists of five basic operations. *ChunkGen* generates the shadow chunks and places them on the *attested area*. *MetaGen* generates the verification metadata and stores them on the *occupied area*. *ChalGen* generates the information of challenged chunks. *Retrieve* executes the forensics of challenged chunks and calculates their hash values. $Verify$ operation compares the result of *Retrieve* with the verification metadata recorded in *MetaGen* and makes the decision of the dedication verification. Table 1 summarizes all the variables used in this paper.

- ***ChunkGen***$(n,\ l_k,\ t_h,\ t_f) \rightarrow K = \{k_1, k_2, ..., k_n\}$**:** *TerraCheck* fills *attested area* with a set of chunks $K = \{k_1, k_2, ..., k_n\}$ and $n * l_k = s_u$. Each chunk k_i has a header tag t_h and a footer tag t_f to represent the start and the end of a chunk, respectively. The total length of the header and the footer $l_{t_h} + l_{t_f}$ is less than l_k. This algorithm takes the number of chunks, the length of each chunk, the header t_h, the footer t_f as inputs and generates n temporary files $F = \{f_1, f_2, ..., f_n\}$ first. Every file f_i in F starts with t_h, ends with t_f and the rest of it is filled by random bits. Every file f_i has the same length as l_k. All the files in F are stored on attested area and then deleted from the file system. The bits left on attested area associated with each file f_i are the set of chunks $K = \{k_1, k_2, ..., k_n\}$. Each chunk contains three parts - the header, the footer, and a random body.
- ***MetaGen***$(n,\ t_h,\ t_f,\ img_{AA},\ h) \rightarrow \{meta_{DB},\ \perp\}$**:** It takes the number of chunks, the header, footer tag information, the disk image of *attested area* and a hash function as inputs, returns the verification metadata or abortion. $h : \{0,1\}^* \rightarrow \{0,1\}^m$ denotes a fixed hash function that outputs m bits hash value. The MetaGen algorithm retrieves the chunks from img_{AA} by matching the t_h and t_f and calculates the hash value of each chunk. The results of verification metadata $meta_{DB}$ is stored on occupied area. $meta_{DB} = \{(id_{k_i}, b_i, e_i, h(k_i)) | i \in \{1, 2...n\}, k_i \in K\}$ lists the ID of a chunk

and the boundary of each chunk on the disk, such as the start block number b_i and the end block number e_i of chunk k_i, and the hash value of each chunk $h(k_i)$. Each chunk can be retrieved from the raw disk based on the start and end block number without the help of the file system. Let $|meta_{DB}|$ be the number of items in $meta_{DB}$. If $|meta_{DB}| \neq n$, it indicates that some chunks either cannot be recovered from the disk image of attested area or a mismatched header or footer involved among the chunks. In this case, MetaGen fails and outputs abortion symbol $\perp$.

- ***ChalGen***$(meta_{DB}, id_{k_i}) \rightarrow chal$**:** This algorithm generates a challenge *chal* based on $meta_{DB}$ and the ID of the queried chunk. $chal = (id_{k_i}, b_i, e_i, h(k_i)) \in meta_{DB}$ is the chunk to be examined.
- ***Retrieve***$(chal, h) \rightarrow result$**:** It takes the challenge and the hash function as inputs and calculates the hash value after retrieving the chunk based on the information specified in *chal*. It returns the hash value of the chunk in *chal*.
- ***Verify***$(result, chal) \rightarrow$**{"success", "failure"}:** The Verify algorithm takes *result* and *chal* as inputs and compares the hash value in *result* with that in *chal*. If the two hash values match, it outputs "success" and otherwise outputs "failure".

4 *TerraCheck* Schemes

We propose two schemes. The basic *TerraCheck* can accurately verify the violation of the dedication with a high computation and storage overhead. The advanced *TerraCheck* can detect the violation of the dedication with a high probability, while reducing the verification overhead dramatically.

4.1 Basic Scheme

Our goal is to make sure that the attested area hasn't been allocated to other users. Our basic *TerraCheck* scheme consists of four phases.

- **Initial.** In the initial phase, the attested area is filled by all zeros. This operation prevents the existing content on the disk from affecting our placement results.
- **Placement.** We place the shadow chunks on the attested area by using the $ChunckGen$ and $MetaGen$ algorithms. If $MetaGen \rightarrow \perp$, a failure occurs, *TerraCheck* should be restarted from the initial phase. Otherwise, $MetaGen$ generates valid verification metadata $meta_{DB}$.
- **Verification** is a procedure to patrol on the dedicated storage device and collect the evidence for the undesired occupation by calling $Challenge$, $Retrieve$ and $Verify$ algorithms until each shadow chunk placed in the attested area has been checked. The $Verification$ phase would be stopped once $Verify$ algorithm returns a "failure" for any chunk. The dedication property is preserved if all the chunks passed the examination.

– **Update** is executed when the size of attested area is subject to change. It is difficult to predict the set of affected chunks since the allocation of disk space depends on the disk scheduling. Therefore, both the shadow chunks and their associated verification metadata become useless and subjects to deletion. The initial phase and placement phase should be restarted with the new attested area.

The basic *TerraCheck* can successfully check the dedication property with high accuracy. If $n - t$ shadow chunks are recoverable, it means that t chunks are altered so that around $t * l_k$ out of s_u disk space has been allocated or corrupted maliciously. Theoretically, we can 100% detect the alternation of any number of chunks. However, the basic *TerraCheck* scheme has two main limitations:

– **Computational Cost.** The verification phase has to read through the whole *attested area* and calculate the hash value for every shadow chunk.
– **Update Operation.** When the size of *attested area* has to be changed, *TerraCheck* should be restarted from the initial phase against the new *attested area*.

4.2 Advanced Scheme

To mitigate the limitations of the basic *TerraCheck* scheme, we propose a probabilistic based *TerraCheck* scheme. To reduce the computational cost, we randomly sample the chunks during the *Verification* procedure. In order to provide efficient update operation, we introduce multiple regions within the attested area, we call them *attested region*. The attested region is the smallest unit for C to scale up the size of the occupied area. For example, C plans to attach a certain size of disk space to a newly launched VM. When the size of the occupied area is shrunk due to the termination of a VM, a new attested region will be created. Each attested region contains multiple shadow chunks. The shadow chunk is the smallest unit for challenge and verification. In addition, we use Bloom filter to reduce the storage for saving the verification metadata.

Attested Region. We introduce attested region for conveniently scaling up and down the size of attested area. The attested area is divided into multiple attested regions. The size of attested region depends on how a user uses the dedicated disk. For example, if it uses the disk as the attached secondary storage for running VMs, and each VM is attached by a fixed amount of disk space, such amount is an optimal size for each attested region. When an attested region should be deleted, the related verification metadata are deleted and excluded from the *TerraCheck* procedure. The attested region can also serve the purpose of preventing the cloud provider from manipulating the allocation status of the dedicated storage. That is, only the user of dedicated storage can extend the size of occupied area by generating more attested region.

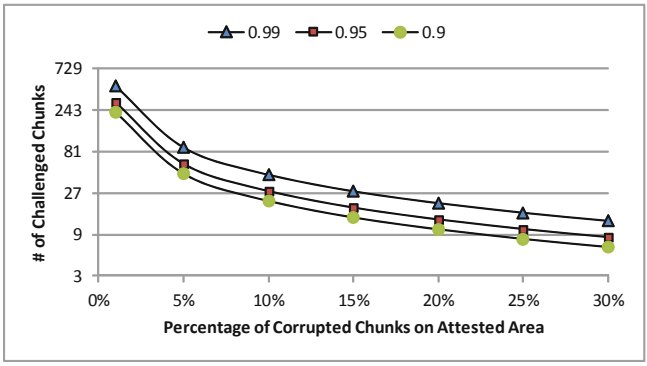


Fig. 2. Probabilistic Framework of Advanced *TerraCheck*

Probabilistic Verification. The sampling would greatly reduce the computational cost, while still achieving a high detection probability. We now analyze the probabilistic guarantees offered by a scheme that supports chunk sampling.

Suppose the client probes p chunks during the *Challenge* phase. Clearly, if the cloud provider destroys chunks other than those probed, the cloud provider will not be caught. Assume now that t chunks are tampered and become unrecoverable, so that at least $s_t = t * l_k$ size of disk space are maliciously allocated. If the total number of chunks is n, the probability that at least one of the probed chunks matches at least one of the tampered chunks is $\rho = 1 - \frac{n-t}{n} \cdot \frac{n-t-1}{n-1}, ..., \cdot \frac{n-p+1-t}{n-p+1}$. Since $\frac{n-t-i}{n-i} \geq \frac{n-t-i-1}{n-i-1}$, it follows that $\rho \geq 1 - (\frac{n-t}{n})^p$.

When t is a fraction of the chunks, user C can detect misbehaviors by asking for a constant amount of chunks, independently on the total number of file blocks. As shown in Fig. 2, if $t = 1\%$ of n, then *TerraCheck* asks for 459 chunks, 300 chunks and 230 chunks in order to achieve the probability of at least 99%, 95% and 90%, respectively. When the number of corrupted chunks goes up to 10% of the total chunks, the violation can be detected with 95% probability, by only challenging 29 chunks. As the number of corrupted chunks increases, the number of chunks required to be checked is decreased. The sampling is overwhelmingly better than scanning all chunks in the basic *TerraCheck* scheme. Therefore, we can challenge a fixed number of chunks to achieve certain accuracy. The size of each chunk will determine the computation cost. When the size of each chunk is small, the overhead for retrieving all challenged chunks from dedicated disk is low.

Advanced Operations. For establishing efficient *TerraCheck*, we need to refine both the $MetaGen$ and $ChalGen$ algorithms.

***MetaGen*(n, t_h, t_f, img_{AA}, h)$\rightarrow${$meta_{DB}$, $\perp$}:** The results of verification metadata $meta_{DB} = \{(id_{AR_x}, id_{k_i}, b_i, e_i, h(k_i))|i \in \{1, 2...n\}, k_i \in K\}$. It lists the ID of the located attested region, the ID of a chunk and the boundary of each chunk on the disk, such as the start block number b_i and the end block

number e_i of chunk k_i, and the hash value of each chunk $h(k_i)$. Each chunk can be retrieved from the raw disk based on the start and end block number and the ID of the attested region without the help of the file system.

ChalGen$(meta_{DB}) \xrightarrow{r} chal$. It randomly generates a challenge *chal* based on $meta_{DB}$. $chal = (id_{AR_r}, id_{k_r}, b_r, e_r, h(k_r)) \in meta_{DB}$ is the chunk to be examined.

Our advanced *TerraCheck* scheme consists of the same phases as the basic *TerraCheck*. Advanced operations will be involved in the related phases, and the update phase should be modified as follows.

- ***Update.*** Since the attested area is further divided into attested regions, when a user needs to extend or shrink the disk space for occupied area, only limited number of attested regions are deleted or added so that the *TerraCheck* against the rest of chunks remains valid. When the occupied area scales up, the metadata related to the erased attested region will be deleted. The rest of metadata are still available for *TerraCheck*.

Reducing Metadata Storage. In the basic *TerraCheck* scheme, the size of $meta_{DB}$ for storing the verification metadata is linear to the number of shadow chunks. The number of chunks could be very large if the user wants to achieve a lower computational cost, as we discussed in the probabilistic verification. In order to reduce the amount of storage for verification metadata in *TerraCheck*, we take advantage of Bloom filter to store the metadata for verification.

Bloom filter [4] is a space-efficient data structure for representing a set in order to support membership queries. Bloom filter is suitable to the place where one might like to keep or send a list for verification, but a complete list requires too much space. We use Bloom filter to represent a set $S = \{x_1, x_2, ..., x_n\}$ of n elements as an array of m counters, initially all set to 0. It uses k independent hash functions $h_1, h_2, ..., h_k$ with range [1, m]. For mathematical convenience, we make the natural assumption that these hash functions map each item in the universe to a random number over the range $\{1, ..., m\}$. For each element $x \in S$, the bits $h_i(x)$ are set 1 for $1 \leq i \leq k$. A location can be set as 1 multiple times. To check if an item y is a member of S, we check whether all $h_i(y)$ are 1. If not, then clearly y is not a member of S. If all $h_i(y)$ are 1, we assume that y is in S. We know that a Bloom filter may yield a false positive, where it suggests that an element x is in S even though it is not.

The probability of a false positive for an element not in the set, or the false positive rate, can be estimated, given our assumption that hash functions are perfectly random. After all the elements of S are hashed into the Bloom filter, the probability that a specific bit is still 0 is $PR_{zero} = 1 - \frac{1}{m}^{kn} \approx e^{-\frac{kn}{m}}$. The probability of a false positive is $(1 - PR_{zero})^k$. A Bloom filter with an optimal value for the number of hash functions can improve storage efficiency.

We modify our *TerraCheck* model for utilizing Bloom filter to reduce the storage cost of the verification metadata.

- ***BF-MetaGen***$(t_h, t_f, img_{AA}, h) \rightarrow \{meta_{FILTER}, \perp\}$ The algorithm takes the header, footer tag information, the disk image of *attested* area and a hash function as inputs, returns the verification metadata or an abortion. $meta_{FILTER}$ is a Bloom filter which involves the hash value of every shadow chunk.
- ***BF-Verify***$(result, meta_{FILTER}) \rightarrow \{$"success", "failure"$\}$: It takes $result$ and $meta_{FILTER}$ as inputs and checks if the hash value in $result$ is valid and associates with any chunks. If the hash value can be found from $meta_{FILTER}$, the algorithm outputs "success" and otherwise "failure".

5 Implementation and Evaluation

We implement and evaluate both basic *TerraCheck* scheme and advanced *TerraCheck* scheme. All experiments are conducted on a Dell PowerEdge460 server with Intel Core i5 CPU running at 3.10GHz, and with 4096 MB of RAM. The system runs Ubuntu 12.04 (LST) that is configured with Xen Hypervisor. The dedicated storage device is a WestDigital SATA 7200 rpm hard disk with 1TB capacity and 64MB cache. For evaluation purpose, we used SHA-1 as the hash function h. The random values used for challenging the chunks in the advanced *TerraCheck* are generated using the function proposed by Shoup [7]. All data represent the mean of 20 trials.

We implement a large attested area in basic *TerraCheck* and implement an attested region in advanced *TerraCheck* as a logical volume. The occupied area may involve multiple logical volumes. LVM (Logical Volume Management) technology is exploited to automate the update operation when the size of the occupied disk space varies. We rely on the retrievability of the shadow chunks on each logical volume to check the dedication property. We utilize Scalpel [14], which is an open source file recovery utility with an emphasis on speed and memory efficiency, to retrieve the shadow chunks based on their header tag and footer tag. To perform file recovery, Scalpel makes two sequential passes over each disk image. The first pass reads the entire disk image and searches for the headers, footers and a database of the locations of these headers. The second pass retrieves the files from the disk image based on the location information of the header and footer. Scalpel is file system-independent and will carve files from FATx, NTFS, ext2 and ext3, or raw partitions.

We evaluate both the computation overhead and storage cost during each phase of *TerraCheck*and demonstrate the compliance with the requirements of both accuracy and efficiency identified in Section 3.1.

Initial Phase. During the initial phase, the attested area is filled by all zeros. The time for this phase is determined by, and linear to the size of attested area s_u. It takes about 10 seconds for cleaning 1 GB of the attested area. Both basic *TerraCheck* and advanced *TerraCheck* have the same performance at this phase.

Placement Phase. There are two steps for placing the chunks. The first step is to generate and store the chunks to the attested area. The cost of this operation is

Table 2. Time for Retrieving Chunks

Chunk Size	512KB	1MB	2MB	4MB	8MB	16MB
Retrieve Time	13 ms	15 ms	20 ms	29 ms	48 ms	86 ms

determined by the chunk size and the size of the attested area. On our testbed, it takes 12 seconds to store 100MB of shadow chunks. The second step is to generate the metadata. It takes 8.198 seconds for Scalpel to scan 1 GB of the attested area in the first pass and store the location information.

Verification Phase. The basic *TerraCheck* examines all the chunks based on the verification metadata recorded in $meta_{DB}$. Therefore, the time for generating the challenge can be ignored. The advanced *TerraCheck* randomly challenges the chunks. The generation of random number takes less than 0.1 ms. The challenged chunks are retrieved from the *attested area* based on the start and end location recorded as the verification metadata. Therefore, the performance is determined by the disk access time. Tab. 2 shows the disk access time in our experiment.

After retrieving the challenged chunks, *TerraCheck* compares the hash value of the retrieved chunk with the verification information. In basic *TerraCheck*, all the chunks residing on the attested area should be checked, which uses the time for calculating the hash value of all the chunks. The advanced *TerraCheck* scheme randomly challenges the chunks to achieve the detection of undesired disk occupation. We simulate the behaviors that a proportion of attested area is altered. For instance, if a random 1% of an attested area with 10000 chunks are altered, such a situation could be detected with a 90% probability by challenging 217 chunks on average, which is close to the theoretical result.

Update Phase. For the basic *TerraCheck* scheme, the performance of the update is the same as the overhead of executing the initial and placement phases. The performance of the advanced *TerraCheck* scheme depends on the change of the size of the attested area. When the occupied area is extended, the advanced *TerraCheck* scheme only needs to update the $meta_{DB}$ by deleting the items of affected chunks. When the occupied area is shrunk, more attested regions should be created on the attested area. The generation of each attested region takes about 400 ms regardless the size of the attested region. Therefore, TerraCheck scheme can scale with a low overhead when users update the size of attested area frequently.

Reducing Metadata Storage. *apgbmf* [2] is originally used to manage Bloom filter for restricting password generation in APG password generation software [16]. We use *apgbmf* version 2.2.3 as a standalone bloom filter management tool.

We consider each hash value of the shadow chunk as an item of password dictionary in the context of *apgbmf*. We create a Bloom filter for such hash value dictionary. During the verification phase of *TerraCheck*, if a recovered chunk is unaltered, its hash value will pass the Bloom filter, i.e, the hash value is one of the hash values which associates an original shadow chunk with a high probability.

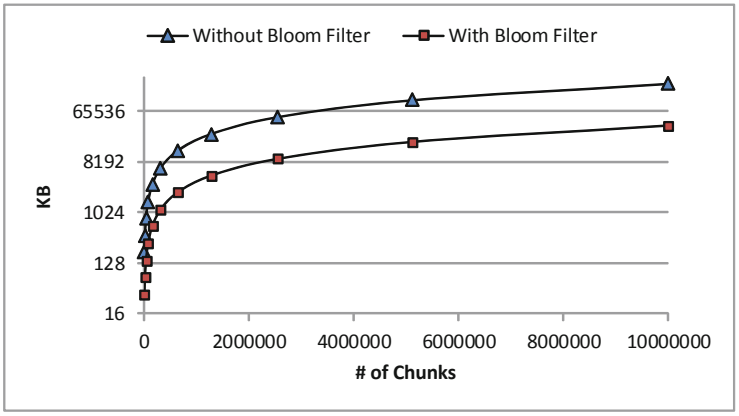


Fig. 3. Comparison of the Storage Cost with/without Bloom Filter (%1 Fault Positive Rate Allowed)

When we allow a 1% fault positive rate, the storage cost with Bloom filter is reduced 5.5 times as shown in Fig. 3. When the number of chunks is more than 10 million, the metadata only requires 36 MB as compared to 200 MB without using Bloom filter.

6 Related Work

Cloud service providers [1,13] are starting to offer physically isolated resources to lower the entry barrier for enterprises to adopt cloud computing and storage. For instance, in Amazon cloud [1], *Dedicated Instances* are a form of EC2 instances launched within the Amazon Virtual Private Cloud, which runs hardware dedicated to a single customer. Some research has been done to guarantee the exclusive occupation of dedicated resources for security reasons. The side channel based on CPU L2 cache has been used to verify the exclusive use of a physical machine [21]. Ristenpart et al. [15] propose to use the existing side channels to verify the co-residency of VMs. [10] allows application designers to build secure applications in the same way as on a dedicated closed platform by using a trusted virtual machine monitor. However, it requires the modification of commercial hypervisor.

Researchers have also investigated techniques to verify various security properties claimed in the SLAs. Dijk et al. [8] prove that the files are stored with encryption at the cloud server side by imposing a resource requirement on the process of translating files from the plain texts to the cipher texts. Proof of Retrievability (PoR) [9] aims to verify if the files are available in the cloud storage at any time. However, PoR cannot verify where the files are located. RAFT [5] can verify that a file is stored with sufficient redundancy by measuring the response time for accessing "well-collected" file blocks. Another work [3] proposes

a mechanism to verify that the cloud storage provider replicates the data in multiple geo-locations, by measuring the network latency. [18] proposes a method to verify the disk storage isolation of conflict-of-interest files so that Chinese Wall security policy [6] can be successfully enforced in cloud storage environment.

7 Conclusion

In this paper, we propose *TerraCheck* to help cloud users verify the exclusive use of their dedicated cloud storage resources. *TerraCheck* places shadow chunks on the dedicated disk and detects the change of the shadow information by taking advantage of disk forensics technique. We further improve the computational efficiency by randomly challenging the chunks and reduce the storage by applying Bloom filter.

Acknowledgement. This material is based upon work supported by the National Science Foundation under grant CT-20013A, by US Army Research Office under MURI grant W911NF-09-1-0525 and DURIP grant W911NF-11-1-0340, and by the Office of Naval Research under grant N0014-11-1-0471.

References

1. Amazon Web Services, `http://aws.amazon.com`
2. APGBFM, `http://linux.die.net/man/1/apgbfm`
3. Benson, K., Dowsley, R., Shacham, H.: Do you know where your cloud files are? In: CCSW, pp. 73–82 (2011)
4. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13(7), 422–426 (1970)
5. Bowers, K.D., van Dijk, M., Juels, A., Oprea, A., Rivest, R.L.: How to tell if your cloud files are vulnerable to drive crashes. In: ACM Conference on Computer and Communications Security, pp. 501–514 (2011)
6. Brewer, D.F.C., Nash, M.J.: The chinese wall security policy. In: IEEE Symposium on Security and Privacy, pp. 206–214 (1989)
7. Dent, A.W.: The Cramer-Shoup encryption scheme is plaintext aware in the standard model. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 289–307. Springer, Heidelberg (2006)
8. Dijk, M.V., Juels, A., Oprea, A., Rivest, R.L., Stefanov, E., Triandopoulos, N.: Hourglass schemes: How to prove that cloud files are encrypted. In: ACM Conference on Computer and Communications Security (2012)
9. Dodis, Y., Vadhan, S., Wichs, D.: Proofs of retrievability via hardness amplification. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 109–127. Springer, Heidelberg (2009)
10. Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: a virtual machine-based platform for trusted computing. SIGOPS Oper. Syst. Rev. 37(5), 193–206 (2003)
11. Jhawar, R., Piuri, V.: Fault tolerance management in iaas clouds. In: Proc. of the 1st IEEE-AESS Conference in Europe about Space and Satellite Telecommunications (ESTEL 2012), Rome, Italy (October 2012)

12. Kurmus, A., Gupta, M., Pletka, R., Cachin, C., Haas, R.: A comparison of secure multi-tenancy architectures for filesystem storage clouds. In: Kon, F., Kermarrec, A.-M. (eds.) Middleware 2011. LNCS, vol. 7049, pp. 471–490. Springer, Heidelberg (2011)
13. Rackspace, `http://www.rackspace.com`
14. Richard III, G.G., Roussev, V.: Scalpel: A frugal, high performance file carver. In: DFRWS (2005)
15. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: ACM Conference on Computer and Communications Security, pp. 199–212 (2009)
16. Spafford, E.: Opus: Preventing weak password choices
17. Varadarajan, V., Kooburat, T., Farley, B., Ristenpart, T., Swift, M.M.: Resource-freeing attacks: Improve your cloud performance (at your neighbor's expense). In: ACM Conference on Computer and Communications Security (2012)
18. Wang, Z., Sun, K., Jajodia, S., Jing, J.: Disk storage isolation and verification in cloud. In: Globecom 2012, Anaheim, CA, USA (2012)
19. Wu, Z., Xu, Z., Wang, H.: Whispers in the hyper-space: High-speed covert channel attacks in the cloud. In: The 21st USENIX Security Symposium (Security 2012) (August 2012)
20. Xu, Y., Bailey, M., Jahanian, F., Joshi, K.R., Hiltunen, M.A., Schlichting, R.D.: An exploration of L2 cache covert channels in virtualized environments. In: CCSW, pp. 29–40 (2011)
21. Zhang, Y., Juels, A., Oprea, A., Reiter, M.K.: Homealone: Co-residency detection in the cloud via side-channel analysis. In: IEEE Symposium on Security and Privacy, pp. 313–328 (2011)
22. Zhang, Y., Juels, A., Reiter, M.K., Ristenpart, T.: Cross-VM side channels and their use to extract private keys. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS, pp. 305–316 (2012)

Fair Private Set Intersection with a Semi-trusted Arbiter

Changyu Dong[1], Liqun Chen[2], Jan Camenisch[3], and Giovanni Russello[4]

[1] Department of Computer and Information Sciences, University of Strathclyde
changyu.dong@strath.ac.uk
[2] Hewlett-Packard Laboratories – Bristol, United Kingdom
liqun.chen@hp.com
[3] IBM Research – Zurich, Switzerland
jca@zurich.ibm.com
[4] Department of Computer Science, University of Auckland
g.russello@aucklanduni.ac.nz

Abstract. A private set intersection (PSI) protocol allows two parties to compute the intersection of their input sets privately. Most of the previous PSI protocols only output the result to one party and the other party gets nothing from running the protocols. However, a mutual PSI protocol in which both parties can get the output is highly desirable in many applications. A major obstacle in designing a mutual PSI protocol is how to ensure *fairness*. In this paper we present the first fair mutual PSI protocol which is efficient and secure. Fairness of the protocol is obtained in an optimistic fashion, i.e. by using an offline third party arbiter. In contrast to many optimistic protocols which require a fully trusted arbiter, in our protocol the arbiter is only required to be semi-trusted, in the sense that we consider it to be a potential threat to both parties' privacy but believe it will follow the protocol. The arbiter can resolve disputes without knowing any private information belongs to the two parties. This feature is appealing for a PSI protocol in which privacy may be of ultimate importance.

1 Introduction

An interesting problem in secure computation is private set intersection (PSI). Namely, how to enable two mutually untrusted parties to compute jointly the intersection of their private input sets. PSI has many potential applications in private data mining, online recommendation services, online dating services, medical databases and so on. There have been many protocols proposed to solve the PSI problem [1–10]. The majority of them are single-output protocols, i.e. only one party obtains the intersection and the other party gets nothing. However, there are many motivating scenarios in which both parties want to know the intersection. Several examples have been given in [6] to demonstrate the need for such *mutual* PSI protocols:

- *Two real estate companies would like to identify customers (e.g., homeowners) who are double-dealing, i.e., have signed exclusive contracts with both companies to assist them in selling their properties.*
- *A government agency needs to make sure that employees of its industrial contractor have no criminal records. Neither the agency nor the contractor are willing to disclose their respective data-sets (list of convicted felons and employees, respectively) but both would like to know the intersection, if any.*

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 128–144, 2013.

A mutual PSI protocol must be fair, i.e. if one party knows the intersection, the other party should also know it. However fairness is hard to achieve in cryptographic protocols (see Section 2 for a brief overview). To efficiently achieve fairness, most fair cryptographic protocols are *optimistic* which requires help from an offline arbiter who is a trusted third party. The arbiter only participates if one party unfairly aborts the protocol and can recover the output from the protocol for the honest party. Incorporating optimistic fairness in PSI protocols is not easy for two reasons: firstly, although there is a generic structure, there is no generic construction for optimistic fair protocols. Secondly, the arbiter usually has to get access to some private information and therefore has to be fully trusted. However, in reality it is hard to find such a fully trusted third party. Think about the examples above: an independent entity, e.g. an auditing service provider, could be well qualified to resolve the disputes, however giving a third party access to private data may raise privacy concerns. We can find more cases in which the two parties may trust a third party for fairly resolving disputes, but may not trust it for privacy.

In this paper, we present the first fair mutual PSI protocol. The protocol has built-in support for optimistic fairness and does not require setup assumptions such as certified input sets. In addition, the third party acting as the arbiter can resolve disputes without knowing the private inputs or the output of the PSI protocol. Hence we can significantly reduce the trust placed on the arbiter. This makes the protocol more flexible in terms of practical usage as any third party can become an arbiter as long as they are believed to be able to correctly carry out instructions.

2 Related Work

Private Set Intersection (PSI) protocols allow two parties, each with a private set, to securely compute the intersection of their sets. It was first introduced by Freedman et al in [1]. Their protocol is based on oblivious polynomial evaluation. Dachman-Soled et al [2], Hazay and Nissim [3] followed the oblivious polynomial evaluation approach and proposed protocols which are more efficient in the presence of malicious adversaries. Hazey and Lindell [4] proposed another approach for PSI which is based on oblivious pseudorandom function evaluation. This approach is further improved by Jarecki and Liu [5]. De Cristofaro et al [6, 7] proposed PSI protocols with linear communication and computational complexities. Huang et al [11] presented a PSI protocol based on garble circuits, and shows in the semi-honest model the protocol can be very efficient. There are also protocols based on commutative encryption [12, 13].

All of the above protocols are single-output, i.e. one party gets the output and the other party gets nothing. This is a traditional way to simplify protocol design in the malicious model because it removes the need for fairness, i.e. how to prevent the adversary from aborting the protocol pre-maturely after obtaining the output (and before the other party obtains it) [14].

Nevertheless, there have been a few mutual PSI protocols which are designed to output the intersection to both parties. Kissner and Song [8] proposed the first mutual PSI protocol. The protocol itself does not guarantee fairness, but relies on the assumption that the homomorphic encryption scheme they use has a fair threshold decryption

protocol. However, unless there is an online trusted third party, it is also non-trivial to achieve fairness in threshold decryption protocols. On the other hand, if an online trust third party is available, the PSI functionality can be trivially computed by giving the input sets to the trusted party. Camenisch and Zaverucha [9] sketched a mutual PSI protocol which requires the input sets to be signed and certified by a trusted party. Their mutual PSI protocol is obtained by weaving two symmetric instances of a single-output PSI protocol with certified input sets. Fairness is obtained by incorporating an optimistic fair exchange scheme. However this protocol does not work in general cases where inputs are not certified because it is hard to force the two parties to use the same inputs in the two instances. Another mutual PSI protocol is proposed by Kim et al [10], but they specifically state that fairness is not considered in their security model.

Fairness is a long discussed topic in cryptographic protocols. Cleve [15] showed that *complete fairness* is impossible in two-party protocols in the malicious model. However, *partial fairness* can be achieved. Partial fairness means that one party can have an unfair advantage, but the advantage is computationally insignificant. Many protocols achieve partial fairness by using the gradual release approach [16–18]. However, this approach is very inefficient in nature. The *Optimistic* approach, which uses an offline trusted third party, has been widely used to obtain fairness efficiently. It is called optimistic because it cannot prevent the unfair behaviour but later the trusted third party can recover the output for the honest party. There has been a long line of research in this direction [19–25]. Previously, the trusted third party in an optimistic fair protocol which requires non-trivial computation on the inputs needs to be fully trusted and can get the output or inputs of the protocol if one party raises a dispute. This might not be desirable when the output or inputs should be strictly kept private. There are also other approaches for achieving partial fairness efficiently. But usually they work only for a specific problem. For example, the concurrent signatures protocol [26] allows two parties to produce and exchange two ambiguous signatures until an extra piece of information (called keystone) is released by one of the parties. The two parities obtain the signature from the other party concurrently when the keystone is released and therefore fairness is achieved. Kamara el al [27] proposed a new computation model in which a non-colluding server is involved. Fairness can be achieved in this model if there is a semi-trusted server, but the server has to be online during the computation. In our protocol we also require a semi-trusted server but it can be offline most of the time.

3 Building Blocks

3.1 Homomorphic Encryption

A semantically secure homomorphic public key encryption scheme is used as a building block in the protocol. There are two types of homomorphic encryption, additive and multiplicative. The additive homomorphic property can be stated as follows: (1) given two ciphertexts $E_{pk}(m_1), E_{pk}(m_2)$, $E_{pk}(m_1 + m_2) = E_{pk}(m_1) \cdot E_{pk}(m_2)$; (2) given a ciphertext $E_{pk}(m_1)$ and a constant c, $E_{pk}(c \cdot m_1) = E_{pk}(m_1)^c$. The multiplicative homomorphic property can be stated as follows: (1) given two ciphertexts $E_{pk}(m_1), E_{pk}(m_2)$, $E_{pk}(m_1 \cdot m_2) = E_{pk}(m_1) \cdot E_{pk}(m_2)$; (2) given a ciphertext $E_{pk}(m_1)$ and a constant c, $E_{pk}(m_1^c) = E_{pk}(m_1)^c$.

3.2 The Freedman-Nissim-Pinkas (FNP) Protocol

Our starting point is the PSI protocol in the semi-honest model proposed by Freedman et al. [1], which is based on oblivious polynomial evaluation. In this protocol, one party A has an input set X and another party B has an input set Y such that $|X| = |Y| = n$.[1] The two parties interact as follows

1. A chooses a key pair (pk, sk) for an additive homomorphic encryption scheme and makes the public key pk available to B.
2. A defines a polynomial $Q(y) = (y-x_1)(y-x_2)\dots(y-x_n) = \sum_{i=0}^{n} d_i y^i$, where each element $x_i \in X$ is a root of $Q(y)$. A then encrypts each coefficient d_i using the public key chosen in the last step and sends the encrypted coefficients $E_{pk}(d_i)$ to B.
3. For each element $y_j \in Y$, B evaluates $Q(y_j)$ obliviously using the homomorphic property $E_{pk}(Q(y_j)) = \prod_{i=0}^{n} E_{pk}(d_i)^{y_j^i}$. B also encrypts y_j using A's public key. B then chooses a random r_j and uses the homomorphic property again to compute $E_{pk}(r_j \cdot Q(y_j) + y_j) = E_{pk}(Q(y_j))^{r_j} \cdot E_{pk}(y_j)$. B sends each $E_{pk}(r_j \cdot Q(y_j) + y_j)$ to A. [2]
4. A decrypts each ciphertext received from B. If $y_j \in X \cap Y$, then $Q(y_j) = 0$, thus the decryption will be y_j which is also an element in X, otherwise, the decryption will be a random value. By checking whether the decryption is in X, A can output $X \cap Y$ while learns nothing about other elements in Y but not in X.

3.3 Zero Knowledge Proof

A zero knowledge proof protocol allows a prover to prove the validity of a statement without leaking any other information. The protocol presented in Section 3.2 is secure against semi-honest adversaries. However, in the presence of malicious adversaries we have to prevent the adversaries from deviating from the protocol. We enforce this by requiring each party to use zero knowledge proofs to convince the other party that it follows the protocol correctly. We will name the protocols as $PK_{(\dots)}$ and use the notation introduced in [28] to present the protocols in the rest of the paper

$$\text{ﻨ}\{\omega_i \in \mathcal{I}^*(m_{\omega_i})\}_{i=1}^{n} : \exists\{\chi_j \in \mathcal{I}^*(m_{\chi_j})\}_{j=1}^{m} : \phi(\omega_1, \dots, \omega_n, \chi_1, \dots, \chi_m)$$

In short, the prover is proving the knowledge of $\omega_1, \dots, \omega_n$ and the existence of $\chi_1, \dots, \chi_m$ such that these values satisfy certain predicate $\phi(\omega_1, \dots, \omega_n, \chi_1, \dots, \chi_m)$. Each ω_i and χ_j belongs to some integer domain $\mathcal{I}^*(m_{\omega_i})$ and $\mathcal{I}^*(m_{\chi_j})$. Each predicate is a boolean formula built from atomic predicates of discrete logarithms $y = \prod_{i=1}^{n} g_i^{F_i(\omega_1,\dots,\omega_n)}$, where F_i is an integer polynomial. All quantities except $\omega_1, \dots, \omega_n$ are assumed to be publicly known.

[1] In our protocol described in 4, we have a different requirement on the size of the input sets. This is due to the fact that the FNP protocol is a single output PSI protocol and ours is a mutual PSI protocol.

[2] For the sake of simplicity, we neglect the optimisations made in the paper to polynomial evaluation by using balanced allocation scheme and Horner's rule.

For example, the following means that given a certain group structure and a tuple (α, β, g, h), the prover can prove in zero knowledge that it knows the discrete logarithm x of α and there exists some s such that $\beta = h^x g^s$.

$$\text{ɪ}\, x \in \mathbb{Z}_q : \exists s \in \mathbb{Z}_q : \alpha = g^x \wedge \beta = h^x g^s$$

3.4 Verifiable Encryption

In a nutshell, a verifiable encryption scheme is a public key encryption scheme accompanied by an efficient zero knowledge proof of the plaintext satisfies certain properties [29]. It has numerous applications in key escrow, secret sharing and optimistic fair exchange. In optimistic fair exchange protocols, a convention is to let a party create a verifiable escrow of a data item. The escrow is essentially an encryption of the escrowed item under the offline arbiter's public key. A public data called a *label* is attached so that the arbiter can verify the decryption against the label to ensure certain properties hold. It also allows efficient zero knowledge proof of correct decryption to be constructed.

3.5 Perfectly Hiding Commitment

In our protocol, we also use a perfectly hiding commitment scheme [30] in zero knowledge proof protocols. Generally speaking, a commitment scheme is a protocol between two parties, the committer and the receiver. The committer can commit to a value v by generating a commitment $com(v)$ and sends it to the receiver. The commitment can be used as input to zero knowledge proof protocols. The commitment has two properties: *hiding* which means it is infeasible for the receiver to find v; *binding* which means it is infeasible for the committer to find another v' such that $com(v') = com(v)$. The strength of hiding and binding can be perfect or computational. In our case, we want a perfectly hiding commitment scheme which means the receiver cannot recover the value committed, even with unbounded computational power.

4 Overview of the Protocol

In this section, we give a high level view of the protocol as depicted in Fig. 1. The protocol has two sub-protocols: a PSI protocol to compute the set intersection between A and B and a dispute resolution protocol. Note in our protocol, all encryptions are in exponential form, i.e. rather than encrypting directly a message m, we encrypt g^m where g is a generator of a certain group. This modification is necessary to allow zero knowledge proof, and the modification does not affect the correctness or security of the encryption schemes. With this modification, oblivious polynomial evaluation is still possible if we use a multiplicative homomorphic encryption scheme rather than an additive one. The polynomial is moved to the exponent and the evaluation is done by operations on exponents. This is a standard technique in homomorphic encryption. For example, given $E_{pk}(g^a)$, $E_{pk}(g^b)$ and x, we can evaluate $ax + b$ obliviously and get $E_{pk}(g^{ax+b})$ by computing $(E_{pk}(g^a))^x \cdot E_{pk}(g^b)$. Having polynomial evaluation results on exponents is sufficient for our protocol, as the parties only need to test whether for certain y, $Q(y)$ is 0. This can be done effectively because $Q(y) = 0$ iff $g^{Q(y)} = 1$.

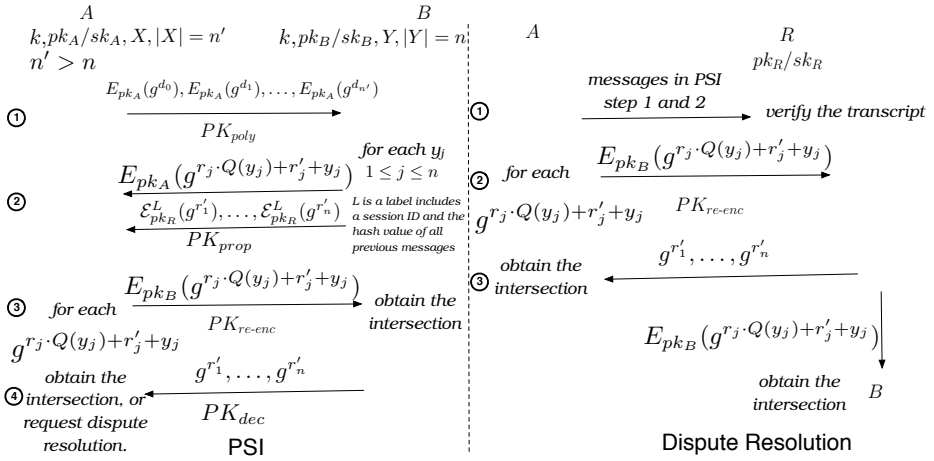


Fig. 1. Overview of the Fair PSI protocol

- **Setup:** Choose a homomorphic encryption scheme E, a verifiable encryption scheme $\mathcal{E}$, publish the public parameters. The offline arbiter R also generates a key pair for $\mathcal{E}$ and publishes the public key through a CA.
- **Private Set Intersection:** A and B are parties who engage in the computation of the set intersection, and each has a private input set X and Y respectively. In our protocol we require that A's set contains at least one random dummy element in each protocol execution. The sizes of X and Y are also required to be different. Namely, $|X| = n', |Y| = n$ such that $n' > n$. The requirements are placed to protect A's polynomial (see remark 1). A and B each also generates a random key pair for E and sends the public key to the other. They also negotiate a message authentication code (MAC) key k. This key is used by both parties to ensure the messages in the protocol execution comes from the other party. A general method to achieve this is using a MAC algorithm. To simplify presentation, we omit the MAC in the protocol description .
  1. A generates a polynomial based on A's set X as described in Section 3.2. If $d_{n'}$ is zero, regenerates the random dummy elements in X and the polynomial again until $d_{n'}$ is not zero. A encrypts all the coefficients as $E_{pk_A}(g^{d_0}), ...,$ $E_{pK_A}(g^{d_{n'}})$ and sends the ciphertexts to B. A then runs a zero knowledge proof protocol PK_{poly} to prove that the polynomial is indeed correctly constructed.
  2. For each element $y_j \in Y$, B evaluates the polynomial using the homomorphic property. Unlike in the FNP protocol that evaluates to $E_{pk_A}(r_j \cdot Q(y_j) + y_j)$, in our protocol, B also uses another random blinding factor r'_j to blind the result. So the polynomial evaluates to $E_{pk_A}(g^{r_j \cdot Q(y_j) + r'_j + y_j})$. B sends all ciphertexts to A. B then encrypts all the blinding factors r'_j using R's public key with a label L as $\mathcal{E}^L_{pk_R}(g^{r'_j})$. L includes a session ID and a hash value of all communication in the the protocol execution so far (see remark 2). B sends the encrypted

blinding factors to A, and uses PK_{prop} to prove that (1) the polynomial evaluation is properly done and (2) the encryption of blinding factors is properly done.

3. A decrypts each $E_{pk_A}(g^{r_j \cdot Q(y_j)+r'_j+y_j})$ and then encrypts each $g^{r_j \cdot Q(y_j)+r'_j+y_j}$ using B's public key. Each ciphertext $E_{pk_B}(g^{r_j \cdot Q(y_j)+r'_j+y_j})$ is sent to B and A must prove to B that the ciphertext is a correct re-encrypted ciphertext of the corresponding $E_{pk_A}(g^{r_j \cdot Q(y_j)+r'_j+y_j})$. B then decrypts each ciphertext and checks whether there is $g^{y_j+r'_j}$ that matches the decryption $g^{r_j \cdot Q(y_j)+r'_j+y_j}$, if so y_j is in $X \cap Y$.
4. B then sends $g^{r'_1}, ..., g^{r'_n}$ and proves they are correct with regard to the encryption sent in step 2. Then A will be able to test all combinations of $g^{x_i+r'_j}$ to see whether there is a match of a decryption $g^{r_j \cdot Q(y_j)+r'_j+y_j}$ it obtained in step 3, if so x_i is in $X \cap Y$. If B does not send $g^{r'_1}, ..., g^{r'_n}$ or fail to prove they are valid, A can raise a dispute with R by sending a dispute resolution request.

– **Dispute Resolution**:

1. A sends all messages sent and received in the first two setps of the PSI protocol execution to R. R verifies it by checking the consisitence between the messages and the label. If the transcript ends before the end of step 2 of the PSI protocol, R simply aborts as neither party gets any advantage.
2. A then encrypts each $g^{r_j \cdot Q(y_j)+r'_j+y_j}$ using B's public key. The ciphertext $E_{pk_B}(g^{r_j \cdot Q(y_j)+r'_j+y_j})$ is sent to R and A must prove to R that the ciphertext is a correct re-encrypted ciphertext of the corresponding $E_{pk_A}(g^{r_j \cdot Q(y_j)+r'_j+y_j})$ in the transcript.
3. R decrypts $\mathcal{E}^L_{pk_R}(g^{r'_1}), ..., \mathcal{E}^L_{pk_R}(g^{r'_n})$ and sends $g^{r'_1}, ..., g^{r'_n}$ to A, so that A can learn the intersection $X \cap Y$.
4. R also sends all $E_{pk_B}(g^{r_j \cdot Q(y_j)+r'_j+y_j})$ to B.

Remark 1: In the initialisation stage of the PSI protocol, we require A to randomise its set X by adding at least one random and secret dummy element, and make sure $|X| > |Y|$. This is to protect A's privacy. Plaintext in each $E_{pk_B}(g^{r_j \cdot Q(y_j)+r'_j+y_j})$ needs to be released to B in the PSI protocol. As r_j and r'_j are chosen by B, B might be able to recover $g^{Q(y_j)}$. B can recover A's polynomial if it can obtain at least n' $(g^{Q(y_j)}, y_j)$ pairs. In any execution of the protocol, B can recover at most n pairs. Because $n' > n$, the attack is not possible. Randomising the polynomial in each execution prevents B from pooling information gathered from different executions to recover A's polynomial.

Remark 2: We let B to encrypt blinding factors with a label L in step 2. The label L is for two purposes: (1) to ensure timeliness of dispute resolution. A session ID is attached to each protocol execution and B uses it as an input when generating the label. We assume a standard format and semantics of the session ID have been agreed by all parities beforehand, so that R can verify the identities of the two parties involved and that the protocol execution is within a certain time window. (2) To ensure the integrity of the messages in the first two steps of the protocol. As only A can raise a dispute resolution, B needs to ensure A cannot get any advantage by modifying critical messages, e.g. the encrypted coefficients and polynomial evaluation results. By using the hash of past communication as an input for the label, B can ensure that. This is because

the ciphertext with the label is encrypted under R's public key so cannot be modified without R's private key, and any modification to the messages will invalidate the label so R can detect it.

Remark 3: In our protocol B adds an additional blinding factor r'_j when evaluating A's polynomial. This is because if we follow the FNP protocol and do not add this blinding factor, then there is no good way to deal the case in which A aborts after decrypting all $E_{pk_A}(g^{r_j \cdot Q(y_j)+y_j})$. In this case to maintain fairness, B needs R to recover the set intersection. A would have to to provide a verifiable encryption of its private key sk_A in order for R to decrypt $E_{pk_A}(g^{r_j \cdot Q(y_j)+y_j})$ for B. But that will violate A's privacy because given the private key R can also recover A's polynomial coefficients from the transcript. Our design is better because now R only gets random numbers $g^{r'_1}, ..., g^{r'_n}$ which contain no information about both parties' sets.

Remark 4: In the last step of the dispute resolution protocol, R sends $E_{pk_B}(g^{r_j \cdot Q(y_j)+r'_j+y_j})$ to B. This is needed because from the transcript, R cannot tell whether A has sent them to B or not. It is possible that A unfairly aborts the protocol after finishing step 2 and then uses R to recover the result. we add this step to make sure B also receives the output in this case. And because this is the only case that A can gain advantage by unfairly aborting the protocol, we do not need a dispute resolution protocol for B.

5 A Concrete Construction

5.1 Verifiable Encryption

As a setup requirement. the arbiter R must have a key pair of a verifiable encryption scheme. In the second step of the PSI protocol, B must encrypt the blinding factors $r'_1, r'_2, ..., r'_n$ under R's public key. The encryption scheme used by R is the Cramer-Shoup encryption [31] with a small modification. The system works in this way:

- **Setup:** On input 1^k, output two prime numbers p, q such that q divides $p-1$, a cyclic group $\mathbb{G}$ with two generator g, h such that $\mathbb{G}$ is the unique order q subgroup of $\mathbb{Z}_p^*$. Choose $u_1, u_2, v_1, v_2, w \overset{R}{\leftarrow} \mathbb{Z}_q$. Compute $a = g^{u_1}h^{u_2}, b = g^{v_1}h^{v_2}, c = g^w$. Then publish (a, b, c) along with $\mathbb{G}, q, g, h$ as the public key and retain (u_1, u_2, v_1, v_2, w) as the private key.
- **Encryption:** To encrypt a message m, calculate the following:
  - $e_1 = g^z, e_2 = h^z, e_3 = c^z m$ where $z \overset{R}{\leftarrow} \mathbb{Z}_q$.
  - $\sigma = H(e_1, e_2, e_3, L)$, where H is a hash function and L is the label.
  - $e_4 = a^z b^{z\sigma}$
  - The ciphertext is (e_1, e_2, e_3, e_4).
- **Decryption:** To decrypt, compute $\sigma = H(e_1, e_2, e_3, L)$, then verify $e_1^{u_1} e_2^{u_2} (e_1^{v_1} e_2^{v_2})^{\sigma} = e_4$. If the verification succeeds, then decrypt $m = e_3/(e_1^w)$

The only modification we made to the original Cramer-Shoup encryption is that L is added as an ingredient of σ. All security properties of the Cramer-Shoup encryption are inherited.

5.2 A Homomorphic Encryption Scheme

At the core of our construction is a semantically secure homomorphic encryption scheme. Our choice is the ElGamal [32] encryption scheme. This allows us to construct efficient zero knowledge proofs needed in the protocol. To simplify design, we share certain parameters between E and $\mathcal{E}$. The scheme is described as follows:

- **Setup:** Use the same group $\mathbb{G}$ and generator g as in section 5.1. Choose $x \overset{R}{\leftarrow} \mathbb{Z}_q$ and compute g^x. The public key is $pk = (\mathbb{G}, g, g^x, q)$ and the private key is $sk = x$.
- **Encryption:** Choose $r \overset{R}{\leftarrow} \mathbb{Z}_q$ and output the ciphertext $c(m) = (g^r, m(g^x)^r)$.
- **Decryption:** The ciphertext is decrypted as $m(g^x)^r \cdot (g^r)^{-x} = mg^{rx-rx} = m$.

ElGamal is multiplicative homomorphic, so it is suitable in our protocol. As mentioned before we will convert the plaintext m to g^m before encryption, so that oblivious polynomial evaluation is possible using ElGamal.

5.3 Zero Knowledge Proof Protocols

PK_{poly}: Proof of Correct Construction of a Polynomial In step 1 of the PSI protocol, A has to prove to B that the polynomial is constructed correctly. Namely, A has to convince B that it knows the polynomial and the polynomial has no more than n' roots. For each coefficient d_i, the ciphertext is $E_{pk_A}(g^{d_i}) = (g^{t_i}, g^{d_i} g^{x_A t_i}) = (\alpha_{d_i}, \alpha'_{d_i})$, where t_i is a random number in $\mathbb{Z}_q$. To prove it knows the polynomial, A runs the following protocol:

$$\aleph\, d_i \in \mathbb{Z}_q : \exists t_i \in \mathbb{Z}_q : \alpha_{d_i} = g^{t_i} \wedge \alpha'_{d_i} = g^{d_i} (g^{x_A})^{t_i}$$

As the maximum degree of the polynomial is determined beforehand and can be verified by counting the number of encrypted coefficients received, then for a polynomial of degree n', the only case that it can have more than n' roots is when all coefficients are zero. To show the coefficients are not all zero, we require A to prove that $d_{n'}$ is not zero by running

$$\exists t_{n'}, t'_{n'} \in \mathbb{Z}_q : \alpha_{d_{n'}} = g^{t_{n'}} \wedge \alpha'_{d_{n'}} = (g^{x_A})^{t'_{n'}} \wedge t_{n'} \neq t'_{n'}$$

Intuitively, $t'_{n'} = t_{n'} + d_{n'}/x_A$ and therefore $t_{n'} = t'_{n'}$ iff $d_{n'} = 0$. So by verifying $t_{n'} \neq t'_{n'}$, B can be convinced that $d_{n'} \neq 0$. To prove the inequality of discrete logarithms, we can use the protocol proposed in [29].

PK_{prop}: Proof of Proper Polynomial Evaluation and Encryption In step 2 of the PSI protocol, B must prove that each $E_{pk_A}(g^{r_j \cdot Q(y_j) + r'_j + y_j})$ is a proper ciphertext for $g^{r_j \cdot Q(y_j) + r'_j + y_j}$, and also each $\mathcal{E}^L_{pk_R}(g^{r'_j})$ is a proper encryption under R's public key and the label L.

Recall that for an encrypted coefficient d_i, $E_{pk_A}(g^{d_i}) = (g^{r_i}, g^{d_i} g^{x_A r_i}) = (\alpha_{d_i}, \alpha'_{d_i})$. Then for each term $d_i y_j^i$ of the polynomial, the ciphertext computed using the homomorphic property from $E_{pk_A}(g^{d_i})$ is $E_{pk_A}(g^{d_i y_j^i}) = ((\alpha_{d_i})^{y_j^i}, (\alpha'_{d_i})^{y_j^i})$. Similarly, for each $r_j \cdot Q(y_j)$, the ciphertext is

$$E_{pk_A}(g^{r_j \cdot Q(y_j)}) = ((\prod_{i=0}^{n'} (\alpha_{d_i})^{r_j y_j^i}), (\prod_{i=0}^{n'} (\alpha'_{d_i})^{r_j y_j^i}))$$

B also encrypts $g^{r'_j+y_j}$ by itself, and the ciphertext $E_{pk_A}(g^{r'_j+y_j}) = (g^{\widehat{r}'_j}, g^{r'_j} g^{y_j} g^{x_A \widehat{r}'_j})$. The ciphertext of the whole can be obtained by multiplying the corresponding components of the two:

$$E_{pk_A}(g^{r_j \cdot Q(y_j)+r'_j+y_j}) = (\alpha, \beta) = ((\prod_{i=0}^{n'} (\alpha_{d_i})^{r_j y_j^i}) \cdot g^{\widehat{r}'_j}, (\prod_{i=0}^{n'} (\alpha'_{d_i})^{r_j y_j^i}) \cdot g^{r'_j} g^{y_j} g^{x_A \widehat{r}'_j})$$

For each $\mathcal{E}^L_{pk_R}(g^{r'_j})$, the ciphertext is $(e_{1j}, e_{2j}, e_{3j}, e_{4j})$, such that $e_{1j} = g^{z_j}, e_{2j} = h^{z_j}, e_{3j} = c^{z_j} g^{r'_j}, e_{4j} = a^{z_j} b^{z_j \sigma}$ where $z_j \overset{R}{\leftarrow} \mathbb{Z}_q$ and $\sigma = H(e_{1j}, e_{2j}, e_{3j}, L)$.

The proof has two steps. In the first step, B commits to y_j and $r_j y_j^i$ for each $y_j \in Y$ and $0 \leq i \leq n'$. We use the Pedersen Commitment Scheme [30] here. This commitment scheme is known to be perfectly hiding and computationally binding. It is a discrete logarithm based scheme, that enables us to re-use the parameters used for the encryption schemes. We use the same group $\mathbb{G}$, and parameters q, g, h as in section 5.1. To commit to v, choose a random s and create $com(v) = g^v h^s$. So we have $com(y_j) = g^{y_j} h^{\widetilde{s}_j}$, and $com(a_{j,i}) = g^{r_j y_j^i} h^{s_i}$ for each $a_{j,i} = r_j y_j^i$. Then starting from $i = 1$, B must prove that the value committed in $com(a_{j,i})$ is the product of the values committed in $com(a_{j,i-1})$ and $com(y_j)$. To do this, we use the protocol from [33] which proves a committed value in γ_i is the product of two other values committed in δ, γ_{i-1}:

$$\exists y_j, a_{j,i-1}, a_{j,i}, \widetilde{s}_j, s_{i-1}, s_i \in \mathbb{Z}_q : \gamma_i = g^{a_{j,i}} h^{s_i} \wedge \delta = g^{y_j} h^{\widetilde{s}_j} \wedge \gamma_{i-1} = g^{a_{j,i-1}} h^{s_{i-1}}$$

The protocol is correct because $a_{j,i} = a_{j,i-1} \cdot y_j$. Now A has a series of correct commitments of a geometric sequence $a_{j,i} = r_j y_j^i$ for $0 \leq i \leq n'$. In the second step, B runs the following protocol for each $0 \leq j \leq n$:

$$\begin{aligned} \aleph\, r'_j, y_j \in \mathbb{Z}_q : \exists a_{j,0}, ..., a_{j,n'}, \widehat{r}'_j, z_j \in \mathbb{Z}_q : \delta = g^{y_j} h^{\widetilde{s}_j} \bigwedge_{i=0}^{n'} \gamma_i = g^{a_{j,i}} h^{s_i} \\ \wedge \alpha = (\prod_{i=0}^{n'} (\alpha_{d_i})^{a_{j,i}}) \cdot g^{\widehat{r}'_j} \wedge \beta = (\prod_{i=0}^{n'} (\alpha'_{d_i})^{a_{j,i}}) \cdot g^{r'_j} g^{y_j} g^{x_A \widehat{r}'_j} \\ \wedge e_{1j} = g^{z_j} \wedge e_{2j} = h^{z_j} \wedge e_{3j} = c^{z_j} g^{r'_j} \wedge e_{4j} = a^{z_j} b^{z_j \sigma} \end{aligned}$$

B proves in the first two lines that it knows y_j, r'_j, also each exponent $a_{j,i}$ in α and β match the value committed in γ_i, y_j in β matches the value committed in δ, r'_j matches the value encrypted in e_{3j}, and (α, β) is a proper ciphertext of the polynomial evaluation result. In the last line, B proves that the verifiable encryption is correct.

***$PK_{re\text{-}enc}$*: Proof of Correct Re-encryption** In step 3 of the PSI protocol and step 2 of the dispute resolution protocol, A must prove that each value sent is the correct ciphertext $E_{pK_B}(g^{r_j \cdot Q(y_j)+r'_j+y_j})$. A generates the ciphertext by first decrypting $E_{pK_A}(g^{r_j \cdot Q(y_j)+r'_j+y_j})$, and then re-encrypting the result using B's public key. The two ciphertexts are

$$E_{pk_A}(g^{r_j \cdot Q(y_j)+r'_j+y_j}) = (g^{t_j}, g^{r_j \cdot Q(y_j)+r'_j+y_j} g^{x_A t_j}) = (g^{t_j}, m_j g^{x_A t_j})$$

$$E_{pk_B}(g^{r_j \cdot Q(y_j)+r'_j+y_j}) = (g^{t'_j}, g^{r_j \cdot Q(y_j)+r'_j+y_j} g^{x_B t'_j} = (g^{t'_j}, m_j g^{x_B t'_j}))$$

where t_j, t'_j are random numbers. The protocol is then:

$$\exists x_A, t'_j \in \mathbb{Z}_q : pk_A = g^{x_A} \wedge \alpha = m_j (g^{t_j})^{x_A} \wedge \beta = g^{t'_j} \wedge \gamma = m_j (g^{x_B})^{t'_j}$$

The proof shows that the two ciphertexts are correct and encrypt the same plaintext.

PK_{dec}: Proof of Correct Decryption In step 4 of the PSI protocol, B needs to prove that each $g^{r_j'}$ is a correct decryption of $\mathcal{E}^L_{pk_R}(g^{r_j'})$. For each $\mathcal{E}^L_{pk_R}(g^{r_j'})$, the ciphertext is $(e_{1j}, e_{2j}, e_{3j}, e_{4j})$, such that $e_{1j} = g^{z_j}, e_{2j} = h^{z_j}, e_{3j} = c^{z_j} g^{r_j'}, e_{4j} = a^{z_j} b^{z_j \sigma}$ where $z_j \overset{R}{\leftarrow} \mathbb{Z}_q$ and $\sigma = H(e_{1j}, e_{2j}, e_{3j}, L)$. What B needs to show is that it knows z_j and z_j is used consistently in all ciphertext compoents.

$$\exists z_j \in \mathbb{Z}_q : e_{1j} = g^{z_j} \wedge e_{2j} = h^{z_j} \wedge e_{3j} = c^{z_j} g^{r_j'} \wedge e_{4j} = a^{z_j} (b^{\sigma})^{z_j}$$

If $g^{r_j'}$ is not the correct decryption, then B cannot find a z_j that satisfies the relation.

5.4 Complexity Analysis

Now we give an account of the complexity of the protocol. The computational and communication complexity of the zero knowledge proof protocol is linear in the number of statements to be proved, so we separate it from the main protocol. In the PSI protocol, A needs to perform $3n'$ exponentiations to encrypt the coefficients in step 1, and $3n$ exponentiations to decrypt and re-encrypt the polynomial evaluation results in step 3, B needs $2(n'n + 2n)$ exponentiations to evaluate the polynomial obliviously and $3n$ exponentiations for the verifiable encryption in step 2. The messages sent in the protocol consist of $2n' + 9n$ group elements. In the dispute resolution protocol, R needs $6n$ exponentiations to verify and decrypt the ciphertexts of the verifiable encryption. The total traffic generated includes $5n$ group elements, plus the transcript sent in step 1. In total, the computational complexity is $O(nn')$ and the communication complexity is $O(n + n')$. The complexity of the zero-knowledge proof protocols: PK_{poly} is $O(n')$, PK_{prop} is $O(nn')$, $PK_{re\text{-}enc}$ is $O(n)$, and PK_{dec} is $O(n)$. The complexity of our protocol is similar to other PSI protocols in the malicious model [2, 3].

6 Security Analysis

6.1 Security Model

The basic security requirements of our protocol are correctness, privacy and fairness. Informally, correctness means an honest party is guaranteed that the output it receives is correct with regard to the actual input and the functionality realised by the protocol; privacy means no party should learn more than its prescribed output from the execution of the protocol; fairness means a dishonest party should receive its output if and only if the honest party also receives its output.

We define a security model to capture the above security requirements in terms of the simulation paradigm [14]. We model the parties A, B and R as probabilistic interactive Turing machines. A *functionality* is denoted as $f : \mathcal{X}_A \times \mathcal{X}_B \rightarrow \mathcal{Y}_A \times \mathcal{Y}_B$, In our protocol, the functionality to be computed by A and B is the set intersection. The model is similar to the one used in the optimistic fair secure computation protocol [23]. Generally speaking the protocol is executed in a real world model where the participants may be corrupted and controlled by an adversary. To show the protocol is secure, we define an ideal process which satisfies all the security requirements. In the ideal process, there is an incorruptible trusted party which helps in the computation of the functionality, e.g. in our case the set intersection. The protocol is said to be secure if for every adversary

in the real world model there is also an adversary in the ideal world model who can simulate the real world adversary.

The Real World. The protocol has three participants A, B and R. All participants have the public parameters of the protocol including the function $f_\cap$, the security parameter κ, R's public key and other cryptographic parameters to be used. A has a private input X, B has a private input Y and R has an input $\in \{\diamond, \perp\}$. The participants of the protocol can be corrupted by an adversary. The adversary can corrupt up to two parties in the protocol. We use C to denote the adversary. The adversary can behave arbitrarily, e.g. substitute local input, abort the protocol prematurely, and deviate from the protocol specification. At the end of the execution, an honest party outputs whatever prescribed in the protocol, a corrupted party has no output, and an adversary outputs its view. For a fixed adversary C, and input X, Y, the joint output of A, B, R, C is denoted by $O_{ABRC}(X, Y)$ which is the random variable consisted of all the outputs as stated.

The Ideal Process. In the ideal process, there is an incorruptible trust party T, and parties $\bar{A}, \bar{B}, \bar{R}$. $\bar{A}$ has input X, $\bar{B}$ has input Y and $\bar{R}$ has an input $\in \{\diamond, \perp\}$. The operation is as follows:

- $\bar{A}$ sends X' or $\perp$ to T, then $\bar{B}$ sends Y' or $\perp$ to T, then $\bar{R}$ sends two messages $b_A \in \mathcal{Y}_A \cup \{\diamond, \perp\}$ and $b_B \in \mathcal{Y}_B \cup \{\diamond, \perp\}$ to T. The actual input X' and Y' may be different from X and Y if the party is malicious.
- T sends private delayed output to $\bar{A}$ and $\bar{B}$. T's reply to $\bar{A}$ depends on $\bar{A}$ and $\bar{B}$'s messages and b_A. T's reply to $\bar{B}$ depends on $\bar{A}$ and $\bar{B}$'s messages and b_B.
  - T to $\bar{A}$: (1) If $b_A = \diamond$, $\bar{A}$ sends X' and $\bar{B}$ sends Y', T sends $X' \cap Y'$ to $\bar{A}$.
    (2) Else if $b_A = \diamond$, but $\bar{A}$ or $\bar{B}$ sends $\perp$, T sends $\perp$ to $\bar{A}$
    (3) Else if $b_A \neq \diamond$, T sends b_A to $\bar{A}$.
  - T to $\bar{B}$: (1) If $b_B = \diamond$, $\bar{A}$ sends X' and $\bar{B}$ sends Y', T sends $X' \cap Y'$ to $\bar{B}$.
    (2) Else if $b_B = \diamond$, but $\bar{A}$ or $\bar{B}$ sends $\perp$, T sends $\perp$ to $\bar{B}$.
    (3) Else if $b_A \neq \diamond$, T sends b_B to $\bar{B}$.

Honest parties in the ideal process behave as follows: $\bar{A}$ and $\bar{B}$ send their input to T and $\bar{R}$ sends $b_a = \diamond$ and $b_B = \diamond$. The ideal process adversary $\bar{C}$ controls the behaviours of corrupted parties. It gets the input of a corrupted party and may substitute them. It also gets T's answer to corrupted parties. For a fixed adversary $\bar{C}$, and input X, Y, the joint output of $\bar{A}, \bar{B}, \bar{R}, \bar{C}$ in the ideal process is denoted by $O_{\bar{A}\bar{B}\bar{R}\bar{C}}(X, Y)$.

Simulatability. The security definition is in terms of simulatability:

Definition 1. *Let $f_\cap$ be the set intersection functionality. We say a protocol Π securely computes $f_\cap$ if for every real-world adversary C, there exists an adversary $\bar{C}$ in the ideal process such that for all $X \in \mathcal{X}_A$, for all $Y \in \mathcal{X}_B$, the joint distribution of all outputs of the ideal process is computationally indistinguishable from the outputs in the real world, i.e.,*

$$O_{\bar{A},\bar{B},\bar{R},\bar{C}}(X, Y) \stackrel{c}{\approx} O_{ABRC}(X, Y)$$

The design of the ideal process captures the security we want to achieve from the real protocol. Our assumption is that in real world, we can find a semi-trusted arbiter that can be trusted for fairly resolving disputes, but not for privacy. Then by incorporating

such an arbiter in a two-party private set intersection protocol, we can achieve fairness, correctness and privacy. In the ideal process, if $\bar{R}$ follows the protocol and does not collude with $\bar{A}$ or $\bar{B}$ then all security properties are guaranteed. In this case, $\bar{A}$ and $\bar{B}$ will always get the correct intersection with regard to the actual input to the protocol, and know nothing more than that. On the other hand, if $\bar{R}$ is corrupted and colludes with $\bar{A}$ or $\bar{B}$, then fairness is not guaranteed. However, even in this case privacy is guaranteed. That is, the corrupted parties will not get more information about the honest party's set other than the intersection.

6.2 Security Proof

We are now ready to state and prove the security of our protocol. The protocol uses zero knowledge proof protocols as subprotocols. As they are obtained by using existing secure protocols and standard composition techniques, they are consequently secure and we omit the security proofs of them. To prove the main theorem below, we work in a *hybrid model* in which the real protocol is replaced with a hybrid protocol such that every invocation of the subprotocols is replaced by a call to an ideal functionality computed by a trusted party. In our case we need ideal functionalities for zero knowledge proofs and certification authority. If the subprotocols are secure, then by the composition theorem [34] the output distribution of the hybrid execution is computationally indistinguishable from the output distribution of the real execution. Thus, it suffices to show that the ideal execution is indistinguishable from the hybrid execution.

Theorem 1. *If the encryption E and $\mathcal{E}$ are semantically secure, and the associated proof protocols are zero knowledge proof, the optimistic fair mutual private set intersection protocol securely computes $f_{\cap}$.*

Because of limited space, below we only sketch the proof. The detailed proof will appear in the full version.

Proof. Let's first consider the cases that the adversary C corrupts two parties.

Case 1: C corrupts and controls A and B. This is a trivial case because C has full knowledge on X, Y and if the encryption scheme used by R is semantically secure, a simulator can always be constructed.

Case 2: C corrupts and controls A and R. We construct a simulator S in the ideal process that corrupts and controls $\bar{A}$ and $\bar{R}$. It uses the adversary C as a subroutine and we will show the simulatability holds in this case.

1. S is given A and R's inputs, S invokes an ideal functionality CA to obtain R's key pair, then invokes C and plays the role of B.
2. S generates a public/private key pair pk_B/sk_B and gives the public key to C.
3. S receives the encrypted coefficients $E_{pk_A}(d_i)$ from C. S also receives $d_i, 0 \leq i \leq n'$ for the ideal computation of PK_{poly}, where d_i is a coefficient of the polynomial. If the polynomial is not correctly constructed, then S instructs $\bar{A}$ to send $\perp$ to T and terminates the execution. If the polynomial is correct, S extracts input X' from the coefficients, instructs $\bar{A}$ to send X' to T and instructs $\bar{R}$ to send $b_A = \diamond$ to T. S then receives the intersection $X' \cap Y$ from T.

4. S then constructs Y' from the intersection received in last step by adding random dummy elements until $|Y'| = n$. Then It generates a set of random blinding factors $r'_1, r'_2, ..., r'_n$, computes $E_{PK_A}(g^{r_j \cdot Q(y'_j)+r'_j+y'_j})$ and encrypts all blinding factors using R's public key. It also generates commitments for each y'_j and $r_j y'^i_j$. S sends all commitments and ciphertexts to C and also emulates the ideal computation of PK_{prop} by sending "accept" to C. Depends on C's reply, executes step 5, 6 or 7. In the next three steps, S will send an instruction to T when it is ready to output, then T sends the delayed output to $\bar{B}$
5. If C instructs both A and R to abort, then S instructs $\bar{R}$ to send $b_B = \perp$ to T, then outputs whatever C outputs and terminates.
6. If C instructs A to abort and instructs R to send n ciphertexts, S decrypts them using B's private key, constructs a set by testing whether any elements in Y' match the decryption results. Then S collects all matching elements, put them in a set and instructs $\bar{R}$ to send the set as b_B. Then S outputs whatever C outputs and terminates.
7. If C instructs A to send n ciphertexts, then S extracts a set of elements from the reply and engages in the ideal computation of PK_{re-enc}. If the reply is correct, S instructs $\bar{R}$ to send $b_B = \diamond$ to T and sends $g^{r'_1}, ..., g^{r'_n}$ to C. If the reply is not correct and C instructs R to abort, S instructs $\bar{R}$ to send $b_B = \perp$ to T. If the reply is not correct and C instructs R to send n ciphertexts, S extracts a set of elements from the cipehrtexts and instructs $\bar{R}$ to send the set as b_B to T. Then it outputs whatever C outputs and terminates.

In the joint output, the honest parties' outputs are always the same. All we need to check is whether the view of the simulator is indistinguishable from the view of an adversary in the hybrid execution. The difference between a simulation and a hybrid execution is that in the simulation S uses Y' which is not the same as Y. However, this does not affect the distribution of the views. From how Y' is constructed we can see that Y' contains the correct intersection ($Y \cap X' \subseteq Y'$). For those elements in the intersection, they produce the same distributions in the simulation (using Y') and the hybrid execution (using Y). For any elements $y'_j \in Y'$ and $y_j \in Y$ not in the intersection, the commitments produced should be indistinguishable because the commitment scheme is perfectly hiding. Also $g^{r_j \cdot Q(y_j)+r'_i+y'_j}$ and $g^{r_j \cdot Q(y'_j)+r'_i+y_j}$ are uniformly random because $Q(y_j)$ and $Q(y'_j)$ are both non-zero, and so are the ciphertexts of them. The blinding factors and their ciphertexts are uniformly random in both the simulation and the hybrid execution. Therefore the two views are indistinguishable.

Case 3: C corrupts and controls B and R. We construct a simulator S in the ideal process that corrupts and controls $\bar{B}$ and $\bar{R}$. It uses the adversary C as a subroutine.

1. S is given B and R's inputs, S invokes an ideal functionality CA to obtain R's key pair, then invokes C and plays the role of A.
2. S generates a key pair pk_A/sk_A and gives the public key to C.
3. S generates a random set X' such that $|X'| = n'$, then constructs a polynomial using elements in X'. S encrypts the coefficients, sends them to C and simulates the ideal computation of PK_{poly} by sending "accept" to C.

4. S receives the commitments and ciphertexts from C, then receives inputs to the ideal computation of PK_{prop}, including $(y_j, r'_j), 0 \leq j \leq n$. If the ciphertexts are not properly produced, S instructs $\bar{B}$ to send $\perp$ to T, otherwise S extract Y' and instructs $\bar{B}$ to send Y' to T and instructs $\bar{R}$ to send $b_B = \diamond$ to T, and receives $X \cap Y'$ from T.
5. S constructs another set X'' such that $X \cap Y' \subseteq X''$ and $|X''| = n'$. S then constructs another polynomial Q'', and evaluates the polynomial using $(y_j, g^{r'_j})$ to construct $E_{pk_B}(g^{r_j \cdot Q''(y_j) + r'_j + y_j})$. The ciphertexts are sent to C, S also simulates the ideal computation of PK_{re-enc} by sending "accept" to C. Depends on C's reply, executes step 6,7 or 8. In the next three steps, S will send an instruction to T when it is ready to output, then T sends the delayed output to $\bar{A}$
6. If C instructs B to send the blinding factors, then S instructs $\bar{R}$ to send $b_A = \diamond$, outputs whatever C outputs and terminates.
7. If C instructs both B and R to abort, then S instructs $\bar{R}$ to send $b_A = \perp$, outputs whatever C outputs and terminates.
8. If C instructs B to abort and R to send n blinding factors, use the blinding factors to extract a set, and then instructs $\bar{R}$ to send the extracted set as b_A to T. S then outputs whatever C outputs and terminates.

The difference between a simulation and a hybrid execution is that the simulator uses X' and X'' rather than the honest party's input X. Using X' does not affect the distribution of the view if E is semantically secure, because the ciphertexts generated using A's public key are indistinguishable. Using X'' also does not affect the distribution of the view. For the two sets X'' and X, two polynomials are constructed from them Q'' and Q. We also know $X'' \cap Y' = X \cap Y'$, so $Q''(y_j) = 0$ iff $Q(y_j) = 0$ for any $y_j \in Y'$. For each $g^{r_j \cdot Q''(y_j) + r'_j + y_j}$ and $g^{r_j \cdot Q(y_j) + r'_j + y_j}$, if $Q''(y_j) = 0$ then $Q(y_j) = 0$ so the distribution of the two depends only on y_j and r'_j, if $Q''(y_j) \neq 0$ then $Q(y_j) \neq 0$ and both $Q''(y_j)$ and $Q''(y_j)$ are uniformly random, so $g^{r_j \cdot Q''(y_j) + r'_j + y_j}$ and $g^{r_j \cdot Q(y_j) + r'_j + y_j}$ are also uniformly random. Therefore the distributions of the views are indistinguishable.

For cases that C corrupts only one party, proofs can be constructed similarly. In the case that R is corrupted, R is not involved in the protocol because A and B are honest, so it is trivial to construct a simulator. In the case that A or B is corrupted, the simulator can be constructed as in case 2 step 1 – 4 or case 3 step 1 – 5, except now $\bar{R}$ is honest and always sends $\diamond$ to T. The view from the simulation is still indistinguishable.

7 Conclusion and Future Work

In this paper, we have presented a fair mutual PSI protocol which allows both parties to obtain the output. The protocol is optimistic which means fairness is obtained by using an offline third party arbiter. To address the possible privacy concerns raised by introducing a third party, the protocol is designed to enable the arbiter to resolve dispute blindly without knowing any private information from the two parties. We have analysed and shown that the protocol is secure.

The communication and computation complexity of our protocol are both $O(nn')$. The main overhead comes from the oblivious polynomial evaluation and the large accompanying zero knowledge proof. We would like to investigate PSI protocols based on

other primitives, e.g. [6, 7], to see whether efficiency can be improved. Another area we would like to investigate is whether the protocol structure that we use to obtain fairness can be made general so that it can be applied to other secure computation protocols.

Acknowledgements. We would like to thank the anonymous reviewers. Changyu Dong is supported by a Science Faculty Starter Grant from the University of Strathclyde.

References

1. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
2. Dachman-Soled, D., Malkin, T., Raykova, M., Yung, M.: Efficient robust private set intersection. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 125–142. Springer, Heidelberg (2009)
3. Hazay, C., Nissim, K.: Efficient set operations in the presence of malicious adversaries. In: Public Key Cryptography, pp. 312–331 (2010)
4. Hazay, C., Lindell, Y.: Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 155–175. Springer, Heidelberg (2008)
5. Jarecki, S., Liu, X.: Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 577–594. Springer, Heidelberg (2009)
6. De Cristofaro, E., Tsudik, G.: Practical private set intersection protocols with linear complexity. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 143–159. Springer, Heidelberg (2010)
7. De Cristofaro, E., Kim, J., Tsudik, G.: Linear-complexity private set intersection protocols secure in malicious model. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 213–231. Springer, Heidelberg (2010)
8. Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
9. Camenisch, J., Zaverucha, G.M.: Private intersection of certified sets. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 108–127. Springer, Heidelberg (2009)
10. Kim, M., Lee, H.T., Cheon, J.H.: Mutual private set intersection with linear complexity. IACR Cryptology ePrint Archive 2011, 267 (2011)
11. Huang, Y., Evans, D., Katz, J.: Private set intersection: Are garbled circuits better than custom protocols? In: 19th Network and Distributed Security Symposium (2012)
12. Agrawal, R., Evfimievski, A.V., Srikant, R.: Information sharing across private databases. In: SIGMOD Conference, pp. 86–97 (2003)
13. Vaidya, J., Clifton, C.: Secure set intersection cardinality with application to association rule mining. Journal of Computer Security 13(4), 593–622 (2005)
14. Goldreich, O.: Foundations of Cryptography: Volume II Basic Applications. Cambridge University Press (2004)
15. Cleve, R.: Limits on the security of coin flips when half the processors are faulty (extended abstract). In: STOC, pp. 364–369 (1986)
16. Blum, M.: How to exchange (secret) keys. ACM Trans. Comput. Syst. 1(2), 175–193 (1983)
17. Ben-Or, M., Goldreich, O., Micali, S., Rivest, R.L.: A fair protocol for signing contracts. IEEE Transactions on Information Theory 36(1), 40–46 (1990)

18. Pinkas, B.: Fair secure two-party computation. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 87–105. Springer, Heidelberg (2003)
19. Asokan, N., Schunter, M., Waidner, M.: Optimistic protocols for fair exchange. In: ACM Conference on Computer and Communications Security, pp. 7–17 (1997)
20. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures (extended abstract). In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 591–606. Springer, Heidelberg (1998)
21. Bao, F., Deng, R.H., Mao, W.: Efficient and practical fair exchange protocols with off-line ttp. In: IEEE Symposium on Security and Privacy, pp. 77–85 (1998)
22. Ateniese, G.: Efficient verifiable encryption (and fair exchange) of digital signatures. In: ACM Conference on Computer and Communications Security, pp. 138–146 (1999)
23. Cachin, C., Camenisch, J.: Optimistic fair secure computation. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 93–111. Springer, Heidelberg (2000)
24. Micali, S.: Simple and fast optimistic protocols for fair electronic exchange. In: PODC, pp. 12–19 (2003)
25. Dodis, Y., Lee, P.J., Yum, D.H.: Optimistic fair exchange in a multi-user setting. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 118–133. Springer, Heidelberg (2007)
26. Chen, L., Kudla, C., Paterson, K.G.: Concurrent signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 287–305. Springer, Heidelberg (2004)
27. Kamara, S., Mohassel, P., Riva, B.: Salus: A system for efficient server-aided multi-party computation. In: ACM Conference on Computer and Communications Security (CCS 2012) (2012)
28. Camenisch, J., Krenn, S., Shoup, V.: A framework for practical universally composable zero-knowledge protocols. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 449–467. Springer, Heidelberg (2011)
29. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
30. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
31. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
32. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
33. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified vss and fact-track multiparty computations with applications to threshold cryptography. In: PODC, pp. 101–111 (1998)
34. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptology 13(1), 143–202 (2000)

Bloom Filter Bootstrap: Privacy-Preserving Estimation of the Size of an Intersection

Hiroaki Kikuchi[1,⋆] and Jun Sakuma[2]

[1] Department of Frontier Media Science,
School of Interdisciplinary Mathematical Sciences,
Meiji University 4-21-1 Nakano, Nakano Ku, Tokyo, 164-8525 Japan
kikn@meiji.ac.jp
[2] Graduate School of SIE, Computer Science Department,
University of Tsukuba 1-1-1 Tennodai, Tsukuba, 305-8573 Japan
jun@cs.tsukuba.ac.jp

Abstract. This paper proposes a new privacy-preserving scheme for estimating the size of the intersection of two given secret subsets. Given the inner product of two Bloom filters (BFs) of the given sets, the proposed scheme applies Bayesian estimation under assumption of beta distribution for an a priori probability of the size to be estimated. The BF retains the communication complexity and the Bayesian estimation improves the estimation accuracy.

An possible application of the proposed protocol is an epidemiological datasets regarding two attributes, *Helicobactor pylori* infection and stomach cancer. Assuming information related to Helicobactor Pylori infection and stomach cancer are separately collected, the protocol demonstrates that a χ^2-test can be performed without disclosing the contents of the two confidential databases.

1 Introduction

With the rapid development of database systems and online services, large amounts of information are being collected and accumulated from various data sources independently and simultaneously. Privacy- preserving data mining (PPDM) has been attracting significant attention as a technology that could enable us to perform data analysis over multiple databases containing sensitive information without violating subjects' privacy.

In this paper, we investigate the problem of set intersection cardinality. Given two private sets, the goal of this problem is to evaluate the cardinality of the intersection without disclosing the sets mutually. Set intersection cardinality has been extensively studied as a building block of PPDM, including association rule mining [17], model and attribute selection[16], and other aspects [4]. Our major application of this problem is epidemiological analysis, including

⋆ This work was done when author was in Tokai University.

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 145–163, 2013.

privacy-preserving cohort studies. We wish to perform cohort studies over multiple independently collected medical databases, which are not allowed to disclose identifying information about patients.

Consider two databases developed independently by two organizations. One organization collects individual medical information, including patient ID, patient name, patient address, presence or absence of disease 1, disease 2, and so on. The other organization collects individual genome information from research participants; including participant ID, participant name, participant address, presence or absence of genome type 1, genome type 2, and so on. The objective of a cohort study may be to investigate the association between the outbreak of a specific disease and genomes. For this analysis, the analyst makes use of four-cell contingency tables; each cell counts the number of patients who have (do not have) a specific disease and have (do not have) a specific genome type. If both tables are private, the set intersection cardinality may be used for evaluating of the count of each cell without sharing database content. In this study, we consider the following four requirements for practical situations.

Requirement 1. The time and communication complexity should be linear with respect to the number of records n. This is because statistical analysis, including cohort studies, usually treats databases with a large number of records.

Requirement 2. The time and communication complexity should be independent of the size of the ID space. In the use case described above, both organizations independently collect information from individuals. Thus, unique IDs are not given to records. Instead, the protocol must generate a unique ID for each record with the combination of individual attributes, such as the name and address. Because the space required for the combination of such user attributes is often much larger than the number of individuals, this requirement is important.

Requirement 3. The protocol should be designed considering the asymmetry of computational capabilities of organizations. Assume that a research institute that holds genome information provides epidemiological analysis services upon request to hospitals that hold medical information. In such a case, it is expected that the computational capabilities of the hospitals are poor. Therefore, a reasonable solution can be the outsourcing of computation; the research institute offers servers with high computational power and the hospital outsources most of the computation required for the analysis to the research institute. This example indicates that the protocol of set intersection cardinality should be designed considering the asymmetry of computational capabilities.

Requirement 4. The outputs of the protocol may be random shares. This requirement implicitly suggests that the set intersection cardinality may be used as a part of a larger-scale protocol. If the outputs of the protocol are random shares, these can be seamlessly used for inputs to other privacy-preserving protocols.

In this paper, we propose a set intersection cardinality protocol that satisfies these requirements.

Related Work

Let S_A and S_B be private inputs of the set intersection cardinality. Let n_A and n_B be the cardinalities of S_A and S_B, respectively.

Agrawal et al. [1] presented a set intersection cardinality protocol using commutative encryption under DDH (Decisional Diffie-Hellman) assumption. The time complexity of this protocol is $O(n_A + n_B)$; this is linear in the size of the databases and is independent of the size of the ID space. However, this protocol assumes that the two parties have nearly the same computation power. Furthermore, the protocol cannot output random shares. De Cristofaro and Tsudik [5] introduced an extension of [1]. It also requires $O(n)$ computation by both parties.

Freedman et al. [7] proposed a set intersection protocol using oblivious polynomial evaluation. This protocol can be converted to the set intersection cardinality with a slight modification, and achieves $O(n_B + \log\log n_A)$ time/communication complexity. Furthermore, the time complexity is independent of the ID space size and random shares can be output. This protocol also assumes that both parties have equal computational power.

All the above protocols guarantee exact outputs. Kantarcioglu et al. [11] approach the set intersection cardinality differently. Their protocol maps the input set onto a binary vector using a Bloom filter (BF)[2], and the set intersection cardinality is statistically estimated from the scalar product of the two binary vectors. With this approach, the results become approximations, although the computation cost is expected to be greatly reduced. The dimensionality of the vector used in this protocol is equal to the ID space size; this does not meet Requirement 2. In [11], a technique to shorten large IDs using hash functions was used with their protocol. As shown later by our theoretical analysis, given an error rate ϵ, the optimal range of hash functions for n elements is $O(n^2)$. This indicates that such Naive ID generation can be too inefficient for practical use.

Camenisch and Zaverucha [3] has introduced the certified set intersection cardinality problem. This protocol considers asymmetry in the security assumptions of the parties, but does not consider asymmetry in their computational capability.

Ravikumar et al. used the TF-IDF measures to estimate the scalar product in [15]. As for epidemiological study, Lu et al. studied the contingency tables in [13].

Thus, to our knowledge, no set intersection cardinality protocol satisfies the four requirements above, which should be met for practical privacy-preserving data analysis, especially for the outsourcing models.

Our Contribution

In this manuscript, we present a protocol that satisfies the four requirements. Considering the first and second requirement, the sets are independently mapped onto BFs, and then the set intersection cardinality is statistically estimated from the scalar product of the two binary vectors representing the BFs.

As discussed later, the size of the BF must be $O(n^2)$ to control the false positive rate in [11]; this does not meet Requirement 2. Our protocol therefore uses a number of BFs of size $O(n)$. The set intersection cardinality is obtained by iteratively applying Bayesian estimation to the scalar products of the BFs.

In the proposed protocol, the scalar product protocol is used as a building block. Modulo exponentiation is performed only by one party and this fits well with the outsourcing model (Requirement 3). In addition, the outputs can naturally be made random shares (Requirement 4).

We demonstrate our protocol with an epidemiological datasets regarding two attributes, *Helicobactor pylori* infection and stomach cancer. Assuming information related to Helicobactor Pylori infection and stomach cancer are separately collected, we demonstrate that a χ^2-test can be performed without disclosing the contents of the two databases.

2 Preliminary

2.1 Bloom Filter

A BF is a simple space-efficient data structure for representing a set to support membership queries[2]. Recently, BFs have been used not only for database applications but also for network problems including detecting malicious addresses, packet routing, and the measurement of traffic statistics.

A BF for representing a set $S = \{a_1, \ldots, a_n\}$ of n elements is an array of m bits, initially all set to 0. The BF uses k independent hash functions $H_1, \ldots, H_k$ such that $H_i : \{0,1\}^* \to \{1, \ldots, m\}$. The hash functions map each element in the map to a random number uniformly chosen from $\{1, \ldots, m\}$. Let $B(S)$ be a set representing a BF defined by $B(S) = \bigcup_{a \in S} B(a)$ such that $B(a) = \{H_1(a), \ldots, H_k(a)\}$. Now let $\boldsymbol{b}$ be an m-dimensional vector, $(b_1, \ldots, b_m)$, which is an alternative representation of the BF, defined by $b_i = \begin{cases} 1 \text{ if } i \in B(S), \\ 0 \text{ if } i \notin B(S), \end{cases}$ for $i = 1, \ldots, m$. For example, the hash functions that map an element a as $H_1(a) = 2$, $H_2(a) = 7$ characterize a BF with $m = 8$, $B(a) = \{2, 7\}$. Alternatively, $\boldsymbol{b}(a) = (0, 1, 0, 0, 0, 0, 1, 0)$. We can use either the set or vector representation of BF, depending on the cryptographic building blocks used. Note the following relationship between the set and vector representations, $\boldsymbol{b}(S_1) \cdot \boldsymbol{b}(S_2) = |B(S_1) \cap B(S_2)|$.

To test if a is an element of set S, we can verify that

$$\forall i = 1, \ldots, k \, H_i(a) \in B(S), \tag{1}$$

which holds if $a \in S$. However, it also holds, with a small probability, even if $a \notin S$. That is, BFs suffer from false positives. According to [2], after all the elements of S are hashed into the BF, the probability that element i does not belong to $B(S)$, i.e., that the i-th bit of $\boldsymbol{b}(S)$ is 0, is $p = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-kn/m}$. We therefore have a probability of false positives given by $p' = \left(1 - (1 - \frac{1}{m})^{kn}\right)^k \approx \left(1 - e^{-kn/m}\right)^k$. If k is sufficiently small for given m and n, Equation (1) is likely to hold only for the element of S. Conversely, with too large a value for k, the BF is mostly occupied by 1 values. In [2,6], the optimal BF was found for $k^* = \ln 2 \cdot (m/n)$, which minimized the false-positive probability.

2.2 Cryptographic Primitives

Secure Scalar Product. The scalar product of two vectors is performed securely by using a public-key encryption algorithm in Algorithm 1.

Algorithm 1. Secure Scalar Product

Input: Alice has an n-dimensional vector $\boldsymbol{x} = (x_1, \ldots, x_n)$. Bob has an n-dimensional vector $\boldsymbol{y} = (y_1, \ldots, y_n)$.
Output: Alice has s_A and Bob has s_B such that $s_A + s_B = \boldsymbol{x} \cdot \boldsymbol{y}$.

1. Alice generates a homomorphic public-key pair and sends the public key to Bob.
2. Alice sends to Bob n ciphertexts $E(x_1), \ldots, E(x_n)$, encrypted with her public key.
3. Bob chooses s_B at random, computes $c = E(x_1)^{y_1} \cdots E(x_n)^{y_n} / E(s_B)$ and sends c to Alice.
4. Alice uses her secret key to decrypt c to obtain $s_A = D(c) = x_1 y_1 + \cdots + x_n y_n - s_B$

Security Model. We assume that the parties are *honest-but-curious*, which is known as *semi-honest* model, with parties that own private datasets following protocols properly but trying to learn additional information about the datasets from received messages.

We also assume the Decisional Diffie-Hellman hypothesis (DDH), that is, a distribution of (g^a, g^b, g^{ab}) is indistinguishable from a distribution of (g^a, g^b, g^c), where a, b, c are uniformly chosen from Z_q.

3 Difficulties in ID-less Datasets

3.1 Problem Definition

We are considering the problem of a two-party protocol that can evaluate the size of the intersection of two sets without revealing the sets themselves.

Let A and B be parties owing subsets S_A and S_B, respectively. For an agreed threshold t, they each wish to know if

$$X = |S_{A \cap B}| = |S_A \cap S_B| \geq t \tag{2}$$

is true, without revealing S_A or S_B to the other party. Here, X is a random variable describing the size of the intersection $S_{A\cap B}$.

Note that we are not interested in learning about the intersection, itself but are only interested in evaluating its size because the size is often useful in many privacy-preserving applications. For example, an epidemic study might test if the difference between two subsets is statistically significant. The difference of $|X_{A\cap B}|$ and t may even be confidential in some applications.

3.2 Naïve ID Generation

Consider a dataset of n elements with multiple attributes, such as name, sex, age and address, but with no unique identity being assigned. Instead, the elements are uniquely specified by attributes, e.g., name and birthday. Let A be a set of attrbutes $A = \{a_1, \ldots, a_n\}$.

The simplest way to generate a pseudo identity is to use a hash function $h : \{0,1\}^* \to \{1, \ldots, \ell\}$. Using this hash function, we assign $h(a_i)$ to the i-th element. For efficiency reasons, we assume the range is sufficiently large that we can neglect the occurrence of a collision such that $h(a_i) = h(a_j)$ for some $i \neq j$. Letting h_A be the set of all pseudo identities, defined as $h_A = \{h(a_i) \mid a_i \in A\}$, we can see observe any collision of identities by testing whether $|h_A| = n$.

If the size ℓ of the ID set increases, collisions can be avoided, but the computational cost will accordingly increase with ℓ. Clearly, $\ell \geq n$, but finding the optimal size is not trivial. To solve the tradeoff between accuracy and performance reduction, let us assume we have an optimal ℓ that is sufficiently large to uniquely determine the given set of n elements.

This problem is equivalence to the problem known as "birthday paradox", whereby, among a set of n randomly chosen people, there is a probability that some pair of them have the same birthday. When identities (birthdays) are chosen with a uniform probability of $1/\ell$, the probability that all n identities are unique is given by

$$\prod_{j=1}^{n-1}\left(1-\frac{j}{\ell}\right) \approx \prod_{j=1}^{n-1} e^{-j/\ell} = e^{-n(n-1)/2\ell} \approx e^{-n^2/2\ell}.$$

Therefore, given the probability ϵ with which n hash values are unique, we have $\frac{n^2}{2\ell} = \ln \epsilon^{-1}$, from which follows the solution of our problem. The optimal range of hash functions for n elements is given as $\ell = n^2/2 \ln \epsilon^{-1}$, for which n elements will have distinct identities with a probability of ϵ.

3.3 Kantarcioglu's Scheme

In [11], Kantarcioglu, Nix and Vaidya proposed the following cryptographic protocol using BF in an approximate algorithm for the threshold scalar (dot) product.

Let Y be a random variable representing the number of matching bits in the two BFs of S_A and S_B. That is, Y is defined by $Y = |B(S_A) \cap B(S_B)|$. There is

a positive correlation between X, defined by true size of intersection $S_{A\cap B}$, and Y, which enables us to predict X from Y which can be obtained from BFs in a secure way.

Based on the properties of BFs [2], Equation (2) is equivalent to

$$Z_A + Z_B + Z_{AB} \geq Z_A Z_B \frac{1}{m}(1 - \frac{1}{m})^{-kt}, \tag{3}$$

where Z_A (Z_B) is the number of 0s in $B(S_A)$ ($B(S_B)$), respectively. Z_{AB} is the number of matching 0s in the two BFs of S_A and S_B. That is, $Z_{AB} = m - |B(S_A) \cap B(S_B)| = m - Y$. To evaluate the inequality privately, Kantarcioglu et al. performs a secure protocol for the scalar product of two vectors [8] to obtain u_1 and u_2 such that $\boldsymbol{b}(S_A) \cdot \boldsymbol{b}(S_B) = m - Z_{AB} = u_1 + u_2$ and a secure protocol for the multiplication of two integers Z_A and Z_B to obtain v_1 and v_2 such that $v_1 + v_2 = (1 - 1/m)^{-kt}/mZ_AZ_B$. Finally, they use SFE for the shared comparison of two integers to test if $(Z_A + u_1 - m) + (Z_B + u_2) \geq (v_1 + v_2)$.

According to their experimental results [11], their approximation algorithm using BFs with $m = 3,000, k = 2$, and $n = 20,000$ ran in 4 minutes, whereas an *exact* version required 27 minutes.

3.4 Difficulties in ID-less Datasets

In [11], Kantarioglu et al. claim that as long as, $m \ll n$, their method would be much faster than the typical implementation of a secure scalar (dot) product protocol[1]. Their experimental results show that the accuracy of approximation increases as m increases[2]. We will show that these properties do not hold in our target, *ID-less datasets* model, where the two datasets have no consistent identities and hence n elements are specified with some unique attribute(s).

1. **(Accuracy)** The size of intersection is approximated in their scheme based on the expected value of probability of common bits in BFs. The accuracy is expected to be improved as m increases. However, this is not true in large m because that the vector becomes too sparse. To be adaptively dense vector, we must increase the number of hash functions, k. This is not trivial. In [11], the experimental behavior with some parameters were shown and no guarantee in accuracy.
2. **(Performance)** The size of BF, m, increases up to n^2 in ID-less datasets. As we discussed in Section 3.2, the range of hash function should be as large as n^2 in order to minimize the probability to fail to uniquely identify elements. This is too large to find the intersection since some schemes running in $O(n)$ complexity in private set intersection are known, e.g., [1], [5].

[1] In Section 2.2 (Computation and Communicational cost). In Section 3, they assume that the vector of 20000 elements, whose density was 10 %, that is, the vector contains 2000 1's ($= n$), and it performs 20000-dimensional vector's scalar product for exact match and $m = 3000$ BF for their scheme.

[2] In Section 3.1, Figure 1(b).

Table 1. Comparison between [11] and ours

item	[11]	Proposed
approximation	Equation (3)	Equation (7), (6)
priori distribution	–	Beta distribution
BF size (m)	large (n^2)	small ($n/\ln 2$)
accuracy	no guarantee	improved with Bayesian estimation from s tests

3. **(Overhead)** Their scheme requires the secure multiplication as well as scalar product. It is not necessary in private set intersection.

In later section, we will present our scheme which overcomes the above limitations. Table 1 gives a summary of comparison between the scheme in [11] and proposed scheme.

4 Proposed Scheme

4.1 Probability Distribution of Matching Bits in BFs

Suppose that given $S_{A\cap B} = S_A \cap S_B$, random variable X of the cardinality of $S_{A\cap B}$, and instance $x = X$, we wish to estimate the number of matching 1s bits in their two BFs, i.e., $y = |B(S_A) \cap B(S_B)|$. The quantity y is equal to the number of 1s values in the conjunction of the two BF vectors. This subsection presents the mathematical properties of BFs, which will be used to estimate X in the subsequent subsection.

An element a in $S_A \cup S_B$ belongs to either $S_{A\cap B}$ or $S_A \cup S_B - S_{A\cap B}$. The former case always ensures that $a \in B(S_A) \cap B(S_B)$. Therefore, the probability that a certain bit in the conjunction of BFs is 0 after k random bits are set to 1 is $q_X = (1 - \frac{1}{m})^{kx}$. In the latter case, an element in $S_A \cup S_B - S_{A\cap B}$ does not always have a value of 1 because it yields a false positive. That is, an element a in S_A can have the same hash value $H_i(a) = H_j(b)$ as some element $b \neq a$ in S_B. The probability that a certain bit is 0 in the BF for a in $S_A - S_{A\cap B}$ is $q_A = (1 - \frac{1}{m})^{k(n_A - x)}$. Similarly, the BF of an element in $S_B - S_{A\cap B}$ having a certain bit being 0 has a probability of $q_B = (1 - 1/m)^{(n_B - x)k}$. Therefore, the probability of a certain bit in the BF for $S_A \cup S_B - S_{A\cap B}$ being 1 is given by the product of the compliment of each event, namely $(1 - q_A)(1 - q_B) = 1 - q_A - q_B + q_A q_B$.

Because the conjunction of BF has 1 for a certain bit by being either an element of $S_{A\cap B}$ or $S_A \cup S_B - S_{A\cap B}$, we have the probability θ for a bit being 1 as the disjunction of the two events namely

$$\begin{aligned}\theta &= 1 - q_X(1 - (1 - q_A)(1 - q_B)) \\ &= 1 - (1 - \frac{1}{m})^{kn_A} - (1 - \frac{1}{m})^{kn_B} + (1 - \frac{1}{m})^{k(n_A + n_B - x)}. \end{aligned} \tag{4}$$

Consequently, the conditional probability of $Y = |B(S_A) \wedge B(S_B)|$ being y, given $x = |S_A \cap S_B|$, is given by the binomial distribution $B(m, \theta)$, of m independent binary events with success probability θ. That is,

$$Pr(Y = y|X = x) = \binom{m}{y}\theta^y(1-\theta)^{m-y}. \quad (5)$$

4.2 Bayesian Estimation of X

Given known parameter values and $Pr(X|Y)$, we wish to identify the posterior distribution $Pr(Y|X)$ using Bayes' rule.

One possible solution is an approximation based on a the likelihood value from a single observation, as described by Kantarcioglu et al. [11]. Their scheme suffers from the complexity of $O(m)$. That is, a secure scalar product will require m ciphertexts, which is greater than n. Moreover, the accuracy achieved is inadequate.

Instead, we will use recursive Bayesian estimation using several small BFs. That is more efficient because each individual BF used to perform the secure scalar product between two BFs will be smaller. Moreover, the iteration over multiple BFs improves the accuracy of the estimation. Given the properties of beta distribution, the iteration process can be performed with lightweight overheads.

Using the conjugate prior distribution of Equation (5), we assume a beta distribution $Be(\alpha, \beta)$, which gives

$$Pr(\theta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{\int_0^1 \theta^{\alpha-1}(1-\theta)^{\beta-1}dy}.$$

The initial prior distribution is given by $Be(1, 1)$, which yields a uniform distribution $Pr(\theta) = 1$. Using Bayes' theorem, we obtain the posterior probability of θ given y as

$$Pr(\theta|y) = \frac{Pr(\theta)Pr(y|\theta)}{\int Pr(\theta)Pr(y|\theta)d\theta} \propto Pr(\theta)Pr(x|\theta) \propto \theta^{\alpha-1+y}(1-\theta)^{\beta-1+m-y},$$

which results again in a beta distribution $Be(\alpha', \beta')$ with new parameters as $\alpha' = \alpha + y$, and $\beta' = \beta + m - y$.

Helicobacter Pylori infection is considered to be an event that occurs to each individual independently. Modeling such a situation with the binomial distribution is considered to be reasonable; beta distribution, the natural conjugate prior distribution of the binomial distribution, is used as the prior distribution in our protocol mainly due to its mathematical convenience. The initial prior was set to the non-informative uniform distribution in the experiments. Nonetheless, it is difficult to exclude the subjectivity from the settings of the prior distributions, and the obtained experimental results need to be carefully examined.

The mean of the beta distribution is denoted by $E[\theta] = \alpha/(\alpha + \beta)$. We can therefore estimate $\hat{\theta}$ when the BFs of two sets have y matching bits as follows, $\hat{\theta} = \frac{\alpha'}{\alpha'+\beta'} = \frac{1+y}{2+m}$. After estimating $\hat{\theta}$, the size of the intersection is given by the inverse of Equation (4), a mapping θ^{-1}, as

$$\hat{x} = n_A + n_B - \frac{1}{k}\log_{1-\frac{1}{m}}\left(\hat{\theta} - 1 + (1-\frac{1}{m})^{kn_B} + (1-\frac{1}{m})^{kn_A}\right). \quad (6)$$

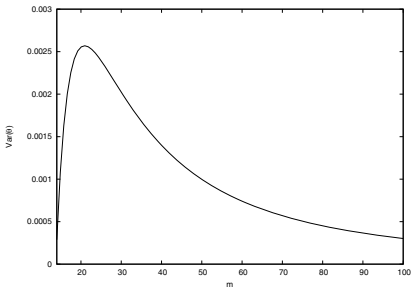


Fig. 1. (1) Distribution of the variance of $\hat{\theta}$, $Var[\theta]$, with respect to m, the size of the BF, for $n = 10, k = 3$, and $y = 14$

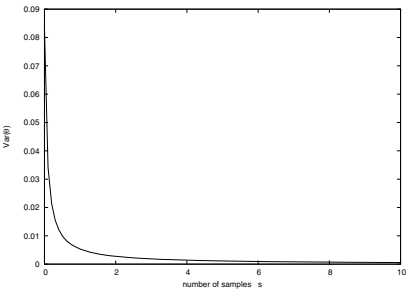


Fig. 2. (2) Distribution of the variance of $\hat{\theta}$, $Var[\theta]$, estimated from s independent BFs of the same size

The inverse mapping can be evaluated locally in the final stage of privacy preservation (without encryption). We are not concerned that if Equation (6) might appear complicated to evaluate.

4.3 "Bootstrap" of BFs

To improve the accuracy, there are two approaches.

(1) Enlarge the size of BF, m, and the estimate $\hat{\theta}$, [3]
(2) Estimate $\hat{\theta}$ from multiple observations of $Y_1, Y_2, \ldots, Y_s$.

Using a BF with more bits m could decrease the false positives in the membership test with the cost increasing as m. It is of interest that the value of m does not play a significant role in estimating of the intersection size, as we had expected. We will now show the mathematical properties that explain this observation.

(1) Variance of the Beta Distribution for a Large BF. According to the known variance of the beta distribution $Var[\theta] = \alpha\beta/((\alpha+\beta)^2(\alpha+\beta+1))$, we illustrate the change of variance with respect to m in Fig. 1. Since the variance determines the standard deviation, which provides a confidence interval for the estimation, we can predict the accuracy via the reduction in variance. Fig. 1 shows that the variance of $\hat{\theta}$ decreases slightly as m increases. However, the reduction in variance is not significant, given the increased cost of the required ciphertexts. For example, a BF with $m = 100$ requires 10 times more ciphertexts than that for an element in S with $n = |S| = 10$.

[3] We do not consider the number of hash functions k because there are some constraints between m and k, such as $kn < m$ and $k = (\ln 2)m/n$ for minimizing false positives.

(2) Variance from "Bootstrap" s Small BFs. Let $y_1, y_2, \ldots, y_s$ be the sequence of matching bits in s independent BFs for S_A and S_B. Recursive Bayesian estimation based on the sequence gives the posterior probability $Pr(\theta|y_1, \ldots, y_s)$ for the beta distribution $Be(\alpha', \beta')$ defined by

$$\alpha' = \alpha + \sum_{i=1}^{s} y_i, \qquad \beta' = \beta - \sum_{i=1}^{s} y_i + sm.$$

The estimation of $\hat{\theta}$ is provided from the mean of the beta distribution, namely

$$\hat{\theta} = \frac{\alpha + \sum_{i=1}^{s} y_i}{\alpha + \beta + sm} \tag{7}$$

Fig. 2 illustrates the reduction in the variance of $\hat{\theta}$. It implies that the bootstrapping reduces the confidence interval for the estimation of θ significantly with increasing s.

4.4 Proposed Scheme

We give the procedure for estimating the size of the intersection without revealing each set in Algorithm 2. At Step 1, both parties A and B compute BFs for their n-element sets S_A and S_B with parameters, size of BF m and the number of hash function k such that $k = (m/n) \ln 2$. For tradeoff between efficiency and accuracy, $k = 1$ and $m = n/\ln 2$ can be used. Since this process can be performed locally and the hash function performs very efficiently, we consider the overhead is negligible. Both parties participate the secure scalar product protocol (Algorithm 1), which is the most significant part in computation. The scalar product of two BFs, y, gives the number of common 1's bit in BFs, which can be divided into two integers, making the SFE possible to approximate $\hat{\theta}$ in Equation (7) without revealing any y_i. Step 4 is performed in public (or locally) after $\hat{\theta}$ reaches at convergence.

Instead of extend the size of BF, we perform the secure scalar product protocols multiple times to get the sequence of $y_1, y_2, \ldots, y_s$, which will be used to predict the $\hat{\theta}$ in Bayesian estimation. Both parties iterate the test until the expected accuracy is given. The confidence interval is given by the standard deviation of estimated value.

4.5 Security

The following theorem shows the security of Algorithm 2.

Theorem 1. *Suppose A and B behaves in the semi-honest model. Let S_A and S_B be inputs for Bloom Filter Bootstrap. Then, after execution of Bloom Filter Bootstrap, A and B learns random shares of y_i for $i = 1, \ldots, s$; nothing but y_i and what can be inferred from y_i is learned by both A and B.*

Algorithm 2. Bloom Filter Bootstrap $BFB(S_A, S_B)$

Input: S_A, S_B of n elements, m (size of the BF), k (number of hash functions).
Output: $\hat{x}$ (estimate of the size of the intersection of s_A and S_B).

1. A computes BF $\boldsymbol{b}(S_A)$ for S_A and B computes BF $\boldsymbol{b}(S_B)$.
2. A and B jointly perform Algorithm 1 to obtain $y_i = \boldsymbol{b}(S_A) \cdot \boldsymbol{b}(S_B)$ for $i = 1, \ldots, s$.
3. Estimate $\hat{\theta}$ using Equation (7).
4. Identify $\hat{x}$ using Equation (6).

Sketch of the Proof. Message exchange occurs only in step 2, so the security of step 2 is proved. Since step 2 is multiple invocation of the scalar product protocol, the security is reduced to that of the scalar product protocol. By following the security proof in [8], the security of Bloom Filter Bootstrap is immediately proved. Note that computation in step 4 is performed by A without communication with B, the security is not compromised by execution of these steps.

4.6 Complexity

We examine the complexities of our proposed scheme in terms of computation and communication costs. When these quantities are almost identical, we unify these by simply n. Protocols are compared in Table 2. In comparison with [11], we assume the ID-less model, where the size of BF can increase up to n^2.

Table 2 shows that the computational cost for A is linear to ms, while the cost for B is 0 (no modular exponentiation is required). Hence, it is preferable for outsourcing solution to our Requirement 3, where hospitals do not have powerful computational resources and become B in our protocol.

The protocols are classified into three groups. The first group is the scheme based on Oblivious Polynomial Evaluation. Scheme FNP[7] is designed to reveal not only the size of intersection but also the elements in the intersection. We show the performance for comparison purpose.

The second class, consisting of AES[1] and CT[5], is classified as Oblivious Pseudo-Random Functions (OPRF). AES depends on the commutative one-way function, while CT uses the RSA (Fig. 3 in [5]) and the blind RSA (Fig. 4) encryptions. The privacy of scheme (Fig. 3 in [5]) is proved as the view of honest-but-curious party is indistinguishable under the One-More Gap Diffie-Hellman assumption in the random oracle model.

Table 2. Complexity Comparison of protocols

	FNP[7]	AES[1]	CT[5]	KNV[11]	Proposed
primitives	OPE	commutative enc.	(blind) RSA	SSP w. BF	SSP w. BF
comp. at A	$n_A \log\log n_B$	$n_A + n_B$	$2n_A + 1$	m	ms
BF size	–	–	–	$n^2 \geq m > kn$	$m = n/\ln 2$
comp. at B	$n_B + n_A \log\log n_B$	$2n_A + n_B$	$n_A + n_B + 1$	0	0
complexity	$O(n_A \log\log n_B)$	$O(n)$	$O(n)$	$O(n^2)$	$O(n)$
comm. cost	$n_A + n_B$	$n_A + n_B$	$2n_A + n_B$	$m + 1$	$ms + 1$

OPE (Oblivious Polynomial Evaluation), SSP (Secure Scalar Product).

Table 3. Results of estimating X for various intersection sizes, x, for the dataset ($n_A = n_B = 100, m = 400, k = 3$)

x	20	40	60	80
$E[Y]$	125.24	141.45	160.98	184.11
$\sigma(Y)$	6.78	5.92	5.34	5.15
$E(\theta)$	0.31	0.35	0.40	0.46
$\hat{x}$	19.523	38.869	58.969	79.411

Table 4. Results of estimating X for various BF sizes, m for the dataset ($n_A = n_B = 100, x = 40$)

m	200	400	600	800
k	1	3	4	6
$E[Y]$	46.62	141.45	189.64	283.66
$\sigma(Y)$	3.146	5.923	6.436	7.488
$E(\theta)$	0.24	0.35	0.32	0.35
$\hat{x}$	39.490	38.869	39.604	39.227

The last class is based on BF and Secure Scalar Product schemes. KNV[11] uses a single BF with large size, while ours iterates s independent BFs with small size. The sizes are shown in Table.

5 Accuracy Evaluation

5.1 Simulation with DBLP Dataset

We evaluate the accuracy of the proposed scheme using a public dataset of author names, DBLP[4].

Four pairs of datasets S_A and S_B with $n_A = n_B = 100$ were chosen from DBLP with the intersection sizes $x = 20, 40, 60, 80$. Table 3 shows the experimental results for the estimation of x, for $x = 20, 40, 60$, and 80, where we used a BF with of size $m = 400$, a number of hash functions $k = 3$, and iterated the estimation s times. The results show that our scheme estimates the intersection within an error of ± 1. The numbers of matching bits in the BFs, Y, are distributed according to the binominal distribution. Note that all BFs estimate a size of the intersection close to the actual size of 40, but the differences are unstable.

5.2 Optimal BF Design

The accuracy of estimation depends on the size of BF, m, and the number of hash function, k, and the iteration of testing, s. In order to clarify the strategy for optimal accuracy, we examine the Mean Absolute Error (MAE) with respect to m and k. Figure 3 shows MAE in terms of m from 40 through 280, where $n_A = n_B = 100$, $x = 20$, $k = 1$ and $s = 20$. Figure 4 shows MAE with respect to $k = 1, \ldots, 6$ where $m = 200$. The MAE decreases as m increases, while the computational/communicational overhead increases accordingly. On the other hand, the increase of k does not reduce MAE.

A possible reason for the source of the error might be the restriction of m and k. As we discussed in Section 4.3, the optimal size for the BF is not trivial. We

[4] DBLP, A Citation Network Dataset, V1, (`http://arnetminer.org/citation`).

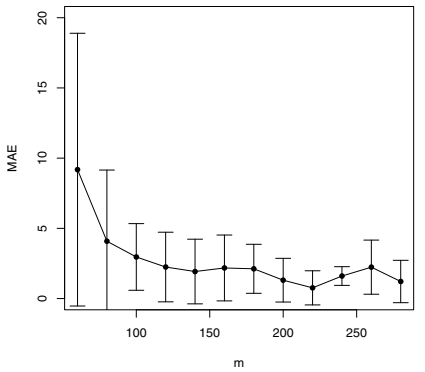


Fig. 3. Mean Absolute Error (MAE) with respect to the size of BF, m

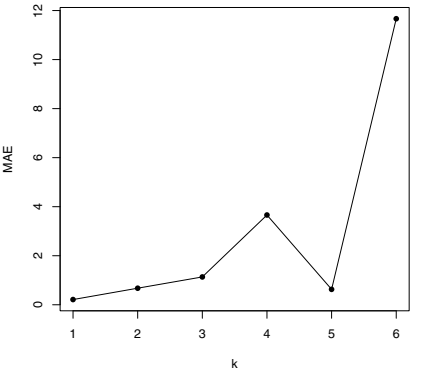


Fig. 4. Mean Absolute Error (MAE) with respect to the number of hash functions, k

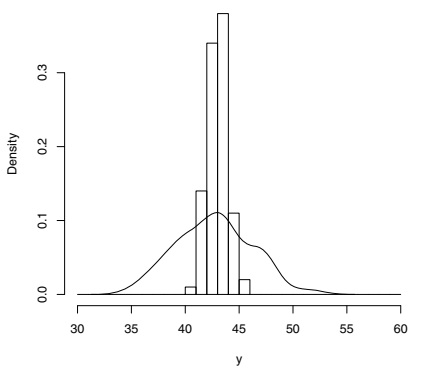


Fig. 5. Distributions of matching bits in the BFs, y, drown in solid line, and distribution of the means, $E[y]$, in bars

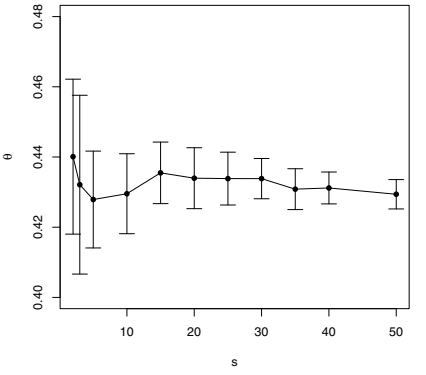


Fig. 6. Conversion of probability of 1s bit in BF, θ, with respect to iteration of BFs, s

therefore suggest choosing $k = 1$ first and then determining a near-optimal BF size by

$$m = kn/\ln 2 = 1 \cdot 100/\ln 2 = 144.26.$$

Since large m increases the computational cost at secure scalar product, we conclude minimize k, i.e., $k = 1$ and optimize $m = n/\ln 2$.

The accuracy can be improved by iteration of small BF tests rather than increasing the size of BFs. In fact, Figure 5 demonstrates the reduction of variance of observation of $E[Y]$, indicated by bar plot, when $s = 10$. The solid line represents the distribution of Y, which is widely distributed than that of $E[Y]$. It is known as Central Limit Theorem[14], that as s increases, the amount of sampling variation decreases. Figure 6 shows that the variance of estimated probability $\hat{\theta}$

reduces as the iteration s increases. The experiment shows even small $s = 10$ gives conversion of probability θ. The selection of optimal s can be made based on the variance of the prediction of θ. As we have showed in Section 4.3, the variance of beta distribution decreases with s, which determines the accuracy of approximation.

Finally, we obtain the estimate of intersection size, $\hat{x}$, by Equation 6. We illustrate the distribution of θ and the corresponding estimation of x.

5.3 Performance

We implemented the proposed scheme in Java, JDK 1.6, with BigInteger class. As additive homomorphic public key algorithm, we use Paillier cryptosystem with 1024 bit key. With platform of commodity PC, Intel Core (TM) i7-663DQM, 2 GHz, 4 GB, running Windows 7 (64 bit), the encryption runs in $t_e = 15.7$ [s], the decryption takes $t_d = 21.5$ [s] in average. The secure scalar product of 64-bit vectors ($n_A = n_B = 64, x = 5$) is performed in 5.28 [s], i.e., 82.5 [ms/element]. With this platform, the processing time to deal with the problem in [11], $n = 2000$, $k = 1$, and $m = n/\ln 2 = 2885$, is 4 minute and 125 second.

6 Privacy-Preserving Risk Analysis of *H. pylori*

Helicobacter pylori, or H. pylori, is a bacterium that is found in the stomachs of two-thirds of the world's population. Epidemiology studies have shown that individuals infected with *H. pylori* have an increased risk of cancer of the stomach[10,12].

Although *H. pylori* has been classified as a cancer-causing agent, it is not known how *H. pylori* infection increases the risk of cancer of the stomach. Some researchers have estimated that the risk of cancer the noncardiac region of the stomach is nearly six times higher for *H. pylori*–infected individuals than for uninfected people[9]. Some cohort studies revealed that the risk of gastric cardiac cancer among *H. pylori*–infected individuals was about one-third of that among uninfected individuals. The source of uncertainty is that the number of gastric cancers in the cohort study was too small to make a definitive statement. Cancer is a highly confidential matter and people will not reveal that they have it.

Our proposed methodology addresses the problem of epidemiology studies that preserve the privacy of the patients. The cryptographic protocol allows several small cohorts to be aggregated and analyzed for more certain evidence of increase or reduction of risk. Given two datasets of patients with cancer and *H. pylori*, the proposed protocol determines the size of the intersection of the two sets without revealing any entries in the datasets. With a secure hash function, the proposed scheme identifies a patient from their personal attributes.

6.1 Contingency Tables

The epidemiology study aims to determine whether an *H. pylori*-infected individual has increased the risk of gastric cancer. The evidence is shown by a measure of *relative risk* (RR), the probability of disease among exposed individuals divided by the probability of disease among the unexposed. Suppose that a sample of N individuals is arranged in the form of the 2×2 contingency table in Table 5; the relative risk (RR) of *H. pylori* is estimated by

$$RR = \frac{\Pr(\text{ cancer } | \text{ H. pylori})}{\Pr(\text{cancer } | \text{unexposed})} = \frac{a}{a+b} \Big/ \frac{c}{c+d} \approx \frac{ad}{bc},$$

where we assume $a \ll b$ and hence $a + b = b$.

To examine whether *H. pylori*-infection increases the risk of cancer, i.e., $RR > 1$, we test the null and the alternative hypotheses.

H_0**:** The proportion of patients with cancer among individuals infected with *H. pylori* is equal to the proportion of patients with cancer among those uninfected.

H_A**:** The proportions of patients with cancer are not identical in the two populations.

The chi-square test compares the observed frequencies in each category of the contingency table, O, with the expected frequencies given that the null hypothesis is true, E. To perform the test, we calculate the sum

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} = \frac{(N-1)\left((ad-bc) \pm N/2\right)^2}{(a+c)(b+d)(a+b)(c+d)},$$

where k is the number of cells in the table. The probability distribution of this sum is approximated by a χ^2 distribution with $(2-1)(2-1) = 1$ degree of freedom. Alternatively, by taking it squire root, we may assume that χ is normally distributed with mean 0 and standard deviation 1.

Table 5. 2×2 Contingency table for *H. pylori* and stomach cancer

H. pylori	Cancer	No cancer	total
Yes	a	b	$a+b$
No	c	d	$c+d$
total	$a+c$	$b+d$	N

Table 6. Chiba Cancer Center dataset `CAN`

year	male	female	total
2003	2,330	1,134	3,464
2004	2,610	1,242	3,852
2005	2,559	1,205	3,763
total	7,500	3,581	11,081

Table 7. MHW dataset of *H. pylori* infections `PYL`

year	male	female	total
2001	2,671	5,206	7,877

6.2 Datasets

In our experiment, we have two datasets collected by independent agencies.

1. Patients with gastric cancer `CAN`.
   The Chiba Cancer Center has performed an epidemiology study of causes and effects of cancer conditions since 1975 in Chiba Prefecture, Japan. Table 6 shows the statistics for three years from 2003, used in this study. The dataset contains private attributes, including name, gender, birthday, mailing address, ZIP code, and medical treatments, e.g., patient ID, days of operations, day of death, type of cancers, and degree of tumor differentiation.
2. Individuals infected with *H. pylori* `PYL`.
   The Japanese Ministry of Health and Welfare (MHW) carried out a medical examination in 2001 in a small village in Chiba Prefecture. The dataset contains the number of *H. pylori*-infected individuals but their cancer status is not known.

6.3 Hypothesis Testing

Our proposed algorithm estimates the size of the intersection of the two datasets, thus allowing the estimation of relative risk of *H. pylori*.

The statistics show that the population in Chiba Prefecture in 2003 was $6,056,462$ ($3,029,486$ male). The dataset in Table 6 has $n_A = 7401$ recodes of patients with cancer. Table 7 contains $n_B = 2629$ individuals infected with *H. pylori*. We apply a BF with size $m = 14,000$, $k = 1$ and $s = 10$ to the two datasets and obtain the scalar product, $y = \boldsymbol{b}(\texttt{CAN}) \cdot \boldsymbol{b}(\texttt{PYL})$ as $\mu(y) = 1023.9$ on average. Based on Bayes' theorem, we estimate the probability $\hat{\theta}$ in Equation (7) as

$$\hat{\theta} = \frac{\alpha + \sum^{s} y_i}{\alpha + \beta + sm} = 0.073142.$$

From Equation (6), $\hat{x} = 81.1702$, while the exact size of the intersection is 80. The number of individuals who are infected with *H. pylori* but do not have is therefore $n_a - \hat{x} = 2549$. The other values can be obtained similarly. Finally, the numbers of individuals are summarized in Table 8.

Table 8. Experimental results for `CAN` and `PYL`

H. pylori	Cancer	No cancer	total
Yes	80	2,549	2,629
No	7,321	2,990,050	2,997,371
total	7,401	2,992,599	3,000,000[5]

[5] The number is referred from statistics in Chiba prefecture. There are potential individuals infected by H. Pylori who was not counted in the table.

An estimate of the relative risk of having cancer among *H. pylori*-infected individuals is therefore

$$RR = \frac{80 \cdot 222,964}{2,549 \cdot 7,321} = 12.81.$$

The chi-square test of the null hypothesis yields

$$\begin{aligned}\chi &= \frac{\sqrt{3,000,000-1}(80 \cdot 222,964 - 2,549 \cdot 7,321 - 3,000,000/2)}{\sqrt{7,401 \cdot 2,992,599 \cdot 2,629 \cdot 230,285}} \\ &= 28.71 > N(.05/2) = 1.960,\end{aligned}$$

which is too high to assume the null hypothesis. Therefore, we reject the null hypothesis at the 0.05 level of confidence.

In the experiment in Intel Xeon E5620 2.40 GHz, Memory 16GB, the processing of the BF takes 17,030 second (=4.7 hour), while the naive ID generation requires a scalar product of $n^2 = 4.9 \times 10^7$, which is estimated to be 223 hours.

7 Conclusions

We have proposed an efficient algorithm for the estimation of the size of the intersection of two private sets. The proposed scheme gives a Bayesian estimation of the intersection size based on the mathematical properties of the number of matching bits in two BFs. A well-known secure scalar product protocol enables us to evaluate the number of matching bits in a privacy-preserving way and to test hypothesizes that are useful in epidemiological studies. We have shown the properties of the accuracy of estimation for various parameters and the experimental results for the DBLP public dataset. One of our main results is that the bootstrap approach, iterating small BFs several times, is better than using a single large BF.

The extension of scalar product protocol to multiple parties can be done by replacing the Step 3 as that Bob forwards n ciphertexts computed with his secret vector as $E(x_1)^{y_1}, \ldots, E(x_n)^{y_n}$ to Carol who then perform the original Step 3 as $c = E(x_1)^{y_1 z_1} \cdots E(x_n)^{y_n z_n} / E(s_B)$. The extension of Bloom filter to multiple parities is not trivial and one of our future work.

References

1. Agrawal, R., Evfimievski, A., Srikant, R.: Information sharing across private databases. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 86–97. ACM Press (2003)
2. Broder, A., Mitzenmacher, M.: Network applications of bloom filters: A survey. In: Internet Mathematics, pp. 636–646 (2002)
3. Camenisch, J., Zaverucha, G.M.: Private intersection of certified sets. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 108–127. Springer, Heidelberg (2009)

4. Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., Zhu, M.Y.: Tools for privacy preserving distributed data mining. ACM SIGKDD Explorations Newsletter 4(2), 28–34 (2002)
5. De Cristofaro, E., Tsudik, G.: Practical private set intersection protocols with linear complexity. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 143–159. Springer, Heidelberg (2010)
6. Fan, L., Cao, P., Almeida, J., Broder, A.Z.: Summary cache: a scalable wide-area web cache sharing protocol. IEEE/ACM Trans. Netw. 8(3), 281–293 (2000)
7. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
8. Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On private scalar product computation for privacy-preserving data mining. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 104–120. Springer, Heidelberg (2005)
9. Helicobacter and Cancer Collaborative Group. Gastric cancer and helicobacter pylori: a combined analysis of 12 case control studies nested within prospective cohorts. Gut. 49(3), 347–353 (2001)
10. Atherton, J.C.: The pathogenesis of helicobacter pylori-induced gastro-duodenal diseases. Review of Pathology 1, 63–96 (2006)
11. Kantarcioglu, M., Nix, R., Vaidya, J.: An efficient approximate protocol for privacy-preserving association rule mining. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 515–524. Springer, Heidelberg (2009)
12. Kuipers, E.J., Kusters, J.G., van Vliet, A.H.: Pathogenesis of helicobacter pylori infection. Clinical Microbiology Reviews 19(3), 449–490 (2006)
13. Lu, H., He, X., Vaidya, J., Adam, N.R.: Secure construction of contingency tables from distributed data. In: Atluri, V. (ed.) DAS 2008. LNCS, vol. 5094, pp. 144–157. Springer, Heidelberg (2008)
14. Pagano, M., Gauvreau, K., Pagano, M.: Principles of biostatistics. Brooks/Cole (2000)
15. Ravikumar, P., Ravikumar, P., Fienberg, S.E., Cohen, W.W.: A secure protocol for computing string distance metrics. In: PSDM (2004)
16. Sakuma, J., Wright, R.N.: Privacy-preserving evaluation of generalization error and its application to model and attribute selection. In: Zhou, Z.-H., Washio, T. (eds.) ACML 2009. LNCS, vol. 5828, pp. 338–353. Springer, Heidelberg (2009)
17. Vaidya, J., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data. In: The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 639–644 (2002)

Using Safety Constraint for Transactional Dataset Anonymization

Bechara Al Bouna[1], Chris Clifton[2], and Qutaibah Malluhi[1]

[1] Department of Computer Science and Engineering, Qatar University, Qatar
[2] Dept. of Computer Science/CERIAS, Purdue Univ., West Lafayette, Indiana, USA
bechara.albouna@qu.edu.qa,
clifton@cs.purdue.edu, qmalluhi@qu.edu.qa

Abstract. In this paper, we address privacy breaches in transactional data where individuals have multiple tuples in a dataset. We provide a safe grouping principle to ensure that correlated values are grouped together in unique partitions that enforce l-diversity at the level of individuals. We conduct a set of experiments to evaluate privacy breach and the anonymization cost of safe grouping.

1 Introduction

Data outsourcing is on the rise, and the emergence of cloud computing provides additional benefits to outsourcing. Privacy regulations pose a challenge to outsourcing, as the very flexibility provided makes it difficult to prevent against trans-border data flows, protection and separation of clients, and other constraints that may be required to outsource data. An alternative is encrypting the data [5]; while this protects privacy, it also prevents beneficial use of the data such as value-added services by the cloud provider (e.g., address normalization), or aggregate analysis of the data (and use/sale of the analysis) that can reduce the cost of outsourcing. Generalization-based data anonymization [18,19,12,9] provides a way to protect privacy while allowing aggregate analysis, but doesn't make sense in an outsourcing environment where the client wants to be able to retrieve the original data values.

An alternative is to use *bucketization*, as in the anatomy [23], fragmentation [4], or slicing [11] models. Such a database system has been developed [15,16]. The key idea is that identifying and sensitive information are stored in separate tables, with the join key encrypted. To support analysis at the server, data items are grouped into buckets; the mapping between buckets (but not between items in the bucket) is exposed to the server. An example is given in Figure 1 where attribute *DrugName* is sensitive: Figure 1b is an anatomized version of table prescription with attributes separated into $Prescription_{QIT}$ and $Prescription_{SNT}$.

The bucket size and grouping of tuples into buckets ensures privacy constraints (such as k-anonymity[18,19] or l-diversity[12]) are satisfied.

Complications arise when extending this approach to transactional datasets. Even with generalization-based approaches, it has been shown that transactions introduce new challenges. While approaches as (X, Y)-privacy [21] and

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 164–178, 2013.

Name	Country	Drug Name
Roan (P1)	United States	Mild Exfoliation
Lisa (P4)	Columbia	Azelaic acid
Roan (P1)	United States	Retinoic Acid
Elyse (P2)	United States	Mild Exfoliation
Carl (P3)	France	Azelaic acid
Roan (P1)	United States	Retinoic Acid
Lisa (P4)	Columbia	Cytarabine
Roan (P1)	United States	Azelaic acid
Lisa (P4)	Columbia	Retinoic Acid
Carl (P3)	France	Cytarabine
Carl (P3)	France	Azelaic acid
Roan (P1)	United States	Retinoic Acid
Bob (P5)	Columbia	Esom. Magnesium
Carl (P3)	France	Mild Exfoliation
Alice (P6)	United States	Adapalene

(a) Original Prescription table

Name	Country	GID
Roan (P1)	United States	1
Lisa (P4)	Columbia	1
Roan (P1)	United States	1
Elyse (P2)	United States	2
Carl (P3)	France	2
Roan (P1)	United States	2
Lisa (P4)	Columbia	3
Roan (P1)	United States	3
Lisa (P4)	Columbia	3
Carl (P3)	France	4
Carl (P3)	France	4
Roan (P1)	United States	4
Bob (P5)	Columbia	5
Carl (P3)	France	5
Alice (P6)	United States	5

GID	Drug Name
1	Mild Exfoliation
1	Azelaic acid
1	Retinoic Acid
2	Mild Exfoliation
2	Azelaic acid
2	Retinoic Acid
3	Cytarabine
3	Azelaic acid
3	Retinoic Acid
4	Cytarabine
4	Azelaic acid
4	Retinoic Acid
5	Esom. Magnesium
5	Mild Exfoliation
5	Adapalene

(b) $Prescription_{QIT}$ and $Prescription_{SNT}$

Fig. 1. Table Prescription anonymized

k^m-anonymity [20] include restrictions on the correlation of quasi-identifying values and can be used to model transactional data [3], they still face limitations when applied to bucketization approaches.

We give examples of this based on Figure 1b. The anonymized table satisfies the (X,Y)-privacy and (2,2)-diversity privacy constraints[13]; given the 2-diverse table, an adversary should at best be able to link a patient to a drug with probability 1/2.

Inter-group dependencies occur when an adversary knows certain facts about drug use, e.g., Retinoic Acid is a maintenance drug taken over a long period of time. As P1 is the only individual who appears in all groups where Retinoic Acid appears, it is likely that P1 is taking this drug. Note that this fact can either be background knowledge, or learned from the data.

Intra-group dependencies occur where the number of transactions for a single individual within a group results in an inherent violation of l-diversity (this would most obviously occur if all transactions in a group were for the same individual.) By considering this separately for transactional data, rather than simply looking at all tuples for an individual as a single "data instance", we gain some flexibility.

We present a method to counter such privacy violations while preserving data utility. Our contributions can be summarized as follows:

- An in-depth study of privacy violation due to correlation of individuals' related tuples in *bucketization* techniques.
- A safe grouping technique to eliminate privacy violation. Our safe grouping technique ensures that quasi-identifier and sensitive partitions respect l-diversity privacy constraint.

The approach is based on knowing (or learning) the correlations, and forming buckets with a common antecedent to the correlation. This protects against

inter-group dependencies. Identifiers are then suppressed where necessary (in an outsourcing model, this corresponds to encrypting just the portion of the tuple in the identifier table) to ensure the privacy constraint is met (including protection against intra-group correlation.)

In the next section, we present our adversary model. Section 3 gives further background on prior work and its limitations in dealing with this problem. In Section 4, we define the basic notations and key concepts used in the rest of the paper. A definition of correlation-based privacy violation in transactional datasets is given in Section 5. In Section 6, we present our a safe grouping constraint that forms the basis of our anonymization method. Section 7 gives a safe grouping algorithm. A set of experiments to evaluate both the practical efficiency and the loss of data utility (suppression/encryption) is given in Section 8. We conclude with a discussion of next steps to move this work toward practical use.

2 Adversary Model

In our adversary model, we assume that the adversary has knowledge of the transactional nature of the dataset. We also assume that he/she has outside information on correlations between sensitive data items that leads to a high probability that certain sets of items would belong to the same individual. This is illustrated in the introduction (example 1) where the fact that the drug Retinoic Acid is known to be taken for a long period of time makes it possible to link it to patient P1.

We do not care about the source of such background information; it may be public knowledge, or it may be learned from the anatomized data itself. (We view learning such knowledge from the data as beneficial aggregate analysis of the data.)

3 Related Work

In [21], the authors consider that any transaction known by the adversary could reveal additional information that might be used to uncover a sensitive linking between a quasi-identifier and a sensitive value. They define (X,Y)-*privacy* to ensure on one hand that each value of X is linked to at least k different values of Y, and on the other hand, no value of Y can be inferred from a value of X with confidence higher than a designated threshold. A similar approach proposed in [20] in which the authors extend k-anonymity with k^m-anonymity requiring that each combination of at most m items appears in at least k transactions, where m is the maximum number of items per transaction that could be known by the adversary. (Also related is the problem of trail re-identification[14].) As demonstrated in the example in Figure 1b, these techniques are limited when it comes to *bucketization*, as more subtle in *intra* and *intra* group correlations may lead to a breach of l-diversity.

In [11] the authors proposed a slicing technique to provide effective protection against membership disclosure, but it still remains vulnerable to identity disclosure. An adversary with knowledge of the transactional nature of the dataset may still be able to associate an individual identifier to correlated sensitive values. The authors in [6] discuss privacy violations in the anatomy privacy model [23] due to functional dependencies (FDs). In their approach, they propose to create QI-groups on the basis of a FD tree while grouping tuples based on the sensitive attribute to form l-diverse groups. Unfortunately, dealing with FDs' is not sufficient, as less strict dependencies can still pose a threat.

In [22], the authors consider correlation as foreground knowledge that can be mined from anonymized data. They use the possible worlds model to compute the probability of associating an individual to a sensitive value based on a global distribution. In [8], a Naïve Bayesian model is used to compute association probability. They used exchangeability [1] and DeFinetti's theorem [17] to model and compute patterns from the anonymized data. Both papers address correlation in its general form where the authors show *how* an adversary can violate l-diversity privacy constraint through an estimation of such correlations in the anonymized data. As it is a separate matter, we consider that correlations due to transactions where multiple tuples are related to the same individual ensure that particular sensitive values can be linked to a particular individual when correlated in the same group (i.e., bucket). We go beyond this, addressing any correlation (either learned from the data or otherwise known) that explicitly violates the targeted privacy goal.

4 Formalization

Given a table T with a set of attributes $\{A_1, ..., A_b\}$, $t[A_i]$ refers to the value of attribute A_i for the tuple t. Let U be the set of individuals of a specific population, $\forall u \in U$ we denote by T_u the set of tuples in T related to the individual u. Attributes of a table T that we deal with in this paper are divided as follows;

- *Identifier* (A^{id}) represents an attribute that can be used (possibly with external information available to the adversary) to identify the individual associated with a tuple in a table. We distinguish two types of identifiers; sensitive and nonsensitive. For instance, the attribute *Social Security Number* is a *sensitive identifier*; as such it must be suppressed (encrypted). *Nonsensitive identifiers* are viewed as public information, and include both direct identifiers such as *Patient ID* in Figure 4, and quasi-identifiers that in combination may identify an individual (such as <*Gender, Birthdate, Zipcode*>, which uniquely identifies many individuals.)
- *Sensitive attribute* (A^s) contains sensitive information that must not be linkable to an individual, but does not inherently identify an individual. In our example (Table 1a), the attribute *DrugName* is considered sensitive and should not be linked to an individual.

Definition 1 (Equivalence class / QI-group). *[18] A quasi-identifier group (*QI-group*) is defined as a subset of tuples of* $T = \bigcup_{j=1}^{m} QI_j$ *such that, for any* $1 \leq j_1 \neq j_2 \leq m$, $QI_{j1} \cap QI_{j2} = \phi$.

Note that for our purposes, this can include direct identifiers as well as quasi-identifiers; we stick with the QI-group terminology for compatibility with the broader anonymization literature.

Definition 2 (l-diversity). *[13] a table T is said to be l-diverse if each of the QI-groups $QI_j (1 \leq j \leq m)$ is l-diverse; i.e., QI_j satisfies the condition $c_j(v_s)/|QI_j| \leq 1/l$ where*

- *m is the total number of QI-groups in T*
- *v_s is the most frequent value of A^s*
- *$c_j(v_s)$ is the number of tuples of v_s in QI_j*
- *$|QI_j|$ is the size (number of tuples) of QI_j*

Definition 3 (Anatomy). *Given a table T, we say that T is anatomized if it is separated into a quasi-identifier table (T_{QIT}) and a sensitive table (T_{SNT}) as follows:*

- *T_{QIT} has a schema $(A_1, ..., A_d, GID)$ where A_i $(1 \leq i \leq d)$ is either a nonsensitive identifying or quasi-identifying attribute and GID is the group id of the QI-group.*
- *T_{SNT} has a schema (GID, A^s_{d+1}) where A^s_{d+1} is the sensitive attribute in T.*

To express correlation in transactional data we use the following notation cd^{id} : $A^{id}_1, ..., A^{id}_n \dashrightarrow A^s$ where A^{id}_i is a nonsensitive identifying attribute and A^s is a sensitive attribute, and cd^{id} is a correlation dependency between attributes $A^{id}_1, ..., A^{id}_n$ on one hand, and A^s on the other.

Next, we present a formal description of the privacy violation that can be caused due to such correlations.

Table 1. Notations

T	a table containing individuals related tuples
t_i	a tuple of T
u	an individual described in T
T_u	a set of tuples related to individual u
A	an attribute of T
A^{id}	an identifying attribute of T
A^s	a sensitive attribute of T
QI_j	a quasi-identifier group
T^*	Anonymized version of table T

5 Correlation-Based Privacy Violation

Inter-group correlation occurs when transactions for a single individual are placed in multiple QI-groups (as with P1, P3, and P4 in Figure 1a). The problem arises when the values in different groups are related (as would happen with association rules); this leads to an implication that the values belong to the same individual. Formally:

Definition 1 (Inter QI-group Correlation). *Given a correlation dependency of the form* $cd^{id} : A^{id} \dashrightarrow A^s$ *over* T^**, we say that a privacy violation might exists if there are correlated values in a subset* $QI_j (1 \leq j \leq m)$ *of* T^* *such that* $v_{id} \in \pi_{A^{id}} QI_1 \cap ... \cap \pi_{A^{id}} QI_m$ *and* $|\pi_{A^s} QI_1 \cap ... \cap \pi_{A^s} QI_m| < l$ *where* $v_{id} \in A^{id}$ *is an individual identifying value,* l *is the privacy constant and an adversary knows of that correlation.*

The example shown in Figure 1, explains how an adversary with prior knowledge of the correlation, in this case that Retinoic Acid must be taken multiple times, is able to associate the drug to the patient Roan (P1) due to their correlation in several QI-groups. (The same would also apply to different drugs that must be taken together.)

An intra-group violation can arise if several correlated values are contained in the same QI-group. Here the problem is that this gives a count of tuples that likely belong to the same individual, which may limit it to a particular individual in the group. Figure 2 is an example of Intra QI-group Correlation, formally defined as follows:

Patient ID	GID
Roan (P1)	1
Roan (P1)	1
Roan (P1)	1
Carl (P3)	1
Carl (P3)	1
Roan (P1)	1

GID	Drug Name
1	Retinoic Acid
1	Retinoic Acid
1	Retinoic Acid
1	Azelaic acid
1	Azelaic acid
1	Azelaic acid

Fig. 2. Intra QI-group correlation

Lemma 1 (Intra QI-group Correlation). *Given a QI-group* QI_j *(*$1 \leq j \leq m$*) in* T^* *that is* l*-diverse, we say that a privacy violation might occur if individual's related tuples are correlated in* QI_j *such that* $f(QI_j, u) + c_j(v_s) > |QI_j|$ *where*

- v_s *is the most frequent* A^s *value in* QI_j*,*
- $c_j(v_s)$ *is the number of tuples* $t \in QI_j$ *with* $t[A^s] = v_s$*,*
- u *is the individual who has the most frequent tuples in* QI_j*,*
- $f(QI_j, u)$ *is a function that returns the number of* u *'s related tuples in* QI_j
- $|QI_j|$ *is the size of* QI_j *(number of tuples contained in* QI_j*)*

Proof. Let r be the number of remaining sensitive values in QI_j, $r = |QI_j| - c_j(v_s)$. If $f(QI_j, u) + c_j(v_s) > |QI_j|$, this means that $f(QI_j, u) > |QI_j| - c_j(v_s)$

and therefore $f(QI_j, u) > r$. That is, there are e tuples related to individual u such that $f(QI_j, u) = e$ to be associated to r sensitive values of QI_j where $e > r$. According to the pigeon-hole principle, at least a tuple t of T_u will be associated to the sensitive value v_s which leads to a privacy violation. □

It would be nice if this lemma was "if and only if", giving criteria where a privacy violation would NOT occur. Unfortunately, this requires making assumptions about the background knowledge available to an adversary (e.g., if an adversary knows that one individual is taking a certain medication, they may be able to narrow the possibilities for other individuals). This is an assumption made by all k-anonymity based approaches, but it becomes harder to state when dealing with transactional data.

Let us go back to Figure 2, an adversary is able to associate both drugs (Retinoic Acid and Azelaic Acid) to patient Roan (P1) due to the correlation of their related tuples in the same QI-group.

In the following, we provide an approach that deals with such privacy violations.

6 Safe Grouping for Transactional Data

As we have shown in the previous section, bucketization techniques do not cope well with correlation due to transactional data where an individual might be represented by several tuples that could lead to identify his/her sensitive values. In order to guarantee safety, we present in this section our safe grouping safety constraint .

Safety Constraint (Safe Grouping)**.** *Given a correlation dependency in the form of* $cd^{id} : A^{id} \dashrightarrow A^s$*, safe grouping is satisfied iff*

1. $\forall u \in U$*, the subset* T_u *of* T *is contained in one and only one quasi identifier group* QI_j $(1 \leq j \leq m)$ *such that* QI_j *respects* l*-diversity and contains at least* k *subsets* $T_{u_1}, ..., T_{u_k}$ *where* $u_1, ..., u_k$ *are* k *distinct individuals of the population and,*
2. $Pr(u_{i_1}|QI_j) = Pr(u_{i_2}|QI_j) \leq 1/l$ *where* $u_{i_1}, u_{i_2}, i_1 \neq i_2$ *are two distinct individuals in* QI_j *with* $(1 \leq i \leq k)$ *and* $Pr(u_i|QI_j)$ *is the probability of* u_i *in* QI_j*.*

Safe grouping ensures that individual tuples are grouped in one and only one QI-group that is at the same time l-diverse, respects a minimum diversity for identity attribute values, and every subset T_u in QI_j are of equal number of tuples.

Figure 3 describes a quasi identifier group (QI_1) that respects safe grouping where on one hand, we assume that there are no other QI-groups containing $P1$ and $P3$ and on the other hand, two tuples from T_{P1} are anonymized to guarantee that $Pr(P1|QI_1) = Pr(P3|QI_1) \leq 1/2$. Note that we have suppressed some data in order to meet the constraint; this is in keeping with the model in [15] where some data is left encrypted, and only "safe" data is revealed.

Lemma 1. *Let QI_j for ($1 \leq j \leq m$) be a QI-group that includes k individuals, if QI_j satisfies safe grouping then k is at least equal to l*

Proof. Consider an individual u in QI_j, according to the safe grouping, $Pr(u|QI_j) \leq 1/l$. Or $Pr(u|QI_j)$ is equal to $f(QI_j, u)/|QI_j|$ where $f(QI_j, u) = |QI_j|/k$ represents the number of individual's u related tuples in QI_j. Hence, $1/k \leq 1/l$ and $k \geq l$ □

Corollary 1 (Correctness). *Given an anonymized table T^* that respects safe grouping, and a correlation dependency of the form $cd^{id} : A^{id} \dashrightarrow A^s$, an adversary cannot correctly associate an individual u to a sensitive value v_s with a probability $Pr(A^s = v_s, u|T^*)$ greater than $1/l$.*

Proof. Safe grouping guarantees that individual's u related tuples T_u are contained in one and only one QI-group (QI_j), which means that possible association of u to v_s is limited to the set of correlated values that are contained in QI_j. Hence, $Pr(A^s = v_s, u|T^*)$ can be written as $Pr(A^s = v_s, u|QI_j)$. On the other hand,
$Pr(A^s = v_s, u|QI_j) = \frac{Pr(A^s=v_s,u)}{\sum_{i=1}^{k} Pr(A^s=v_s,u_i)}$ where k is the number of individuals in QI_j and $Pr(A^s = v_s, u_i)$ is the probability of associating individual u_i to a sensitive value v_s. Recall that safe grouping guarantees that for a given individual u_i, $Pr(A^s = v_s, u_i)$ is at the most equal to $1/l$. Summarizing, $Pr(A^s = v_s, u|QI_j)$ is at the most equal to $1/k$ where $k \geq l$ according to Lemma 1. □

Presc ID	Patient ID	GID
1	Roan (P1)	1
5	Roan (P1)	1
7	Roan (P1)	1
2	Roan (P1)	1
10	Carl (P3)	1
12	Carl (P3)	1
9	Carl (P3)	1
6	Carl (P3)	1
8	*	2
11	*	2
3	Lisa (P4)	2
4	Elyse (P2)	2
13	Bob (P5)	2

GID	Drug Name
1	Retinoic Acid
1	Retinoic Acid
1	Retinoic Acid
1	Azelaic acid
1	Azelaic acid
1	Azelaic acid
1	Mild Exfoliation
1	Nexium
2	Adapalene
2	Cytarabine
2	Cytarabine
2	Mild Exfoliation
2	Mild Exfoliation

Fig. 3. Table Prescription respecting our safety constraint

We can estimate[1], for example, $Pr(A^s = RetinoicAcid, A^{id} = P1|T^*)$ to be 4/5 where it is possible to associate Roan (P1) to Retinoic Acid in 4 of 5 QI-groups as shown in Figure 1b. However, as you can notice from Figure 3, safe grouping guarantees that $Pr(A^s = RetinoicAcid, A^{id} = P1|T^*)$ remains limited to the possible association of values in QI_1 and thus bounded by l-diversity.

[1] $Pr(A^s = RetinoicAcid, A^{id} = P1|T^*)$ as calculated remains an estimation where a much deeper aspect on how to calculate the exact probability of values correlated across QI-groups can be seen in [22] and [8]

The safe grouping constraint is restrictive, but may be necessary. While we do not have a formal proof that it is optimal, we can find examples where any straightforward relaxation can result in a privacy violation (we do not elaborate due to space constraints.)

We note that using safe grouping, we do not intend to replace anatomy. In fact, we preserve table decomposition as described in the original anatomy model by separating a table T into two subtables (T_{QIT}, T_{SNT}) while providing a safe grouping of tuples on the basis of the attributes related by a correlation dependency.

7 Safe Grouping Algorithm

In this section, we provide an algorithm to enforce ensure safe grouping for transactional data. The algorithm guaranties the safe grouping of a table T based on an identity attribute correlation dependency $cd^{id} : A^{id} \dashrightarrow A^s$ ($A^{id} \in T_{QIT}$ and $A^s \in T_{SNT}$).

The main idea behind the algorithm is to create k buckets based on the attribute (A^{id}) defined on the left hand side of a correlation dependency in a reasonable time.

The safe grouping algorithm takes a table T, a correlation dependency $A^{id} \dashrightarrow A^s$, a constant l to ensure diversity, and a constant k representing the number of individuals (individuals' related tuples) to be stored in a QI-group. It ensures a safe grouping on the basis of the attribute A^{id}. In Step 2, the algorithm hashes the tuples in T based on their A^{id} values and sorts the resulting buckets. For any individual, all their values will end up in the same bucket. In the group creation process from steps 4-17, the algorithm creates a QI-group with k individuals. If the QI-group respects l-diversity the algorithm adds it to the list of QI-groups and enforces the safety constraint in step 8 by anonymizing tuples in T_{QIT} including values that are frequently correlated in the QI-group. In other terms, it makes sure that individuals' related tuples in the QI-group are of equal number.

If l-diversity for the QI-group in question is not met, the algorithm enforces it by anonymizing tuples related to the most frequent sensitive value in the QI-group. After the l-diversity enforcement process, the algorithms verifies whether the group contains k buckets, and if not anonymizes (which could mean generalizing, suppressing, or encrypting the values, depending on the target model.)

From steps 19 to 26 the algorithm anatomizes the tables based on the QI-groups created. It stores random sensitive attribute values in the T_{SNT} table.

While safe grouping provides safety, its ability to preserve data utility is limited to the number of distinct values of A^{id} attribute.

Algorithm 1. SafeGrouping

Require: a table T, $cd^{id} : A^{id} \dashrightarrow A^s$, l, k, $minConf$, $maxConf$ and exp
Ensure: safe grouping for T

1: $T_{QIT} = \varnothing$; $T_{SNT} = \varnothing$; $gn = 0$; $i = 0, j = 0$;
2: Hash the tuples in T by their A^{id} values (one bucket per A^{id} value)
3: Sort the set of $Buckets$ based on their number of tuples.
4: **while** there are k groups $QI \in Buckets$ **do**
5: **if** QI is l-diverse **then**
6: $gn = gn + 1$
7: $QI_{gn} = QI$
8: Enforce safety constraint on QI_{gn}
9: Remove QI from $Buckets$
10: **else**
11: Enforce l-diversity over QI
12: **if** size of $QI < k$ **then**
13: Suppress tuples in QI
14: **else**
15: Go to Step 6
16: **end if**
17: **end if**
18: **end while**
19: **for** $j = 1$ to gn **do**
20: **for** each tuple $t \in QI_j$ **do**
21: insert tuple $(t[A_1], ..., t[A], ...t[A_m], j)$ into T_{QIT}
22: **end for**
23: **for** each random value v_s of $A^s \in QI_j$ **do**
24: insert tuple (j, v_s) into T_{SNT}
25: **end for**
26: **end for**

8 Experiments

We now present a set of experiments to evaluate the efficiency of our approach, both in terms of computation and more importantly, loss of data utility. We implemented the safe grouping code in Java based on the Anonymization Toolbox [7], and conducted experiments with an Intel XEON 2.4GHz PC with 2GB RAM.

8.1 Evaluation Dataset

In keeping with much work on anonymization, we use the Adult Dataset from the UCI Machine Learning Repository [2]. To simulate real identifiers, we made use of a U.S. state voter registration list containing the attributes *Birthyear*, *Gender* , *Firstname*, and *Lastname*. We combined the adult dataset with the voter's list such that every individual in the voters list is associated with multiple tuples from the adult dataset, simulating a longitudinal dataset from multiple census

years. We have constructed this dataset to have a correlation dependency of the following form
$Firstname, Lastname \dashrightarrow Occupation$; where $Occupation$ is a sensitive attribute, $Firstname, Lastname$ are identifying attributes and remaining attributes are presumed to be quasi-identifiers.

We say that an individual is likely to stay in the same occupation across multiple censuses. Note that this is not an exact longitudinal dataset; n varies between individuals (simulating a dataset where some individuals move into or out of the census area. The generated dataset is of size 48836 tuples with 21201 distinct individuals.

In the next section, we present and discuss results from running our safe grouping algorithm.

8.2 Evaluation Results

We elaborated a set of measurements to evaluate the efficiency of safe grouping. These measurements can be summarized as follows:

- Evaluating privacy breach in a naive anatomization. We note that the same test could be performed on the slicing technique [11] as the authors in their approach do not deal with identity disclosure,
- Determining anonymization cost represented by the loss metric to capture the fraction of tuples that must be (partially or totally) generalized, suppressed, or encrypted in order to satisfy the safe grouping and,
- Comparing the computational cost of our safe grouping algorithm to anatomy [23].

8.2.1 Evaluating Privacy

After naïve anatomization over the generated dataset, we have identified 5 explicit violations due to intra QI-group correlations where values of A^{id} are correlated in a QI-group. On the other hand, in order to determine the number of violations due to inter QI-group correlation, we calculate first the possible associations of A^{id} and A^s values across a naïve anatomized table. This is summarized in the following equation for values v_{id} and v_s respectively.

$$\mathcal{G}(v_{id}, v_s) = \frac{\sum_{j=1}^{m} f_j(v_{id}, v_s)}{\sum_{j=1}^{m} g_j(v_{id})}$$

where

$$f_j(v_{id}, v_s) = \begin{cases} 1 \text{ if } v_{id} \text{ and } v_s \text{ are associated in } QI_j \\ 0 \; otherwise \end{cases}$$

and,

$$g_j(v_{id}) = \begin{cases} 1 \text{ if } v_{id} \text{ exists in } QI_j \\ 0 \text{ otherwise} \end{cases}$$

At this point, a violation occurs for significant[2] A^{id} values if;

1. $\mathcal{G}(v_{id}, v_s) > 1/l$. This represents a *frequent* association between v_{id} and v_s where v_{id} is more likely to be associated to v_s in the QI-groups to which it belongs and,
2. $|\pi_{A^s} QI_1 \cap ... \cap \pi_{A^s} QI_m| < l$ where $QI_1, ..., QI_m$ are the QI-groups to which v_{id} belongs.

After we applied the above test to the anatomized dataset, we have identified for $l = 2$ and $l = 3$, 167 and 360 inter QI-groups correlation violations. We note that a much deeper study on violations due to data correlation can be found in [22][8][10].

8.2.2 Evaluating Anonymization Cost

We evaluate our proposed anonymization algorithms to determine the loss metric ($\mathcal{LM}$) representing the number of tuples in T and T_{QIT} that need to be suppressed in order to achieve the safety constraint. Figure 3 shows a anonymized version of table prescription where grouping is safe and has a loss metric equal to $\mathcal{LM}(Prescription) = 2/13$.

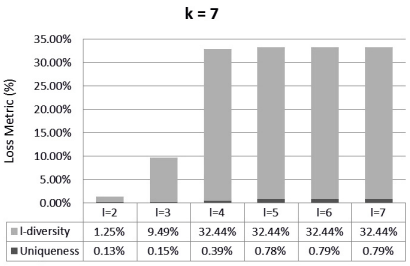
We investigate the anonymization cost for a correlation dependency cd^{id} : $Firstname, Lastname \dashrightarrow Occupation$ using the safe grouping algorithm. We anonymize the dataset with $k = 7, 8, 9$ and $l = 2, 3, 4, 5, 6, 7$ for which the dataset satisfies the eligibility condition (see [13]). At each execution, we compute the $\mathcal{LM}$. The results are shown in Figure 4.

From Figure 4, we can see that the $\mathcal{LM}$ increases with l, and for ($k = 9$, $l = 7$) the computed loss metric $\mathcal{LM}$ is high; notice that the number of tuples to anonymize in order to preserve l-diversity reaches 35%. Nonetheless, for small values of l an acceptable value of $\mathcal{LM}$ is computed. Anonymizing datasets using safe grouping can be seen as a trade-off between cost and privacy where for small values of l, $\mathcal{LM}$ produces values less than 10% leading to a relatively small anonymization cost. Another aspect to consider is how to define k w.r.t l to guarantee a minimum $\mathcal{LM}$. Note that for transactional data, it is possible for k (the number of individuals, not transactions, in a group) to be smaller than l; however, this makes satisfying the privacy criteria difficult, leading to substantial anonymized data. The experiments show that high data utility can be preserved as long as k is somewhat greater than l.

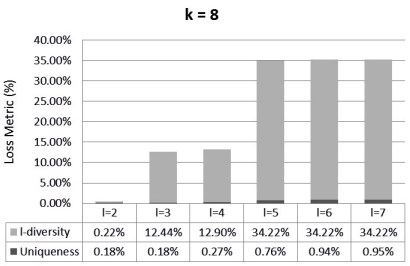
8.2.3 Evaluating Computation Cost

We now give the processing time to perform safe grouping compared to anatomy. Figure 4d shows the computation time of both safe grouping and anatomy over a non-transactional dataset with different k values. Theoretically, a worst case of safe grouping could be much higher; but in practice, for a small values of l the

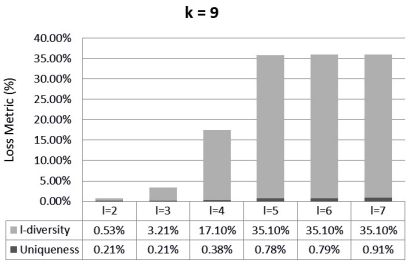
[2] Significance is measured in this case based on the support of A^{id} values and their correlation across QI-groups. For instance, v_{id} is considered significant if it exists in at least α QI-groups where α is a predefined constant greater than 2.



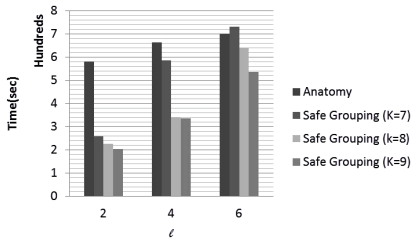
(a) % of tuples anonymized to ensure the safety constraint and l-diversity for $k = 7$



(b) % of tuples anonymized to ensure the safety constraint and l-diversity for $k = 8$



(c) % of tuples anonymized to ensure the safety constraint and l-diversity for $k = 9$



(d) Computational Cost Evaluation

Fig. 4. Safe grouping evaluation in transactional datasets 4a, 4b and 4c

safe grouping has better performance than anatomy. Furthermore, as k increases the safe grouping computation time decreases due to reduced I/O access needed to test QI-groups' l-diversity.

9 Conclusion

In this paper, we proposed a safe grouping method to cope with defects of bucketization techniques in handling correlated values in a transactional dataset. Our safe grouping algorithm creates partitions with an individual's related tuples stored in one and only one group, eliminating these privacy violations. We showed, using a set of experiments, that there is a trade-off to be made between privacy and utility. This trade-off is quantified based on the number of tuples to be anonymized using the safe grouping algorithm. Finally, we investigated the computation time of safe grouping and showed that despite the exponential growth of safe grouping, for a small range of values of l, safe grouping outperforms anatomy while providing stronger privacy guarantees.

Acknowledgements. This publication was made possible by NPRP grant 09-256-1-046 from the Qatar National Research Fund. The statements made herein are solely the responsibility of the authors.

References

1. Aldous, D.J.: Exchangeability and related topics. In: Saad, Y., Yang, T., Ferreira, A., Rolim, J.D.P. (eds.) IRREGULAR 1996. LNCS, vol. 1117, pp. 1–198. Springer, Heidelberg (1996)
2. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)
3. Burghardt, T., Böhm, K., Guttmann, A., Clifton, C.: Anonymous search histories featuring personalized advertisement - balancing privacy with economic interests. Transactions on Data Privacy 4(1), 31–50 (2011)
4. Ciriani, V., De Capitani Di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Combining fragmentation and encryption to protect privacy in data storage. ACM Trans. Inf. Syst. Secur. 13, 22:1–22:33 (2010)
5. Hacıgümüş, H., Iyer, B.R., Mehrotra, S.: Executing SQL over encrypted data in the database-service-provider model. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, Madison, WI, June 4-6, pp. 216–227 (2002)
6. Jiang, X., Gao, J., Wang, T., Yang, D.: Multiple sensitive association protection in the outsourced database. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010. LNCS, vol. 5982, pp. 123–137. Springer, Heidelberg (2010)
7. Kantarcioglu, M., Inan, A., Kuzu, M.: Anonymization toolbox (2010)
8. Kifer, D.: Attacks on privacy and definetti's theorem. In: SIGMOD Conference, pp. 127–138 (2009)
9. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: ICDE, pp. 106–115 (2007)
10. Li, T., Li, N.: Injector: Mining background knowledge for data anonymization. In: ICDE, pp. 446–455 (2008)
11. Li, T., Li, N., Zhang, J., Molloy, I.: Slicing: A new approach for privacy preserving data publishing. IEEE Trans. Knowl. Data Eng. 24(3), 561–574 (2012)
12. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: *l*-diversity: Privacy beyond *k*-anonymity. In: Proc. 22nd IEEE Intl. Conference on Data Engineering (ICDE 2006), Atlanta, GA (April 2006)
13. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: *l*-diversity: Privacy beyond *k*-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD) 1(1) (March 2007)
14. Malin, B.: Trail Re-identification and Unlinkability in Distributed Databases. PhD thesis, Carnegie Mellon University (2006)
15. Nergiz, A.E., Clifton, C.: Query processing in private data outsourcing using anonymization. In: Li, Y. (ed.) DBSec. LNCS, vol. 6818, pp. 138–153. Springer, Heidelberg (2011)
16. Nergiz, A.E., Clifton, C., Malluhi, Q.: Updating outsourced anatomized private databases. In: 16th International Conference on Extending Database Technology (EDBT), Genoa, Italy, March 18-22 (2013)
17. Ressel, P.: De Finetti-type theorems: an analytical approach. Ann. Probab. 13(3), 898–922 (1985)
18. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Trans. Knowl. Data Eng. 13(6), 1010–1027 (2001)
19. Sweeney, L.: k-anonymity: a model for protecting privacy. Intl. Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10(5), 557–570 (2002)

20. Terrovitis, M., Mamoulis, N., Kalnis, P.: Privacy-preserving anonymization of set-valued data. Proc. VLDB Endow. 1(1), 115–125 (2008)
21. Wang, K., Fung, B.C.M.: Anonymizing sequential releases. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2006, pp. 414–423. ACM, New York (2006)
22. Wong, R.C.-W., Fu, A.W.-C., Wang, K., Xu, Y., Yu, P.S.: Can the utility of anonymized data be used for privacy breaches? CoRR, abs/0905.1755 (2009)
23. Xiao, X., Tao, Y.: Anatomy: Simple and effective privacy preservation. In: Proceedings of 32nd International Conference on Very Large Data Bases (VLDB 2006), Seoul, Korea, September 12-15 (2006)

Practical Immutable Signature Bouquets (PISB) for Authentication and Integrity in Outsourced Databases

Attila A. Yavuz

Robert Bosch LLC, Research and Technology Center - North America
2835 East Carson Street, Pittsburgh, PA 15203
attila.yavuz@us.bosch.com

Abstract. Database outsourcing is a prominent trend that enables organizations to offload their data management overhead (e.g., query handling) to the external service providers. Immutable signatures are ideal tools to provide authentication and integrity for such applications with an important property called immutability. Signature immutability ensures that, no attacker can derive a valid signature for unposed queries from previous queries and their corresponding signatures. This prevents an attacker from creating his own de-facto services via such derived signatures. Unfortunately, existing immutable signatures are very computation and communication costly (e.g., highly interactive), which make them impractical for task-intensive and heterogeneous applications.

In this paper, we developed two new schemes that we call *Practical and Immutable Signature Bouquets (PISB)*, which achieve efficient immutability for outsourced database systems. Both *PISB* schemes are very simple, non-interactive, and computation/communication efficient. Our generic scheme can be constructed from any aggregate signature coupled with a standard signature. Hence, it can flexibly provide performance trade-offs for various types of applications. Our specific scheme is constructed from Condensed-RSA and Sequential Aggregate RSA. It has a very low verifier computational overhead and end-to-end delay with a small signature size. We showed that *PISB* schemes are secure and also much more efficient than previous alternatives.

Keywords: Applied cryptography, outsourced databases, immutable digital signatures, distributed systems, public key cryptography.

1 Introduction

It is a growing trend that the data is outsourced and being managed (e.g., query handling, maintenance) on remote servers, which are maintained by third party outsourcing vendors. One such data outsourcing approach is "database as a service" (DAS) model [1], in which clients outsource their data to a database service provider[1,2] that offers a reliable maintenance and access for the hosted data [2].

[1] http://www.ibm.com/software/data/db2
[2] http://www-935.ibm.com/services/us/en/it-services/storage-and-data-services.html

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 179–194, 2013.

Data outsourcing can significantly reduce the cost of data management (e.g., via continuous service, expertise, upgrade/maintanence) and therefore it is highly beneficial for entities with limited management capabilities such as small to medium businesses [2–4]. However, despite its merits, data outsourcing brings various security challenges, since the sensitive data is hosted in a (semi or fully) untrusted environment. These security challenges include but not limited to the confidentiality [5], access privacy [6], authentication and integrity [7]. Another challenge is to provide the security efficiently such that the data outsourcing still remains practical and cost efficient.

In this paper, we focus on providing authentication and integrity of outsourced data via aggregate signatures (e.g., [8]), while also guaranteeing a vital security property called *signature immutability* in a *practical* manner.

In Section 1.1, we give our data and system models. In Section 1.2, we describe the signature mutability problem and limitations of existing solutions. In Section 1.3, we summarize our contributions and highlight the desirable properties of our schemes.

1.1 System and Data Model

We follow Mykletun et al.'s Outsourced Database Model (ODB) [3,7], which is a variant of traditional "database as a service" model [1].

System Model: There are three types of entities in the system; data owners, server (database service provider) and data queriers (clients). These entities behave as follows.

• *Data Owners:* A data owner can be a single or a logical entity such as an organization. Each data owner in the system signs her database elements (e.g., each tuple separately) and then outsources them along with their signatures to *the server*. This protects the integrity and authentication of outsourced data against *both* the server and outside adversaries (e.g., in the case of the server is compromised).

The data owner computes the individual signature of each database element (e.g., each tuple) with an aggregate signature scheme (e.g., [8]), which allows the combination of these signatures according to the content of a query. This enables the server to reply any query on the outsourced data with a compact constant size signature (instead of sending a signature for each element in the query, which entails a linear communication overhead). This outsourcing step is performed *offline*, and therefore its cost is not the main concern.

• *Server (Service Provider):* The server maintains the data and handles the queries of *data queriers*. The server is trusted with these services, but it is *not* trusted with the integrity and authentication of the hosted data. Hence, each data owner digitally signs her data before outsourcing it as described previously.

Once a *data querier* (i.e., clients who perform data queries) queries the server, the server computes a constant size signature by aggregating the corresponding individual signatures of database elements associated with this query. Recall that the server knows these individual signatures, since the data owner provided all individual signatures to the server at the offline phase. The server then performs necessary cryptographic operations to ensure the immutability of this aggregate signature. Observe that the server faithfully follows the immutability operations, since the immutability prevents external parties to offer similar services free of charge.

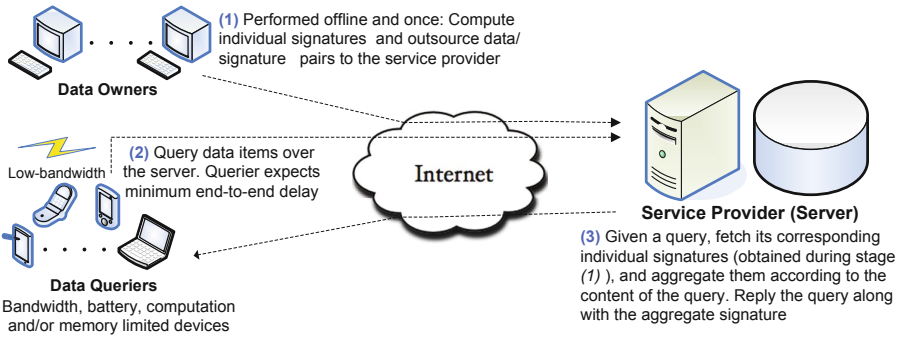


Fig. 1. Mykletun et al.'s Outsourced Database Model (ODB)

The query handling phase is performed online. The server is expected to handle larger number of queries simultaneously with a minimum end-to-end delay. Therefore, the cost of signature immutability operations are *highly critical*.

• *Data Queriers (Clients)*: Queriers are heterogeneous entities, which may be resource-constrained in terms bandwidth, battery and/or computation (e.g., a mobile device or a PDA). A querier can make a query on the database elements belonging to a single or multiple data owners. The former is called *non-cross signer queries* while the latter is called *cross signer queries*. The data querier verifies the corresponding aggregate signature of her query, along with cryptographic tokens transmitted for the immutability.

Figure 1 summarizes the ODB model described above.

Data Model and Scope: We assume that the data is managed with a traditional relational database management system and the queries are formulated with SQL. Our work handles SQL queries involving SELECT clauses, which return the selection of a set of records or fields matching a given predicate. Our work does *not* address SQL queries involving data aggregation that return a single value for a given query.

The granularity of data integrity and authentication may vary according to the application (e.g., attribute level). One possible choice is to provide them at the tuple level (i.e., sign each tuple individually), which offers a balance between the storage, transmission and computation overheads introduced by the cryptographic scheme [3].

1.2 Problem Statement and Limitations of Existing Solutions

Ability to aggregate different signatures into a single signature is advantageous, but it also allows any party to derive new aggregate signatures from the existing ones. For instance, assume that the server provides signatures $(\sigma_{1,l}, \sigma_{1,k})$ on queries $(m_1, \ldots, m_l)$ and $(m_1, \ldots, m_k)$, respectively, where $1 < k < l$ and the aggregation operation is addition (e.g., [8]). Any querier can derive a valid signature on query elements $m_{k+1}, \ldots, m_l$ (that have *not* been queried before) by simply computing $\sigma_{k+1,l} = \sigma_{1,l} - \sigma_{1,k}$.

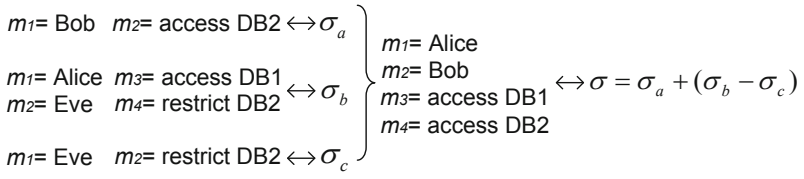


Fig. 2. An example of signature mutability in the content access control applications

This property has undesirable effects on many real-life applications. One example is *content access control mechanisms* for outsourced databases. Assume that the data owner requires the server to enforce an access control policy, in which each client can access only certain parts of the database via an access token (i.e., a signature). A group of clients can possess different access privileges from those owned by each client in isolation. Figure 2 exemplifies how three colluding clients can derive a valid token (i.e., signature) with an access right beyond their actual privileges.

Another example is *paid database services*, in which the server acts as an authorized re-distribution agent for the information contained in the outsourced database. Assume that the server charges a fee for each music album queried (downloaded) over the outsourced database. Signature mutability allows an unauthorized splitting and re-distribution of authentic query replies. Colluding clients may gather various music albums and their signatures, and combine and re-sell them according to their will (without paying/obtaining any authorization) [7].

Mykletun et al. [7] introduce signature immutability techniques to address these problems. Their RSA-based techniques prevent an adversary from deriving new signatures by hiding the actual aggregate signature via an interactive Guillou-Quisquater (GQ) [9] based protocol. This approach is interactive and therefore introduces high communication overhead and end-to-end delay. Their non-interactive RSA variant uses a signatures of knowledge method, which substantially increases the computational cost and has large signature size. Their BLS signature method iBGLS [8] offers a small signature size, but it is very computationally costly due to cryptographic pairing operations. Hence, none of these techniques are suitable for nowadays task-intensive and heterogeneous outsourcing applications.

1.3 Our Contribution

To address these limitations, we develop novel cryptographic schemes called *Practical Immutable Signature Bouquets (PISB)*, which is suitable for outsourced database systems. Specifically, we developed a Condensed-RSA (C-RSA) and Sequential Aggregate RSA (SA-RSA) based scheme called $PISB\text{-}CSA\text{-}RSA$ and a generic scheme called $PISB\text{-}Generic$. We summarize the desirable properties of our schemes below:

- *Non-interactive Signature Immutability*: $PISB$ schemes do not require any multi-round interaction among the server and queriers. Hence, they are much more communication efficient than previous alternatives. For instance, our $PISB\text{-}CSA\text{-}RSA$

Table 1. The client/server overhead of $PISB$ and its counterparts for 10 items in a query (in ms)

-	$PISB$-$Generic$	$PISB$-CSA-RSA	GQ-based [7]	SKROOT [7]	iBGLS [7]
Server Comp.	0.66 / 0.39	4.03	1.5	92.4	2.2
Client Comp.	224.97	0.46	1.57	92.77	245.7
Extra rounds	0	0	3 rounds (each 3 passes)	0	0
(Est.) End-to-end	225.63	4.49	292	185.17	247.9
Signature size	60 byte	1 KB	9 KB	4 KB	20 byte
sk size	40 byte	2 KB	1 KB	5 KB	20 byte
pk size	80 byte	1 KB	1 KB	1 KB	40 byte
Aggregation Type	Cross/non-cross	Non-cross	Non-cross	Non-cross	Cross
Provable Sec.	Yes	Yes	No	No	No
Pre-computability	Yes	No	No	No	No

• Analytical comparison, key/parameter sizes and measurement details are given in Section 6.
• The immutable signature size is the aggregate signature size plus the size of additional cryptographic tags transmitted (e.g., protection signatures, values transmitted for multi-rounds).
• $PISB$-$Generic$ is instantiated with BLS [8] as $ASig$ (20 byte aggregate signature) and with ECDSA [10] as Sig (40 byte protection signature) for pre-computed parameters (0.36 ms) [11] or pre-computed tokens [12] (0.03 ms).
• End-to-end delay is the sum of client and server execution times plus the estimated communication delay introduced by multi-rounds. Only GQ-based scheme requires multi-rounds, which substantially increase its end-to-end delay.
• *Remark*: $PISB$-CSA-RSA is significantly more efficient than all of its counterparts at the client side, which makes it suitable for mobile or resource-constrained queriers. Its end-to-end delay is also *40 to 60 times lower than its alternatives*. $PISB$-$Generic$ offers various performance trade-offs with its generic structure (e.g., only alternative with pre-computability). This instantiation of $PISB$-$Generic$ offers small signature/key sizes and high server efficiency simultaneously. $PISB$-$Generic$ and $PISB$-CSA-RSA are suitable for cross and non-cross signer models, respectively (see Section 1.2). $PISB$ schemes are also provable secure.

incurs only 1KB communication overhead, while GQ-based scheme in [3,7] requires 9KB. Moreover, the non-interactive nature of our schemes make them packet loss tolerant, which is a desirable property for mobile and ad-hoc clients (queriers).

- *High Computational Efficiency*: $PISB$ schemes are much more computationally efficient than their counterparts. $PISB$-CSA-RSA is the most client efficient scheme among its counterparts, being a magnitude of time faster than SKROOT-based and iBGLS schemes in [3,7]. Therefore, $PISB$-CSA-RSA is an ideal alternative for battery and/or computational limited clients (queriers) such as mobile and hand-held devices. It is also plausibly efficient at the server side while achieving this client efficiency. $PISB$-$Generic$ is the most server efficient scheme among its counterparts due to its pre-computability property. This enables the server to responde large number queries in peak times without being bottlenecked.
- *Small Signature Sizes*: $PISB$-CSA-RSA is the only RSA-based scheme that can compute a compact immutable aggregate signature, which makes it more communication efficient than its counterparts [3,7]. $PISB$-$Generic$ has a much smaller signature size than RSA-based schemes and also has a comparable signature size with iBGLS in [3,7] (while being much more computationally efficient).
- *Minimum End-to-end Delay*: $PISB$ schemes have a much smaller end-to-end delay than all of their counterparts, which offers a better service quality.

- *Provable Security*: Previous works (e.g., [3,7]) give only heuristic security arguments regarding the signature immutability. Our work is the only one providing a formal security model and proofs for the signature immutability.

Table 1 outlines the properties and compares $PISB$ schemes with their counterparts.

The remainder of this paper is organized as follows. Section 2 provides definitions used by our schemes. Section 3 defines our security model. Section 4 describes the proposed schemes in detail. Section 5 provides the security analysis. Section 6 gives the performance analysis and compares our schemes with their counterparts. Section 7 outlines related work and Section 8 concludes this paper.

2 Preliminaries

In this section, we give the preliminary definitions used by our schemes.

Definition 1. *A signature scheme Sig is a tuple of three algorithms $(Kg, Sign, Ver)$ defined as follows:*

- $(sk, pk) \leftarrow Sig.Kg(1^\kappa)$*: Given the security parameter 1^κ, the key generation algorithm returns a private/public key pair (sk, pk) as the output.*
- $s \leftarrow Sig.Sign(sk, m)$*: The signing algorithm takes sk and a message m as the input. It returns a signature s as the output.*
- $c \leftarrow Sig.Ver(pk, m, s)$*: The signature verification algorithm takes pk, m and s as the input. It outputs a bit c, with $c = 1$ meaning* valid *and $c = 0$ meaning* invalid.

The standard security notion for a signature scheme is *Existential Unforgeability under Chosen Message Attacks (EU-CMA)* [13], which is defined below.

Definition 2. *EU-CMA experiment for Sig is defined as follows:*

- Setup. *Challenger algorithm $\mathcal{B}$ runs the key generation algorithm as $(sk, pk) \leftarrow Sig.Kg(1^\kappa)$ and provides pk to the adversary $\mathcal{A}$.*
- Queries. *Beginning from $j = 1$ and proceeding adaptively, $\mathcal{A}$ queries $\mathcal{B}$ on any message m_j of her choice up to q_s messages in total. For each query j, $\mathcal{B}$ computes $s_j \leftarrow Sig.Sign(sk, m_j)$ as the signing oracle of $\mathcal{A}$ and returns s_j to $\mathcal{A}$.*
- Forgery. *Finally, $\mathcal{A}$ outputs a forgery (m^*, s^*) and wins the EU-CMA experiment, if $Sig.Ver(pk, m^*, s^*) = 1$ and m was not queried to $\mathcal{B}$.*

Sig is (t, q_s, ϵ)-EU-CMA secure, if no $\mathcal{A}$ in time t making at most q_s signature queries has an advantage at least with probability ϵ in the above experiment.

An aggregate signature scheme (e.g., [8]) aggregates multiple signatures of different signers into a single compact signature. Hence, it can be used for *cross querier* applications.

Definition 3. *An aggregate signature scheme $ASig$ is a tuple of four algorithms $(Kg, Sign, Agg, Ver)$ defined as follows:*

- $(\overrightarrow{sk}, \overrightarrow{pk}) \leftarrow ASig.Kg(1^\kappa)$*: Given the security parameter* 1^κ *and a set of signers* $\mathbb{U} = \{1, \ldots, u\}$*, the aggregate key generation algorithm generates a private/public key pair* (sk_i, pk_i) *for* $i = 1, \ldots, u$*, as in Definition 1 key generation algorithm. The aggregate key generation algorithm returns a private/public key pair* $\overrightarrow{sk} = (sk_1, \ldots, sk_u)$ *and* $\overrightarrow{pk} = (pk_1, \ldots, pk_u)$ *as the output.*
- $s_i \leftarrow ASig.Sign(sk_i, m_i)$*: As in Definition 1 signature generation algorithm.*
- $\sigma_{1,u} \leftarrow ASig.Agg(\{pk_i, m_i, s_i\}_{i=1}^u)$*: The aggregation algorithm takes* $\{pk_i, m_i, s_i\}_{i=1}^u$ *as the input. It combines individual signatures* $s_i, 1 \leq i \leq u$ *and returns an aggregate signature* σ *as the output.*
- $c \leftarrow ASig.Ver(\{pk_i, m_i\}_{i=1}^u, \sigma_{1,u})$*: The aggregate verification algorithm takes* $\{pk_i, m_i\}_{i=1}^u$ *and* $\sigma_{1,u}$ *as the input. It outputs a bit* c*, with* $c = 1$ *meaning* valid *and* $c = 0$ *meaning* invalid*.*

The $EU\text{-}CMA$ experiment for $ASig$ is a straightforward extension of Definition 2, in which $\mathcal{A}$ is required to produce a forgery under a public key $pk \in \overrightarrow{pk}$ that is not under his control during the experiment (see [8] for details).

Condensed-RSA (i.e., $C\text{-}RSA$) [3,7] aggregates RSA signatures computed under *the same private key*. Hence, it is used for *non-cross querier (signer)* applications.

Definition 4. $C\text{-}RSA$ *is a tuple of three algorithms* (Kg, Sig, Ver) *defined as follows:*

- $(sk, pk) \leftarrow C\text{-}RSA.Kg(1^\kappa)$*: Given the security parameter* 1^κ*, the key generation algorithm generates a* RSA *private/public key pair. That is, it randomly generates two large primes* (p, q) *and computes* $n = p \cdot q$*. The public and secret exponents* $(e, d) \in Z^*_{\phi(n)}$ *satisfies* $e \cdot d \equiv 1 \bmod \phi(n)$*, where* $\phi(n) = (p-1)(q-1)$*. The key generation algorithm returns* $sk \leftarrow (n, d)$ *and* $pk \leftarrow (n, e)$ *as the output.*
- $\sigma \leftarrow C\text{-}RSA.Sig(sk, \overrightarrow{m})$*: Given* sk *and messages* $\overrightarrow{m} = (m_1, \ldots, m_l)$*, the signing algorithm returns a signature* $\sigma \leftarrow \prod_{j=1}^l s_j \bmod n$ *as the output, where* $s_j \leftarrow [H(m_j)]^d \bmod n$ *for* $j = 1, \ldots, l$*.* H *is a full domain hash function (e.g., [14]) defined as* $H : \{0,1\}^* \rightarrow Z_n$*.*
- $c \leftarrow C\text{-}RSA.Ver(pk, \overrightarrow{m}, \sigma)$*: Given* $pk = (n, e)$*,* $\overrightarrow{m}$ *and* σ*, if* $\sigma^e = \prod_{j=1}^l H(m_j) \bmod n$ *then the signature verification algorithm returns bit* $c = 1$ *else* $c = 0$*.*

A sequential aggregate signature (e.g., [15]) requires that the signature generation and verification are performed in a specific order. The signature generation and aggregation operations are unified.

In $PISB\text{-}CSA\text{-}RSA$, we use a (simplified) *single signer (and aggregator) instantiation* of $SA\text{-}RSA$ [15] (also see Remark 1 in Section 4). However, for the sake of completeness, we give the full description of $SA\text{-}RSA$ for multiple-signers below.

Definition 5. *Sequential Aggregate RSA (denoted as* $SA\text{-}RSA$*) is a tuple of three algorithms* $(Kg, ASign, Ver)$ *defined as follows:*

- $(\overrightarrow{sk}, \overrightarrow{pk}) \leftarrow SA\text{-}RSA.Kg(1^\kappa)$*: Given the security parameter* 1^κ *and a set of signers* $\mathbb{U} = \{1, \ldots, u\}$*, the key generation algorithm generates a* RSA *private/public key pair* $sk_i \leftarrow (n_i, d_i)$ *and* $pk_i \leftarrow (n_i, e_i)$*, ensuring that* $2^{k-1}(1 + (i-1)/u) \leq n_i < 2^{k-1}(1 + i/u)$*, where* $k = |n_i|$ *for* $i = 1, \ldots, u$*. It returns a private/public key pair* $\overrightarrow{sk} \leftarrow \{n_i, d_i\}_{i=1}^u$ *and* $\overrightarrow{pk} \leftarrow \{n_i, e_i\}_{i=1}^u$ *as the output.*

- $\sigma_{1,u} \leftarrow$ *SA-RSA.Sig*$(sk_u, \{m_i\}_{i=1}^{u-1}, m_u, \{pk_i\}_{i=1}^{u-1}, pk_u, \sigma_{1,u-1})$*: The signer* u *receives aggregate signature* $\sigma_{1,u-1}$ *on messages* $\{m_i\}_{i=1}^{u-1}$ *under public keys* $\{pk_i\}_{i=1}^{u-1}$. *The signer* u *first verifies* $\sigma_{1,u-1}$ *with the verification algorithm SA-RSA.Ver. If it succeeds, the signer* u *computes the signature on* $\overrightarrow{m} = (m_1, \ldots, m_u)$ *under* $\overrightarrow{pk}$ *as* $h_u = H(\overrightarrow{m}||\overrightarrow{pk})$ *and* $y_u = h_u + \sigma_{1,u-1}$. *The sequential aggregate signature outputs the signature* $\sigma_{1,u} \leftarrow y_u^{d_u} \bmod n_u$.
- $c \leftarrow$ *SA-RSA.Ver*$(\overrightarrow{m}, \overrightarrow{pk}, \sigma_{1,u})$*: Given* $\sigma_{1,u}$ *on* $\overrightarrow{m}$ *under public keys* $\overrightarrow{pk} = \{n_i, e_i\}_{i=1}^{u}$, *first check* $0 \leq \sigma_{1,u} \leq n_u$. *If* $gcd(\sigma_{1,u}, n_u) = 1$ *then* $y_u \leftarrow \sigma_{1,u}^{e_u} \bmod n_u$ *else* $y_u \leftarrow \sigma_{1,u}$. *Compute* $h_u \leftarrow H(\overrightarrow{m}||\overrightarrow{pk})$ *and* $\sigma_{1,u-1} \leftarrow (y_u - h_u) \bmod n_u$. *Verify signatures recursively as described until the base case* $u = 1$, *in which check* $(\sigma_{1,1} - h_1) \bmod n_1 = 0$ *where* $h_1 \leftarrow (m_1||pk_1)$. *If it holds return* $c = 1$ *else* $c = 0$.

3 Security Model

Our security model reflects how *PISB* system model works. That is, our security model formally captures the immutability of aggregate signatures for the *EU-CMA* experiment, which we call Immutable-*EU-CMA* (*I-EU-CMA*) experiment.

Definition 6. *I-EU-CMA experiment for PISB is defined as follows:*

- Setup. *Challenger algorithm* $\mathcal{B}$ *runs* $(SK, PK) \leftarrow PISB.Kg(1^\kappa)$ *and provides PK to the adversary* $\mathcal{A}$.
- Queries. $\mathcal{A}$ *queries* $\mathcal{B}$ *on any message* $\overrightarrow{m}_j = (m_{j,1}, \ldots, m_{j,l})$ *of her choice for* $j = 1, \ldots, q_s$. $\mathcal{B}$ *replies each query* j *with a signature* γ_j *computed under* PK.
- Forgery. $\mathcal{A}$ *outputs a forgery* (m^*, γ^*) *and wins the EU-CMA experiment, if*
  (i) $PISB.Ver(PK, m^*, \gamma^*) = 1$,
  (ii) $m^* \not\subseteq \{\overrightarrow{m}_j\}_{j=1}^{q_s}$ *or* $\exists I \subseteq \{1, \ldots, q_s\} : m^* \subseteq ||_{k \in I} \overrightarrow{m}_k$
  That is, the forgery is valid and m^* *has not been queried or it is a subset and/or any combination of previously queried data items* $(\overrightarrow{m}_1, \ldots, \overrightarrow{m}_{q_s})$.

PISB is (t, q_s, ϵ)*-I-EU-CMA secure, if no* $\mathcal{A}$ *in time* t *making at most* q_s *signature queries has an advantage at least with probability* ϵ *in the above experiment.*

4 The Proposed Schemes

In this section, we describe our proposed schemes. For each *PISB* scheme, we first give the intuition behind the scheme followed by its detailed description.

PISB-CSA-RSA Scheme: An effective way to provide the signature immutability is to compute a protection signature over all data items associated with the query. That is, the server computes a signature on all data items in the query with his own private key. He then aggregates the protection signature over the aggregate signature computed from data owner's signatures. Breaking the immutability of this final aggregate signature is as difficult as forging the server's protection signature [3, 7].

Despite its simplicity, this method is *not* applicable to aggregate signatures such as $C\text{-}RSA$, in which only the signatures computed under the same public key can be aggregated (also called as *non-cross signer* aggregate signature). Recall that $C\text{-}RSA$ cannot aggregate signatures belonging to different signers, since an RSA modulus n can not be safely shared among multiple signers (this leads to the factorization of n, exposing the private keys [16]). Hence, despite $C\text{-}RSA$ is an efficient scheme, its immutable variants (e.g., [3, 7]) are inefficient as discussed in Section 1.2.

It is highly desirable to construct a scheme that can compute an aggregate RSA signature involving both a data owner and the server (without exposing their private keys via the factorization of modulo). Our *main observation* is that, this goal can be achieved by levering the sequential aggregate signatures from trapdoor permutations (e.g., $SA\text{-}RSA$ [15]) together with $C\text{-}RSA$. They allow distinct signers to sequentially compute an aggregate signature under distinct public keys (Definition 5).

In $PISB\text{-}CSA\text{-}RSA$, the data owner computes RSA signatures $s_1, \ldots, s_l$ on $m_1, \ldots, m_l$ with her keys (n, d). During the query phase, the server computes a $C\text{-}RSA$ signature σ' by aggregating RSA signatures. The server then uses $SA\text{-}RSA$ to compute an immutable aggregate signature γ on $m_1, \ldots, m_l$ with his keys $(\overline{n}, \overline{d})$ by aggregating it on σ'. The public key of the system is $(\langle n, e\rangle, \langle \overline{n}, \overline{e}\rangle)$. The verification order of the client is with $SA\text{-}RSA$ under $(\overline{n}, \overline{e})$ for γ and then with $C\text{-}RSA$ under (n, e) for σ'.

One may observe that breaking the immutability of $PISB\text{-}CSA\text{-}RSA$ is as difficult as breaking RSA. We give the formal security analysis of $PISB\text{-}CSA\text{-}RSA$ in Section 5 (Theorem 2).

The $PISB\text{-}CSA\text{-}RSA$ algorithms are defined below.

1) $(SK, PK) \leftarrow PISB\text{-}CSA\text{-}RSA.Kg(1^\kappa)$: The data owner executes $(sk, pk) \leftarrow C\text{-}RSA.Kg(1^k)$, where $sk = (n, d)$ and $pk = (n, e)$. The server generates a RSA private/public key pair $\overline{sk} \leftarrow (\overline{n}, \overline{d})$ and $\overline{pk} \leftarrow (\overline{n}, \overline{e})$ such that $n < \overline{n}$. The system private/public key are $SK \leftarrow (sk, \overline{sk})$ and $PK \leftarrow (pk, \overline{pk})$.

2) $\overrightarrow{V} \leftarrow PISB\text{-}CSA\text{-}RSA.Init(\overrightarrow{m}, sk)$: The data owner computes an individual signature $s_j \leftarrow [H(m_j)]^d \bmod n$ for $j = 1, \ldots, l$, where $\overrightarrow{m} = (m_1, \ldots, m_l)$. The data owner sets the message-signature pairs as $\overrightarrow{V} \leftarrow (\overrightarrow{m}, \overrightarrow{S})$ and provide $\overrightarrow{V}$ to the server, where $\overrightarrow{S} = (s_1, \ldots, s_l)$.

3) $\gamma \leftarrow PISB\text{-}CSA\text{-}RSA.Sign(\overline{sk}, \overrightarrow{m}, \overrightarrow{V})$: The server receives a non-cross-signer query $\overrightarrow{m} = (m_1, \ldots, m_l)$. It fetches the corresponding signatures $(s_1, \ldots, s_l)$ on $\overrightarrow{m}$ from $\overrightarrow{V}$ and computes $\sigma' \leftarrow \prod_{j=1}^{l} s_j \bmod n$. It then computes $h \leftarrow H(\overrightarrow{m}||\overline{pk})$, $y \leftarrow (h + \sigma') \bmod \overline{n}$ and $\gamma \leftarrow y^{\overline{d}} \bmod \overline{n}$.

4) $c \leftarrow PISB\text{-}CSA\text{-}RSA.Ver(PK, \overrightarrow{m}, \gamma)$: Given γ, the verifier computes $y' \leftarrow \gamma^{\overline{e}} \bmod \overline{n}$ and $\sigma' \leftarrow (y' - h') \bmod \overline{n}$, where $h' \leftarrow H(\overrightarrow{m}||\overline{pk})$. If $C\text{-}RSA.Ver(pk, \overrightarrow{m}, \sigma') = 1$ then return $c = 1$ else $c = 0$.

Remark 1. In $PISB\text{-}CSA\text{-}RSA$, we use a *simplified* $SA\text{-}RSA$ variant [15] with the following properties: *(i)* $SA\text{-}RSA$ is used in a *single signer* setting (the server as the signer and aggregator). *(ii)* The public key correctness controls (e.g., range check and gcd control) are not required, since the public keys are *already certified* in our system

model. That is, n_i belongs to a legitimate signer and $gcd(e_i, \phi(n_i)) = 1$ holds. This retains the computational efficiency of traditional small RSA exponents.

PISB-Generic Scheme: Our generic scheme relies on *a very simple observation*: It is possible to guarantee the immutability of an aggregate signature by simply computing a standard digital signature on it. That is, the server can simply sign the aggregate signature with his private key and define the immutable signature as a signature pair.

$PISB\text{-}Generic$ slightly increases the signature size, since a secondary signature is transmitted along with the aggregate signature. However, this is actually *more communication efficient* than GQ-based and SKROOT-based methods in [3, 7]. That is, a secondary standard signature (e.g., ECDSA [10] with 40 bytes) is much smaller than cryptographic values transmitted (e.g., up to 9 KB) to achieve the immutability in [3,7].

$PISB\text{-}Generic$ also allows the server to choose *any* signature scheme to provide the immutability. For instance, the server may use ECDSA tokens [12] or offline/online signatures [17], which enable very fast response times in demand peaks via pre-computability. This flexibility makes $PISB\text{-}Generic$ more efficient at the server side than existing alternatives (see Table 1). However, note that, iBGLS has slightly smaller signature size (i.e., 20 byte) than that of $PISB\text{-}Generic$ (with the expense of a much higher server computational overhead).

The $PISB\text{-}Generic$ algorithms are defined below.

1) $(SK, PK) \leftarrow PISB\text{-}Generic.Kg(1^\kappa)$: Execute $(\overrightarrow{sk}, \overrightarrow{pk}) \leftarrow ASig.Kg(1^\kappa)$ for data owners $\mathbb{U} = \{1, \ldots, u\}$. Execute $(\overline{sk}, \overline{pk}) \leftarrow Sig.Kg(1^\kappa)$ for the server. The system private and public keys are $SK = (\overrightarrow{sk}, \overline{sk})$ and $PK = (\overrightarrow{pk}, \overline{pk})$, respectively.

2) $\overrightarrow{V} \leftarrow PISB\text{-}Generic.Init(\overrightarrow{M}, \overrightarrow{sk}, PK)$: Let $\overrightarrow{M} = \{\overrightarrow{m}_1, \ldots, \overrightarrow{m}_u\}$ be database elements to be outsourced, where each $\overrightarrow{m}_i = (m_{i,1}, \ldots, m_{i,l})$ belongs to the data owner $1 \leq i \leq u$. Each data owner i computes $s_{i,j} \leftarrow ASig.Sign(sk_i, m_{i,j})$ for $i = 1, \ldots, u$ and $j = 1, \ldots, l$. Set $\overrightarrow{V} \leftarrow (\overrightarrow{M}, \overrightarrow{S}, PK)$ and provide $\overrightarrow{V}$ to the server, where $\overrightarrow{S} = \{s_{i,j}\}_{i=1,j=1}^{u,l}$.

3) $\gamma \leftarrow PISB\text{-}Generic.Sign(\overline{sk}, \overrightarrow{m}, \overrightarrow{V})$: The server receives a cross-signer query $\overrightarrow{m} = \{m_1, \ldots, m_k\}$ on a subset of k data owners $U \in \mathbb{U}$. Fetch the corresponding public key and signatures on $\overrightarrow{m}$ from $\overrightarrow{V}$ as $V \leftarrow \{pk_i, m_{i,j}, s_{i,j}\}_{i \in U, \exists j: m_{i,j} \in \overrightarrow{M}}$ and compute $\sigma \leftarrow ASig.Agg(V)$. Also compute $s' \leftarrow Sig.Sign(\overline{sk}, \sigma)$ and set $\gamma \leftarrow (\sigma, s')$.

4) $c \leftarrow PISB\text{-}Generic.Ver(PK, \overrightarrow{m}, \gamma)$: Given $\gamma = (\sigma, s')$ and $pk \leftarrow \{pk_i\}_{i \in U}$, if $Sig.Ver(\overline{pk}, s', \gamma) = 1$ and $ASig.Ver(\overrightarrow{pk}, \overrightarrow{m}, \sigma) = 1$ hold return $c = 1$, else $c = 0$.

5 Security Analysis

We prove that $PISB$ schemes are $I\text{-}EU\text{-}CMA$ secure in Theorem 1 and Theorem 2. We ignore terms that are negligible in terms of κ.

Theorem 1. *PISB-Generic* is (t, q_s, ϵ)-*I-EU-CMA* secure, if *ASig* is (t', q_s, ϵ)-*EU-CMA* secure and *Sig* is (t', q_s, ϵ)-*EU-CMA* secure, where $t' = O(t) + q_s \cdot (Op + Op')$ and (Op, Op') denote the cost of signature generation for *ASig* and *Sig*, respectively.

Proof: Suppose algorithm $\mathcal{A}$ breaks (t, q_s, ϵ)-*I-EU-CMA* secure *PISB-Generic*. We then construct a simulator $\mathcal{B}$, which breaks (t', q_s, ϵ)-*EU-CMA* secure *ASig* or (t', q_s, ϵ)-*EU-CMA* secure *Sig* by using $\mathcal{A}$ as subroutine.

We set the *EU-CMA* experiments for *ASig* and *Sig*. $\mathcal{B}$ is given a *ASig* public key $\overrightarrow{pk}$ and a *Sig* public key $\overline{pk}$ as the input, where $(\overrightarrow{sk}, \overrightarrow{pk}) \leftarrow ASig.Kg(1^\kappa)$ and $(\overline{sk}, \overline{pk}) \leftarrow Sig.Kg(1^\kappa)$. $\mathcal{B}$ is given an access to *ASig.Sign* and *Sig.Sign* oracles under $\overrightarrow{sk}$ and $\overline{sk}$ up to q_s signature queries on both, respectively (as in Definition 2).

We then set the *I-EU-CMA* experiment for *PISB-Generic*, in which $\mathcal{B}$ executes $\mathcal{A}$ as follows:

- *Setup:* Given $(\overrightarrow{pk}, \overline{pk})$, $\mathcal{B}$ sets the *PISB-Generic* public key $PK \leftarrow (\overrightarrow{pk}, \overline{pk})$ as in *PISB-Generic.Kg* algorithm. By Definition 6, $\mathcal{B}$ gives PK to $\mathcal{A}$ and also permits $\mathcal{A}$ to make q_s *PISB-Generic* signature queries.
- Queries: $\mathcal{A}$ queries $\mathcal{B}$ on messages $\overrightarrow{m}_j = (m_{j,1}, \ldots, m_{j,u})$ of her choice for $j = 1, \ldots, q_s$. $\mathcal{B}$ handles these queries as follows:
  (a) Given $\mathcal{A}$'s j-th query $\overrightarrow{m}_j$, $\mathcal{B}$ queries *ASig.Sign* oracle on $\overrightarrow{m}_j$ under $\overrightarrow{pk}$. The *ASig.Sign* oracle returns $s_{j,i} \leftarrow ASig.Sign(sk_i, m_{j,i})$ for $i = 1, \ldots, u$. $\mathcal{B}$ then computes the aggregate signature as $\sigma_j \leftarrow ASig.Agg(\overrightarrow{pk}, \overrightarrow{m}_j, s_{j,1}, \ldots, s_{j,u})$. This step is identical to *PISB-Generic.Init* algorithm, where $\overrightarrow{M}$ in this experiment is comprised of u vectors each with q_s data items.
  (b) $\mathcal{B}$ queries *Sig.Sign* oracle on σ_j under $\overline{pk}$. The *Sig.Sign* oracle returns $s'_j \leftarrow Sig.Sign(\overline{sk}, \sigma_j)$ (as in *PISB-Generic.Sign* algorithm, executed by the server). $\mathcal{B}$ replies $\mathcal{A}$ with $\gamma_j = (\sigma_j, s'_j)$.
- Forgery of $\mathcal{A}$: $\mathcal{A}$ outputs a forgery $(m^*, \beta^* = \langle \sigma^*, s'^* \rangle)$ and wins the *I-EU-CMA* experiment if
  (i) $PISB\text{-}Generic.Ver(PK, m^*, \beta^*) = 1$,
  (ii) $m^* \not\subseteq \{\overrightarrow{m}_1, \ldots, \overrightarrow{m}_{q_s}\}$ or $\exists I \subseteq \{1, \ldots, q_s\} : m^* \subseteq ||_{k \in I} \overrightarrow{m}_k$

If $\mathcal{A}$ loses in the *I-EU-CMA* experiment then $\mathcal{B}$ also loses in the *EU-CMA* experiments for *ASig* and *Sig*, and therefore $\mathcal{B}$ *aborts*. Otherwise, $\mathcal{B}$ proceeds for two possible forgeries as follows:

a) If $m^* \not\subseteq \{\overrightarrow{m}_1, \ldots, \overrightarrow{m}_{q_s}\}$ then $\mathcal{B}$ returns the forgery (m^*, σ^*) against *ASig*, which is non-trivial since $\mathcal{B}$ did not ask m^* to *ASig.Sign*. This forgery is valid since $PISB\text{-}Generic.Ver(PK, m^*, \beta^*) = 1$ implies $ASig.Ver(\overrightarrow{pk}, m^*, \sigma^*) = 1$.
b) If $\exists I \subseteq \{1, \ldots, q_s\} : m^* \subseteq ||_{k \in I} \overrightarrow{m}_k$ then $\mathcal{B}$ returns the forgery (σ^*, s'^*) against *Sig*, which is non-trivial since $\mathcal{B}$ did not ask σ^* to *Sig.Sign*. This forgery is valid since $PISB\text{-}Generic.Ver(PK, m^*, \beta^*) = 1$ implies $Sig.Ver(\overline{pk}, \sigma^*, s'^*) = 1$.

The execution time of $\mathcal{B}$ is that of $\mathcal{A}$ plus the time required to handle $\mathcal{A}$'s queries. That is, for each query of $\mathcal{A}$, $\mathcal{B}$ requests one $ASig$ and Sig signature, whose total costs for handling q_s queries is $q_s \cdot (Op + Op')$. Hence, $t' = O(t) + q_s \cdot (Op + Op')$.

$\mathcal{A}$ does not abort the during the query phase, as the simulation of $\mathcal{B}$ is perfectly indistinguishable. That is, the real and simulated views of $\mathcal{A}$ are identical, and each value in these views are computed identically as described during the experiment. The probability that $\mathcal{A}$ wins the experiment without querying $\mathcal{B}$ is negligible in terms of κ. Therefore, $\mathcal{B}$ wins with the probability ϵ that $\mathcal{A}$ wins. □

Theorem 2. *PISB-CSA-RSA* is (t, q_s, ϵ)-*I-EU-CMA* secure if *RSA* signature scheme is $(t', (2 \cdot l)q_s, \epsilon)$-*EU-CMA* secure, where $t' = O(t) + 2(l \cdot q_s)Exp$ and Exp and l denote modular exponentiation and number of messages in a single *PISB-CSA-RSA* query, respectively.

Proof: Suppose algorithm $\mathcal{A}$ breaks (t, q_s, ϵ)-*I-EU-CMA* secure *PISB-CSA-RSA*. We then construct a simulator $\mathcal{B}$ that breaks $(t', (2 \cdot l)q_s, \epsilon)$-*EU-CMA* secure *RSA* by using $\mathcal{A}$ as subroutine.

We set two separate *EU-CMA* experiments for $\mathcal{B}$, in which it is given *RSA* public keys $pk = (n, e)$ and $\overline{pk} = (\overline{n}, \overline{e})$ and provided signature oracles under their corresponding private keys $sk = (n, d)$ (i.e., oracle $\mathcal{O}_1$) and $\overline{sk} = (\overline{n}, \overline{d})$ (i.e., oracle $\mathcal{O}_2$), respectively. $\mathcal{B}$ will simulate $\mathcal{A}$'s signature queries via $(\mathcal{O}_1, \mathcal{O}_2)$. $\mathcal{B}$ then executes $\mathcal{A}$ for the *I-EU-CMA* experiment for *PISB-CSA-RSA* as follows:

- *Setup:* Given $(pk, \overline{pk})$, $\mathcal{B}$ sets the *PISB-CSA-RSA* public key $PK \leftarrow (pk, \overline{pk})$ as in $\overline{PISB}$-*CSA-RSA.Kg* algorithm. By Definition 6, $\mathcal{B}$ gives PK to $\mathcal{A}$ and allows $\mathcal{A}$ to ask q_s *PISB-CSA-RSA* signatures under PK.
- Queries: $\mathcal{A}$ queries $\mathcal{B}$ on messages $\overrightarrow{m}_j = (m_1, \ldots, m_l)$ of her choice for $j = 1, \ldots, q_s$. $\mathcal{B}$ handles these queries as follows:
  (a) Given $\mathcal{A}$'s j-th query $\overrightarrow{m}_j$, $\mathcal{B}$ queries $\mathcal{O}_1$ on each m_i and obtains corresponding s_i under pk for $i = 1, \ldots, l$ (as in *PISB-CSA-RSA.Init*, data owner).
  (b) $\mathcal{B}$ computes $\sigma' \leftarrow \prod_{i=1}^{l} s_i \bmod n$, $h \leftarrow H(\overrightarrow{m}_j || \overline{pk})$ and $y \leftarrow h + \sigma'$. $\mathcal{B}$ queries $\mathcal{O}_2$ on y under $\overline{pk}$ and obtains γ (as in *PISB-CSA-RSA.Sign*, executed by the server).
- Forgery of $\mathcal{A}$: $\mathcal{A}$ outputs a forgery (m^*, γ^*) and wins the *I-EU-CMA* experiment if
  (i) $\overline{PISB}$-*CSA-RSA.Ver*$(PK, m^*, \gamma^*) = 1$,
  (ii) $m^* \not\subseteq \{\overrightarrow{m}_1, \ldots, \overrightarrow{m}_{q_s}\}$ or $\exists I \subseteq \{1, \ldots, q_s\} : m^* \subseteq ||_{k \in I} \overrightarrow{m}_k$
  If $\mathcal{A}$ loses in the *I-EU-CMA* experiment then $\mathcal{B}$ also loses in the *EU-CMA* experiments for *RSA* against $\mathcal{O}_1$ and $\mathcal{O}_2$, and therefore $\mathcal{B}$ *aborts*. Otherwise, $\mathcal{B}$ computes $y^* \leftarrow (\gamma^*)^{\overline{e}} \bmod \overline{n}$ and $\sigma^* \leftarrow y^* - H(m^* || \overline{pk})$ and continues as follows:

a) If $m^* \not\subseteq \{\overrightarrow{m}_1, \ldots, \overrightarrow{m}_{q_s}\}$ then $\mathcal{B}$ returns the forgery (m^*, s^*) against $\mathcal{O}_1$, where s^* is computed from σ^* by removing the corresponding individual signatures of data items in m^* that have been queried before (if m^* is not a vector then use s^* itself). This forgery is non-trivial since $\mathcal{B}$ did not ask m^* to $\mathcal{O}_1$ during the experiment. $\mathcal{B}$ also returns the forgery (y^*, γ^*) against $\mathcal{O}_2$, which is non-trivial since $\mathcal{B}$ did not ask y^* to $\mathcal{O}_2$ during the experiment. Both forgeries are valid since *PISB-CSA-RSA.Ver*$(PK, m^*, \gamma^*) = 1$ implies m^* and y^* are valid under pk and $\overline{pk}$, respectively.

b) If $\exists I \subseteq \{1, \ldots, q_s\} : m^* \subseteq ||_{k \in I} \overrightarrow{m}_k$ holds then $\mathcal{B}$ returns the forgery (σ^*, γ^*) against $\mathcal{O}_2$. This forgery is valid and non-trivial as discussed the above.

The execution time and probability analysis are similar to Theorem 1 (i.e., the simulation is perfectly indistinguishable) and therefore are not repeated here. □

6 Performance Analysis and Comparison

In this section, we present the performance analysis of $PISB$ schemes and compare them with the existing alternatives. Table 1 (see Section 1) and Table 2 summarize our performance analysis and comparison.

Computational Overhead: In $PISB\text{-}Generic$, the server requires a $Sig.Sign$ plus the aggregation of l individual $ASig$ signatures. The client requires a $Sig.Ver$ plus $ASig.Ver$ for l data items. In $PISB\text{-}CSA\text{-}RSA$, the server requires a $C\text{-}RSA.Sig$ computation plus l modular multiplications. The client requires a single $RSA.Ver$ plus $C\text{-}RSA.Ver$ for l data items.

The $PISB\text{-}CSA\text{-}RSA$ is the most *client efficient scheme* among all of its counterparts, since it only requires RSA and $C\text{-}RSA$ signature verifications with a small exponent (e.g., $e = 3$). Therefore, it is an ideal choice for battery or computational limited queriers such as mobile devices. Its server side overhead is also plausible and smaller than SKROOT-based scheme. The end-to-end delay of $PISB\text{-}CSA\text{-}RSA$ ($l = 10$) is *50, 65, 41, and 55 times lower* than that of $PISB\text{-}Generic$, GQ-based, SKROOT-based and iBGLS schemes, respectively.

The $PISB\text{-}Generic$ can be instantiated with various signature schemes, which allows different performance trade-offs. BGLS and ECDSA combination achieves both small signature size and high server efficiency, which is a desirable configuration for many applications. Using ECDSA with pre-computation offers *the smallest server response time among all of its counterparts*. However, BGLS increases the signature verification cost, incurring high overhead to the resource-constrained verifiers. Another alternative is to combine $C\text{-}RSA$ and ECDSA, which achieves *both very low server and client computational overheads with the cost of a slightly larger signature size*. ECDSA can be replaced with an online/offline [17] or one-time signature [18], which offers even faster server response with the expense of a very large signature size.

In both $PISB$ and Mykletun et. al. schemes [3, 7], the initialization phases are performed *offline (before the deployment)* by the data owners, whose costs are similar for all schemes and therefore are omitted in this comparison. We focus on the client/server overheads, since they are the online (real-time) overheads and the most important performance metrics for our envisioned applications.

Communication and Storage Overhead: $PISB$ schemes do not require any multi-round communication to achieve the immutability. Therefore, their signature overhead is the aggregate signature plus the protection signature in $PISB\text{-}Generic$, and only the aggregate signature itself in $PISB\text{-}CSA\text{-}RSA$. The private and public key sizes are the sum of that of their base signature schemes.

$PISB\text{-}Generic$ with BGLS and ECDSA has the smallest key and signature sizes among its counterparts with the exception of iBGLS (which has a much larger

Table 2. The client and server overhead of $PISB$ and its counterparts for l data items (analytical)

-	Client Comp.	Server Comp.	Sig.	SK	PK
$PISB$-$Generic$	$Sig.Ver + ASig.Ver^l$	$Sig.Sign + ASig.Agg^l$	$\|\sigma\|$	$\|sk\|$	$\|pk\|$
$PISB$-CSA-RSA	$3Mul + l(H + Mul)$	$Exp\|_{\|n\|}^{n\|} + l \cdot Mul$	$\|n\|$	$2\|n\|$	$2\|n\|$
GQ-based	$3Exp\|_{\|n\|}^{b\|} + l(H + Mul)$	$3Exp\|_{\|n\|}^{b\|} + l \cdot Mul$	$9\|n\|$	$\|b\| + \|n\|$	$\|b\| + \|n\|$
SKROOT-based	$4Exp\|_{\|2n\|}^{n\|} + l(H + Mul)$	$4Exp\|_{\|2n\|}^{n\|} + l \cdot Mul$	$4\|n\|$	$5\|n\|$	$\|n\|$
iBGLS	$(l + 1)(BM + H)$	$EMul + l \cdot EAdd$	$\|q'\|$	$\|q'\|$	$\|p'\| + \|q'\|$

• *Notation*: $Exp|_{|y|}^{|x|}$ denotes a modular exponentiation with a modulus and exponent sizes $|y|$ and $|x|$, respectively. Mul denotes modular multiplication under modulus n. BM, $EAdd$, and $EMul$ denote ECC bilinear map, scalar addition and scalar multiplication over modulus q', respectively. $ASig.Ver^l$ denotes the aggregate signature verification for l items (the notation applies to $ASig.Agg$). We omit constant number of low-cost operations if there is an expensive operation (e.g., a single H or Mul is omitted if there is an Exp). We use double-point scalar multiplication for ECDSA verifications ($1.3 \cdot Emul$ instead of $2 \cdot EMul$). $|\sigma|$, $|sk|$ and $|pk|$ denote the bit lengths of signature, private key and public key for Sig, respectively (Sig is selected as ECDSA with $|q'|$ in Table 1).
• *Parameters*: Given $\kappa = 80$, we select $|n| = 1024$, $|H| = 160$, $|q'| = 160$, $|p'| = 512$, $|b| = 30$.
• *Multi-round schemes*: GQ-based scheme needs three communication rounds (each three passes) to achieve $\kappa \geq 80$, in which each pass needs to transmit an element from Z_N^*.
• *Measurements*: Table 1 (see Section 1.3) shows the estimated execution times for $l = 10$ data items (query elements). Estimated execution times are measured on a computer with an Intel(R) Core(TM) i7 Q720 at 1.60GHz CPU and 2GB RAM running Ubuntu 10.10. We used MIRACL [11] library.

end-to-end delay and client computation overhead). $PISB$-CSA-RSA also has much smaller signature and key sizes than that of GQ-based and SKROOT-based schemes. Despite GQ-based scheme is client and server computationally efficient, it is not practical due to its multi-round communication requirement (introduces a substantial communication delay as shown in Table 1, Section 1). Note that multi-round communication is undesirable for wireless and low bandwidth applications due to the packet loss potential.

Overall, our analysis indicates that $PISB$ schemes are much more efficient and practical than Mykletun et. al. immutable signatures for outsourced database systems.

7 Related Work

Our schemes rely on aggregate signatures as the building block. Aggregate signatures aggregate n individual signatures associated with n different users (or data items) into a single, compact signature. The first aggregate signature scheme was proposed in [8], and then several new schemes achieving more advanced properties were developed (e.g., sequentiality [19], ID-based for low storage overhead [20]). We discussed aggregate, sequential aggregate [15, 21] and condensed signatures [3, 7] in Section 2.

Mykletun et. al. proposed the first immutable aggregate signatures [3, 7], which have been extensively compared with our schemes in Section 6. Immutable signatures serve as a cryptographic tool for various data outsourcing applications such as database-as-a-service [22] and data protection methods (e.g., [23, 24]). They are also used with other cryptographic primitives such as forward-secure signatures to obtain forward-secure and aggregate logging systems (e.g., [25–27]). Immutability techniques used in these schemes require linear overhead and therefore are not suitable for our envisioned applications.

Note that our work focuses on the authentication and integrity services. There are extensive studies on the data privacy for outsourced database systems (e.g., [28, 29]), which are complementary to our work.

8 Conclusion

In this paper, we developed new cryptographic schemes called $PISB$, which provide practical immutable signatures for outsourced databases. $PISB\text{-}CSA\text{-}RSA$ provides the signature immutability with a very low verifier computational overhead, which is ideal for battery/computation limited queriers (e.g., mobile devices). It also offers a very low end-to-end delay, which is desirable for task-intensive applications by increasing the service quality. $PISB\text{-}Generic$ offers various options such as signature pre-computability with its generic construction, which enables a quick server response during query peak times. Both $PISB$ schemes are non-interactive and have small signature sizes. We demonstrate that $PISB$ schemes are much more efficient than previous immutable signatures. Hence, $PISB$ schemes are ideal choices for providing immutability, authentication and integrity services for outsourced database systems.

References

1. Hacigumus, H., Iyer, B., Mehrotra, S.: Providing database as a service. In: Proceedings of the 18th International Conference on Data Engineering, ICDE 2002, Washington, DC, USA, pp. 29–38 (2002)
2. Sion, R.: Secure data outsourcing. In: Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB), pp. 1431–1432 (2007)
3. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and integrity in outsourced databases. Transaction on Storage (TOS) 2(2), 107–138 (2006)
4. Patel, A.A., Jaya Nirmala, S., Mary Saira Bhanu, S.: Security and availability of data in the cloud. In: Meghanathan, N., Nagamalai, D., Chaki, N. (eds.) Advances in Computing & Inform. Technology. AISC, vol. 176, pp. 255–261. Springer, Heidelberg (2012)
5. Wang, H., Lakshmanan, L.V.S.: Efficient secure query evaluation over encrypted xml databases. In: Proceedings of the 32nd International Conference on Very Large Data Bases, VLDB 2006, pp. 127–138 (2006)
6. Goodrich, M.T., Mitzenmacher, M., Ohrimenko, O., Tamassia, R.: Privacy-preserving group data access via stateless oblivious ram simulation. In: Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 157–167 (2012)
7. Mykletun, E., Narasimha, M., Tsudik, G.: Signature bouquets: Immutability for aggregated/condensed signatures. In: Samarati, P., Ryan, P.Y.A., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 160–176. Springer, Heidelberg (2004)
8. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
9. Guillou, L.C., Quisquater, J.-J.: A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 216–231. Springer, Heidelberg (1990)
10. American Bankers Association: ANSI X9.62-1998: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm, ECDSA (1999)
11. Shamus: Multiprecision integer and rational arithmetic c/c++ library (MIRACL), `http://www.shamus.ie/`

12. Naccache, D., M'Raïhi, D., Vaudenay, S., Raphaeli, D.: Can D.S.A. be improved? Complexity trade-offs with the digital signature standard. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 77–85. Springer, Heidelberg (1995)
13. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman & Hall/CRC (2007)
14. Bellare, M., Rogaway, P.: The exact security of digital signatures: How to sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
15. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (2004)
16. Ding, X., Tsudik, G.: Simple identity-based cryptography with mediated rsa. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 193–210. Springer, Heidelberg (2003)
17. Catalano, D., Di Raimondo, M., Fiore, D., Gennaro, R.: Off-line/on-line signatures: Theoretical aspects and experimental results. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 101–120. Springer, Heidelberg (2008)
18. Reyzin, L., Reyzin, N.: Better than BiBa: Short one-time signatures with fast signing and verifying. In: Batten, L.M., Seberry, J. (eds.) ACISP 2002. LNCS, vol. 2384, pp. 144–153. Springer, Heidelberg (2002)
19. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006)
20. Boldyreva, A., Gentry, C., O'Neill, A., Yum, D.: Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In: Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS 2007), pp. 276–285. ACM (2007)
21. Zhu, H., Zhou, J.: Finding compact reliable broadcast in unknown fixed-identity networks (short paper). In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 72–81. Springer, Heidelberg (2006)
22. Mykletun, E., Tsudik, G.: Aggregation queries in the database-as-a-service model. In: Damiani, E., Liu, P. (eds.) Data and Applications Security 2006. LNCS, vol. 4127, pp. 89–103. Springer, Heidelberg (2006)
23. Samarati, P., di Vimercati, S.D.C.: Data protection in outsourcing scenarios: issues and directions. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010, pp. 1–14 (2010)
24. Thompson, B., Haber, S., Horne, W.G., Sander, T., Yao, D.: Privacy-preserving computation and verification of aggregate queries on outsourced databases. In: Goldberg, I., Atallah, M.J. (eds.) PETS 2009. LNCS, vol. 5672, pp. 185–201. Springer, Heidelberg (2009)
25. Ma, D., Tsudik, G.: A new approach to secure logging. ACM Transaction on Storage (TOS) 5(1), 1–21 (2009)
26. Yavuz, A.A., Ning, P., Reiter, M.K.: BAF and FI-BAF: Efficient and publicly verifiable cryptographic schemes for secure logging in resource-constrained systems. ACM Transaction on Information System Security 15(2) (2012)
27. Yavuz, A.A., Ning, P., Reiter, M.K.: Efficient, compromise resilient and append-only cryptographic schemes for secure audit logging. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 148–163. Springer, Heidelberg (2012)
28. Ostrovsky, R., Skeith III, W.E.: A survey of single-database private information retrieval: techniques and applications. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 393–411. Springer, Heidelberg (2007)
29. Popa, R.A., Redfield, C.M.S., Zeldovich, N., Balakrishnan, H.: Cryptdb: protecting confidentiality with encrypted query processing. In: Proc. of the 23rd ACM Symposium on Operating Systems Principles, SOSP 2011, New York, NY, USA, pp. 85–100 (2011)

Optimal Re-encryption Strategy for Joins in Encrypted Databases

Florian Kerschbaum, Martin Härterich, Patrick Grofig, Mathias Kohler, Andreas Schaad, Axel Schröpfer, and Walter Tighzert

SAP Applied Research
Vincenz-Priessnitz-Str. 1 Karlsruhe, Germany
firstname.lastname@sap.com

Abstract. In order to perform a join in a deterministically, adjustably encrypted database one has to re-encrypt at least one column. The problem is to select that column that will result in the minimum number of re-encryptions even under an unknown schedule of joins. Naive strategies may perform too many or even infinitely many re-encryptions. We provide two strategies that allow for a much better performance. In particular the asymptotic behavior is $O(n^{3/2})$ resp. $O(n \log n)$ re-encryptions for n columns. We show that there can be no algorithm better than $O(n \log n)$. We further extend our result to element-wise re-encryptions and show experimentally that our algorithm results in the optimal cost in 41% of the cases.

Keywords: Encrypted Database, Proxy Re-encryption, Adjustable Join.

1 Introduction

Recently, encrypted databases [2,7,9] that provide the client with additional protection in the cloud or database-as-a-service setting have emerged. In these databases all data are encrypted at the client – where also the keys are exclusively stored – and queries are performed over encrypted data. In order to perform a selection, e.g. `SELECT x FROM y WHERE z = 1`, the columns for selection (`z` in the example) needs to be encrypted using deterministic encryption, i.e., a plaintext always enciphers to the same ciphertext. In order to perform a join, e.g. `SELECT a.b, c.d FROM a, c WHERE a.e = c.f`, the columns for the join (`e` and `f` in the example) need to be encrypted using the same key. This is achieved using an operation called proxy re-encryption (PRE) [3]. In PRE a proxy translates a ciphertext under one key K_A to a ciphertext under another key K_B without knowing either of the two keys.

The encrypted database performs this PRE when required, i.e. when it has received a query performing a join over two previously unjoined columns. The client then issues a PRE key to the databases which re-encrypts at least one column, such that both columns are encrypted under the same key. The (equi-) join operator can then operate using the same algorithm as on an unencrypted database.

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 195–210, 2013.

The reason for dynamically adjusting the database encryption to the queries is that this PRE reveals additional information to an attacker observing the database. He now obtains additional ciphertexts he can use in cryptanalysis of the keys. Deterministic encryption is only provably secure in high-entropy domains [1], such that these additional ciphertexts may be of significant help.

When the database issues the PRE key, it has to choose a column which to re-encrypt; in the example either `e` or `f`. Furthermore, it has to make this choice under an unknown schedule of future joins. Naive approaches may even lead to an infinite number of re-encryptions.

In this paper we present a re-encryption column selection algorithm that results in at most $O(n^{3/2})$ re-encryptions for n columns under any schedule of join operations. Furthermore we give a second algorithm that occasionally leads to re-encryption of both columns to be joined, but which results in a better bound of at most $O(n \log n)$ re-encryptions (where, of course, the PRE of two columns counts as two PREs). We show that this is the best possible bound we can achieve under the assumption of an a priori unknown schedule.

The remainder of the paper is structured as follows. We review related work and background on encrypted databases in Section 2. Section 3 gives an introduction to the problem using the naive approaches. Our algorithms including proofs of bounds on the number of re-encryptions are shown in Section 4 . We show our experimental evaluation in Section 5. Section 6 summarizes our findings.

2 Related Work

The design goal of encrypted databases in the database-as-a-service setting is to move the encryption layer above the query processing layer. All query process operators are supposed to work on encrypted data. This ensures that (almost) any query can be processed on the encrypted data.

The first such database was introduced in [7]. It provided special operators for many queries and it was necessary to post-process and filter many queries. This was improved by [2] where the database operators remain unchanged. This enables using existing, commercial database systems for encryption in the cloud. Nevertheless, it requires the use of special encryption schemes such as order-preserving encryption [4].

In [2] the encryption was replaced by using the identifiers in the data dictionary and leaving the dictionary at the client. This requires even less modification to the database and is as secure as order-preserving encryption. Nevertheless, it does not allow aggregation.

Order-preserving [4] or even deterministic encryption is commonly not considered very secure. It is therefore advisable to encrypt only the columns necessary for performing the queries using these encryption schemes. Yet, these columns may not be known in advance and the database must adjust its encryption state to the queries performed.

In order to adjust the encryption to the queries on the fly, [9] proposed the use of onion encryption. While it is possible to choose an optimally secure encryption, if all queries are known upfront, it is difficult to do so, if any queries are processed on the fly. Therefore each data item is encrypted using *onion encryption* and decrypted to the corresponding onion layer on the fly. Our encryption onion is composed of the following layers:

- *L3 – Randomized Encryption*: IND-CPA secure encryption allowing only retrieval using AES encryption in CBC mode.
- *L2 – Deterministic Encryption*: Allows processing of equality comparisons. In deterministic encryption one plaintext always enciphers to the same ciphertext.
- *L1 – Order-Preserving Encryption*: Allows processing of greater-than comparisons using the encryption scheme of [4].
- *L0 – Data*: The data to be encrypted.

The layers of the onion represent a strict order, i.e. the lower the layer the less secure, but also the more operations it supports. It is important that each lower layer supports all operations that the upper layer supports, such that a decryption never needs to be undone.

The client analyzes each query before executing it. It determines the necessary encryption layer in the onion encryption in the database. Then, before sending the query, the client performs the decryption of the column to that onion layer. No encryption from a lower to a higher encryption layer is ever performed. As such, the level (layer) of encryption in the database is dynamically adjusted to the queries processed.

In order to perform an equi-join operation data is decrypted to the deterministic layer L2, but different columns may still be encrypted using different keys. In this case proxy re-encryption (PRE) [3] can be performed. In PRE a proxy translates a ciphertext encrypted under one key into a ciphertext under another key without decrypting it first, i.e. the proxy does not learn the plaintext or any of the two keys. The proxy does, however, learn a relation between the two keys, such that the security against cryptanalysis is reduced to the secrecy of one key.

We use a proxy re-encryptable, deterministic encryption scheme. An example is the symmetric Pohlig-Hellman encryption [8].

Let p be the prime order of a group $\mathbb{Z}_p$. Let m be an element of $\mathbb{Z}_p$ representing a message to encipher. Let $ord(p) = p-1$ be the order of the multiplicative group $\mathbb{Z}_p^*$ of invertible elements in $\mathbb{Z}_p$. We uniformly choose an element e of $\mathbb{Z}_{ord(p)}$, such that $\gcd(e, ord(p)) = 1$. We encrypt m to the ciphertext c as

$$c = m^e \bmod p$$

We decrypt the ciphertext c as

$$m = c^{e^{-1} \bmod ord(p)} \bmod p$$

The element e is the secret key. Let two database columns A and B have two different keys e_A and e_B, respectively, but both encrypted at the deterministic

layer L2. Furthermore, assume we have chosen to re-encrypt column A to the key of columns B. We then compute the PRE key k as

$$k = e_A^{-1} e_B \bmod ord(p)$$

The database can now perform the proxy re-encryption operation. Each ciphertext c of column A is re-encrypted to a ciphertext c' using the PRE key k as

$$c' = c^k = m^{e_A k} = m^{e_A e_A^{-1} e_B \bmod ord(p)} = m^{e_B} \bmod p$$

In [9] the authors suggest to use this encryption scheme, but on elliptic curves. Unfortunately, their encryption scheme is not decryptable, since they use g^m instead of m. This may require additional storage on the database. Old-fashioned Pohlig-Hellman encryption over multi-precision integers is decryptable. The authors also provide a cryptanalysis of their scheme in [10] under an adjustable join attack. This extends to Pohlig-Hellman encryption.

3 Naive Approaches

Let there be a database with n columns A, B, C and so forth. Initially each column is deterministically encrypted under its own key. We then perform a number of queries on the database, possibly involving join operations. We write

JOIN(A, B)
JOIN(B, C)

for first joining columns and A and B and then columns B and C. Joins with $k \geq 2$ columns can be simulated by joining $k - 1$ pairs. However, the order in which the pairs are chosen is not arbitrary. We will give more details on how to do this efficiently in section 4.8. In order to perform a join operation, at least one column needs to be re-encrypted. We write

JOIN(A, B): $A \leftarrow B$

if column A gets re-encrypted to the key of column B.

The order of the two columns in the join operation is determined by the query string. Therefore the database connector has to choose the right column to re-encrypt.

We consider the effect of a few simple, straight-forward strategies. This should highlight that such simple strategies – while plenty – do not result in the best performance. The first strategy is to always use the first column in the query string. Assume the following schedule

JOIN(A, B): $A \leftarrow B$
JOIN(A, C): $A \leftarrow C$
JOIN(A, B): $A \leftarrow B$
JOIN(A, C): $A \leftarrow C$
$\vdots$

Clearly, this may lead to infinitely many re-encryptions and is therefore inadvisable. There is a maximum number of re-encryptions for any schedule and ideally this should be achieved.

Next, consider a total order of columns, e.g. lexicographically. We always re-encrypt the lower to the upper. Now, assume the following schedule

$$
\begin{aligned}
&\textsc{Join}(A, B)\colon A \leftarrow B\\
&\textsc{Join}(B, C)\colon B \leftarrow C\\
&\textsc{Join}(A, B)\colon A \leftarrow C\\
&\textsc{Join}(C, D)\colon C \leftarrow D\\
&\textsc{Join}(B, C)\colon B \leftarrow D\\
&\textsc{Join}(A, B)\colon A \leftarrow D\\
&\quad\vdots
\end{aligned}
$$

This leads to $\frac{n(n-1)}{2}$, i.e. $O(n^2)$ re-encryptions. Clearly, this is sub-optimal, since the same schedule can be completed with $n-1$ PREs in the following way

$$
\begin{aligned}
&\textsc{Join}(A, B)\colon A \leftarrow B\\
&\textsc{Join}(B, C)\colon C \leftarrow B\\
&\textsc{Join}(A, B)\\
&\textsc{Join}(C, D)\colon D \leftarrow B\\
&\textsc{Join}(B, C)\\
&\textsc{Join}(A, B)\\
&\quad\vdots
\end{aligned}
$$

We follow this idea in the formal definition of our algorithms in the next section.

4 Algorithms

4.1 Data Structures

In our algorithms we store two types of objects: columns and keys. For the sake of exposition, we store these objects as database table rows, but it could as well be Java objects or C/C++ structures. Storing them in database tables enables to be shared between multiple clients of the encrypted database and ensures persistence between different runs of the application of one client.
In the table *Keys* we store

- *KeyId*: An unique identifier for the key. It is the primary database key of the table.
- *Rank*: A rank of the key.

In the table *Columns* we store

- *ColumnId*: An unique identifier for the column. It may be or be generated from the name of the column `TABLE.COLUMN` which enables searching using the name. It is the primary key of the table.

- *Cost*: The cost of re-encrypting this column. For now we assume uniform cost of 1 for each column. We discuss non-uniform costs in 4.9.
- *KeyId*: The identifier of the associated key. This is a foreign key of this table and the primary key of the $Keys$ table.

4.2 Initialization

Algorithm 1. Initialization

```
function INIT
    for all column do
        cost ← 1
        INSERT keyId, cost INTO Keys
        INSERT columnId, cost, keyId INTO Columns
    end for
end function
```

We initialize each column with its own key and cost of 1. Each key is initialized with the cost of the associated column. This is performed as in Algorithm 1. When uploading the encrypted data into the database, the data of each column will be encrypted under its associated key. Subsequently we can perform queries with optional joins.

4.3 Key Retrieval

Algorithm 2. Key Retrieval

```
function GETKEY(column)
    return SELECT keyId FROM Columns WHERE columnId = column
end function
```

When we perform a query we must encrypt parameters and decrypt return values. We therefore need to retrieve the corresponding key identifier for the accessed columns. Algorithm 2 shows that this can be performed using a simple query.

4.4 Column Selection

When performing a join between two columns A and B we need to select one for re-encryption. The function in Algorithm 3 returns the identifier of the column to be re-encrypted. It has already updated the data structure to reflect its new key – that of the other column. We call the column that does not get re-encrpyted the steady column.

Algorithm 3. Column Selection

```
    function JOIN(columnA, columnB)
        keyA ← GETKEY(columnA)
        keyB ← GETKEY(columnB)
        if keyA = keyB then
 5:         return null
        end if
        rankA ← SELECT rank FROM Keys WHERE keyId = keyA
        rankB ← SELECT rank FROM Keys WHERE keyId = keyB
        if rankA > rankB then
10:         lower ← columnB
            (lowerKey, lowerRk) ← (keyB, rankB)
            (upperKey, upperRk) ← (keyA, rankA)
        else
            lower ← columnA
15:         (lowerKey, lowerRk) ← (keyA, rankA)
            (upperKey, upperRk) ← (keyB, rankB)
        end if
        lowerCost ← SELECT cost FROM Columns WHERE columnId = lower
        UPDATE Keys SET rank = lowerRk − lowerCost WHERE keyId = lowerKey
20:     UPDATE Keys SET rank = upperRk + lowerCost WHERE keyId = upperKey
        UPDATE Columns SET keyId = upperKey WHERE columnId = lower
        if lowerRank − lowerCost = 0 then
            DELETE FROM Keys WHERE keyId = lowerKey
        end if
25:     return lower
    end function
```

We make the choice simply by the rank of the key. The column with the key with the lower rank gets re-encrypted. Afterwards, we add the cost of the re-encrypted column to the rank of the steady column and subtract the same cost from the rank of the key of the re-encrypted column.

If the rank of the key of the re-encrypted column reaches 0, then we can delete the key entry, since it no longer encrypts any column.

Note that for any (infinite) schedule of joins the algorithm leads to a finite number of proxy re-encryptions only (i.e. it returns a value different from *null* only a finite number of times). This can be seen easily if we consider a variant of the algorithm where we omit the deletion of keys of rank 0 (lines 22 through 24). Then the sum of the absolute values of differences of the ranks over all pairs of keys is a non-negative integer which is bounded (by $\binom{n}{2}$ times the maximal possible rank) and which increases by at least 2 in each re-encryption step.

The algorithm is reminiscent of the Union-Find algorithm [6], but we do not join the entire group, just the selected column. This reduces the cost for one join operation, since we need to re-encrypt at most one column and not an entire group, but does not increase worst-case cost – due to the re-encryption of columns in shrinking groups – as we show in our analysis next.

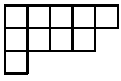
4.5 Analysis

We analyse the worst-case performance of our re-encryption selection algorithms. Here (and also in the analysis of the enhanced algorithm in section 4.7) we obtain the maximal security simply by taking the maximal possible number of different keys given the required functionality, i.e. whenever two columns are not joined in any previous step of the schedule they remain encrypted under different keys.

Let there be n columns and let $t(n)$ be the maximum number of re-encryptions that Algorithm 3 performs where the maximum is taken over all possible schedules of join operations. We now provide a proof that $t(n) = O(n^{3/2})$.

Theorem 1. *Algorithm 3 needs at most $2n^{3/2}$ re-encryptions for any schedule. This bound is optimal in the sense that the asymptotic behavior of $t(n)$ is $O(n^{3/2})$.*

Before we start with the proof of this theorem we recall some notation: A *partition* of n is a sequence $\lambda = (\lambda_1, \ldots, \lambda_k)$ where $\lambda_1 \geq \ldots \geq \lambda_k \geq 1$ are integers such that $\lambda_1 + \lambda_2 + \ldots + \lambda_k = n$. The partition λ can graphically be represented by a *Young diagram* which is composed of k rows containing $\lambda_1, \ldots, \lambda_k$ boxes (where the *top* row has length λ_1). By abuse of notation we use λ to denote both the partition and the associated Young diagram as in

$\lambda = (5, 4, 1) =$ 

The *dominance order* $\trianglerighteq$ on partitions is defined by

$$\lambda \trianglerighteq \mu : \Longleftrightarrow \ \forall i \geq 1 : \lambda_1 + \ldots + \lambda_i \geq \mu_1 + \ldots + \mu_i \ .$$

(Here $\lambda_i = 0$ resp. $\mu_i = 0$ for i greater than the number of rows of λ resp. μ.) It is well known (see [5]) that $\lambda \trianglerighteq \mu$ iff λ can be obtained from μ by succesively moving single boxes from lower to higher rows. The set of all partitions of n together with the dominance order $\trianglerighteq$ is a poset L_n and $(1, \ldots, 1)$ resp. (n) is the unique minimal resp. maximal element of L_n.

Partitions as well as the dominance order occur naturally in our algorithm: Let $\lambda_1 \geq \ldots \geq \lambda_k \geq 1$ be the sizes of the groups at any time during the execution of the algorithm (ordered non-increasingly). Then clearly $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_k)$ is a partition of n.

This way for every given schedule of join operations the Algorithm 3 produces a series of partitions $\lambda^{(1)}, \lambda^{(2)}, \ldots$ of n. The first partition in the sequence is always $(1, 1, \ldots, 1)$ (n ones) and the schedule can be further extended as long as there is more than one group left i.e. until it reaches the maximal partition (n).

We observe how λ changes when a single step of the algorithm is applied. We remove one element (database column) from one group and add it to another group of at least the same size. In terms of the associated Young diagrams this amounts simply to moving one box up into a higher row. In other words, the series of partitions derived from a join schedule is an increasing (w.r.t. $\trianglerighteq$) chain in the poset L_n.

On the other hand for any series of partitions $\lambda^{(1)}, \lambda^{(2)}, \ldots$ of n such that $\lambda^{(i+1)}$ is obtained from $\lambda^{(i)}$ by moving-up one box we can easily derive a join schedule that produces exactly this series of tableau in the way described above. To this end we just annotate each box with a database column label at the begin. Thereafter, we interpret moving-up such a box (keeping its label) by executing a join of the corresponding database column with any other column in the target row.[1]

Therefore, finding the worst case number of steps of our algorithm is equivalent to finding the longest totally ordered chain in the lattice L_n of partitions. Hence the theorem follows directly from

Proposition 1. *(a) The longest increasing sequence in the poset $(L_n, \trianglerighteq)$ consists of at most $2n^{3/2}$ elements.*
(b) The longest increasing sequence in the poset consists of at least $\frac{2}{3}n^{3/2}$ elements.

Proof. (a) Let $k = [\sqrt{n}]$ and consider an increasing (w.r.t. $\trianglerighteq$) series of Young diagrams of maximal length. We obtain this maximal length by counting the total number of move-up operations of boxes (and adding 1 for the initial diagram). To do so we have a look at the path of each of the n boxes individually. The key observation is that in each step the box not only moves up, but also moves to the right at least one place. Therefore after at most k of its moves it has reached (the end of) a row in the Young diagram with length $\geq k+1$. Since there are only $n < (k+1)^2$ boxes altogether this must be a row with index less than $k+1$. Hence, the box we considered can from now on only have $k-1$ further moves which is a total of $2k-1$.

This holds for every box and hence we obtain $n(2k-1)+1 \leq 2n^{3/2}$ as an upper bound on the length of the given sequence.

(b) We write $n = \frac{1}{2}k(k+1)+r$ for suitable non-negative integers k and $r \leq k$. Then we can construct a sequence of "triangular shaped" Young diagrams:

$$(1,\ldots,1) \to (2,1,\ldots,1) \to (3,2,1,\ldots,1) \to (4,3,2,1,\ldots,1) \to \cdots$$
$$\cdots \to (k,k-1,\ldots,2,1,\ldots,1)$$

(In the last diagram there are $1+r$ ones at the end.) Each step in this sequence further decomposes into a certain number of move-up operations for single boxes. Now let's count these single move-up operations . To go from $(j,j-1,\ldots,2,1,\ldots,1)$ to $(j+1,j,\ldots,2,1,\ldots,1)$ we need to move one box j times, the next box $j-1$ times and so on. This sums up to $\frac{1}{2}j(j+1)$. Altogether we have $\sum_{j=1}^{k} \frac{j(j+1)}{2} = \frac{k(k+1)(2k+7)}{12}$ single moves.

Next we move the "remaining" r ones to the first row. This amounts to rk single moves.

We continue from $(k+r,k-1,\ldots,2,1)$ to (n) in a symmetric (but reverse) way. Just interchange the role of rows and columns in the diagrams. This adds

[1] Caution: Be aware that although this reverse construction may suggest so, in the procedure of going from join schedules to Young diagrams a box is *not* associated with a fixed column and a row is *not* associated with a fixed key!

another $\frac{k(k+1)(2k+7)}{12}$ and hence we get the lower bound $\frac{k(k+1)(2k+7)}{6} + kr + 1$ for the length of the maximal chain. From this we get the claim by a direct calculation. □

Algorithm 3 is very well suited for practical purposes. We give some experimental data of its behavior in section 5.

4.6 Enhanced Version of the Algorithms

We show how to improve the worst case behavior of the algorithm. Note that choosing a key that is neither of the two columns but from a third column is *in general* not a viable option. While this may decrease the overall cost, it may also decrease security. Consider an example where columns A and B are joined under C's key.

$\textsc{Join}(E, C)$: $E \leftarrow C$
$\textsc{Join}(D, C)$: $D \leftarrow C$
$\textsc{Join}(A, B)$: $A \leftarrow C$, $B \leftarrow C$

Clearly, if this schedule continues with

$\textsc{Join}(B, C)$
$\textsc{Join}(A, C)$

then the overall cost is optimal, but the operation is speculative in terms of security. If the schedule does not continue, the adversary is given more information. All columns are encrypted under the same key. He now can use all of them for cryptanalysis.

In the alternative where we replace the third join operation with

$\textsc{Join}(A, B)$: $A \leftarrow B$

there are two remaining, disjunct keys: one for C, D, E and one for A, B. Clearly, this complicates cryptanalysis. Choosing one of the two keys of the joined columns always yields the minimal amount of ciphertexts for cryptanalysis, since at least one re-encryption is necessary in order to perform the join.

As a consequence we only consider re-encryption selection algorithms as *admissible* that guarantee that two columns have different keys unless there is a chain of (previous) joins which links these two columns.

Now for our enhanced algorithm we group columns not by the fact that they share a common key but by the fact that there is a chain of previous join operations that links one column to another. To distinguish this from the *groups* (cf. section 4.4) we considered before we will call a *cluster* of columns (at any given time) the set of columns that is connected w.r.t. previous joins. Note that clusters are unions of groups. Let's call a *cluster key* the (common) key of the largest group in a cluster.

We modify our data structures and algorithms to be able to account for cluster keys by introducing the additional column

– *ClusterKeyId*: The identifier of key associated to the cluster this column belongs to.

in the table *Columns*. During initialization the cluster key of a column gets the same value as the key: Hence the insert statement of Algorithm 1 now reads

INSERT *columnId*, *cost*, *keyId*, *keyId* INTO *Columns*

Yet another algorithm (similar to Algorithm 2) defines a function GETCLUSTERKEY to extract the *ClusterKeyId* for a column.

Algorithm 4. Column Selection (enhanced)

```
    function JOIN2(columnA, columnB)
        if GETKEY(columnA) = GETKEY(columnB) then
            return null
        end if
 5:     keyA ← GETCLUSTERKEY(columnA)
        keyB ← GETCLUSTERKEY(columnB)
        rankA ← SELECT rank FROM Keys WHERE keyId = keyA
        rankB ← SELECT rank FROM Keys WHERE keyId = keyB
        if rankA > rankB then
10:         lower ← columnB
            (lowerKey, lowerRk) ← (keyB, rankB)
            upper ← columnA
            (upperKey, upperRk) ← (keyA, rankA)
        else
15:         lower ← columnA
            (lowerKey, lowerRk) ← (keyA, rankA)
            upper ← columnB
            (upperKey, upperRk) ← (keyB, rankB)
        end if
20:     lowerCost ← SELECT SUM(cost) FROM Columns WHERE clusterKeyId =
    lowerKey
        UPDATE Keys SET rank = lowerRk − lowerCost WHERE keyId = lowerKey
        UPDATE Keys SET rank = upperRk + lowerCost WHERE keyId = upperKey
        UPDATE Columns SET keyId = upperKey WHERE columnId = lower
        UPDATE Columns SET clusterKeyId = upperKey WHERE clusterKeyId =
    lowerKey
25:     if lowerRk − lowerCost = 0 then
            DELETE FROM Keys WHERE keyId = lowerKey
        end if
        if GETKEY(upper) = GETCLUSTERKEY(upper) then
            return lower
30:     end if
        UPDATE Columns SET keyId = upperKey WHERE columnId = upper
        return (lower, upper)
    end function
```

The main change in the column selection algorithm (cf. Algorithm 4) is that now it may return two columns which shall be re-encrypted. By keeping track of the cluster a column belongs to we can *without* degrading the security re-encrypt both columns of a join to the cluster key of higher rank (which they will eventually have anyway). This means there need not be a steady column any more.

4.7 Analysis of the Enhanced Algorithm

Consider the function T defined by $T(1) = 0$ and

$$T(n) = T\left(\lfloor \tfrac{n}{2} \rfloor\right) + T\left(\lceil \tfrac{n}{2} \rceil\right) + \lfloor \tfrac{n}{2} \rfloor \qquad \text{for } n = 2, 3, \ldots \tag{1}$$

or, more explicitly,

$$T(2i) = 2T(i) + i \tag{2}$$

$$T(2i+1) = T(i) + T(i+1) + i \quad . \tag{3}$$

Remark. Using these recursions for T one can also easily prove that $T(n) = \sum_{i<n} Q(i)$ where $Q(i)$ is the sum of the digits of i in its binary expansion. We omit the details here because we will not need this representation in the sequel.

Lemma 1. *For all $n \geq 1$ we have $T(n) \leq \frac{n}{2} \log_2 n$ where equality holds iff n is a power of 2.*

Proof. From the equations (2) and (3) we directly derive the claim by induction because $2(\frac{1}{2} i \log_2 i) + i = i\big((\log_2 i) + 1\big) = \frac{1}{2}(2i)\log_2(2i)$ and (using Jensen's inequality applied to the concave function $x \mapsto x \log_2 x$)

$$\tfrac{(i \log_2 i + (i+1)\log_2(i+1))}{2} \leq \tfrac{i+(i+1)}{2} \log_2 \tfrac{i+(i+1)}{2} = \tfrac{2i+1}{2}(\log_2(2i+1) - 1)$$

hence

$$\tfrac{1}{2} i \log_2 i + \tfrac{1}{2}(i+1)\log_2(i+1) + i < \tfrac{1}{2}(2i+1)\log_2(2i+1) \quad .$$

Moreover, it is clear that equality in $T(n) \leq \frac{n}{2} \log_2 n$ holds iff we never have to the recursion (3), i.e. iff n is a power of 2. □

The significance of the function T arises from

Theorem 2. *For any proxy re-encryption algorithm which is admissible in the sense of section 4.6 there is a schedule on n columns that needs at least $T(n)$ re-encryptions, i.e. its asymptotic behavior is $O(n \log n)$.*

Proof. We divide the columns into two sets of sizes $\lfloor \frac{n}{2} \rfloor$ and $\lceil \frac{n}{2} \rceil$. For either of them we have a worst case schedule with at least $T(\lfloor \frac{n}{2} \rfloor)$ resp. $T(\lceil \frac{n}{2} \rceil)$ re-encryptions. After concatenating these two schedules we get a schedule which still ends in two clusters with *different* keys[2] and which can be extendend by another $\lfloor \frac{n}{2} \rfloor$ joins. Hence we end up with a schedule that by equation (1) requires $T(n)$ re-encryptions. □

[2] This is where we use admissibility!

Theorem 3. *Algorithm 4 applied to n columns needs at most $T(n)$ re-encryptions. Hence it has the optimal worst-case behavior.*

The proof of this theorem uses another lemma:

Lemma 2. *For all $n \geq 2$ we have*

$$T(n) = \max_{1 \leq i \leq n/2} \big(T(i) + T(n-i) + i\big) \quad . \tag{4}$$

Proof (Theorem). Let $\widetilde{T}(n)$ be the number of re-encryption for the worst schedule on n columns. To show that in fact $\widetilde{T} = T$ we look at the last step where there are two clusters of sizes i and $n-i$ respectively (for some $i \leq \frac{n}{2}$). Then

$$\widetilde{T}(n) = \max_{1 \leq i \leq n/2} \big(\widetilde{T}(i) + \widetilde{T}(n-i) + i\big) \qquad (\text{for } n \geq 2).$$

because the columns in the cluster of size $n-i$ contribute with $\widetilde{T}(n-i)$ re-encryptions no matter if these are executed before or after the two clusters have been joined. Columns of the cluster of size i contribute with at most $\widetilde{T}(i)$ re-encryptions before the two clusters are joined and with i re-encryptions after the two clusters are joined. Comparing with Lemma 2 shows that $\widetilde{T}$ satisfies the same recursion as T. □

Proof (Lemma). The proof is again by induction: For $1 \leq i < \lfloor \frac{n}{2} \rfloor$ we calculate

$$\begin{aligned}
&T(i) + T(n-i) + i \\
&= T\left(\left\lfloor \tfrac{i}{2} \right\rfloor\right) + T\left(\left\lceil \tfrac{i}{2} \right\rceil\right) + \left\lfloor \tfrac{i}{2} \right\rfloor + T\left(\left\lfloor \tfrac{n-i}{2} \right\rfloor\right) + T\left(\left\lceil \tfrac{n-i}{2} \right\rceil\right) + \left\lfloor \tfrac{n-i}{2} \right\rfloor + i \\
&= T\left(\left\lfloor \tfrac{i}{2} \right\rfloor\right) + T\left(\left\lceil \tfrac{n-i}{2} \right\rceil\right) + \left\lfloor \tfrac{i}{2} \right\rfloor + T\left(\left\lceil \tfrac{i}{2} \right\rceil\right) + T\left(\left\lfloor \tfrac{n-i}{2} \right\rfloor\right) + \left\lfloor \tfrac{n+i}{2} \right\rfloor \\
&= T\left(\left\lfloor \tfrac{i}{2} \right\rfloor\right) + T\left(\left\lceil \tfrac{n-i}{2} \right\rceil\right) + \left\lfloor \tfrac{i}{2} \right\rfloor + T\left(\left\lfloor \tfrac{i+1}{2} \right\rfloor\right) + T\left(\left\lceil \tfrac{n-i-1}{2} \right\rceil\right) + \left\lfloor \tfrac{n+i}{2} \right\rfloor \\
&= T\left(\left\lfloor \tfrac{i}{2} \right\rfloor\right) + T\left(\left\lceil \tfrac{n-i}{2} \right\rceil\right) + \left\lfloor \tfrac{i}{2} \right\rfloor + T\left(\left\lfloor \tfrac{i+1}{2} \right\rfloor\right) + T\left(\left\lceil \tfrac{n-i-1}{2} \right\rceil\right) + \left\lfloor \tfrac{i+1}{2} \right\rfloor \\
&\quad - \left\lfloor \tfrac{i+1}{2} \right\rfloor + \left\lfloor \tfrac{n+i}{2} \right\rfloor \\
&\leq T\left(\left\lfloor \tfrac{i}{2} \right\rfloor + \left\lceil \tfrac{n-i}{2} \right\rceil\right) + T\left(\left\lfloor \tfrac{i+1}{2} \right\rfloor + \left\lceil \tfrac{n-i-1}{2} \right\rceil\right) - \left\lfloor \tfrac{i+1}{2} \right\rfloor + \left\lfloor \tfrac{n+i}{2} \right\rfloor
\end{aligned}$$

Note that in the last step we use the induction hypothesis (and furthermore use that $\lfloor \frac{i+1}{2} \rfloor < \lceil \frac{n-i-1}{2} \rceil$ because $i < \lfloor \frac{n}{2} \rfloor$).

Now $\lfloor \frac{i}{2} \rfloor + \lceil \frac{n-i}{2} \rceil = \lfloor \frac{n}{2} \rfloor$ unless n is odd and i is even (in which case the value is $\lceil \frac{n}{2} \rceil$), and likewise $\lfloor \frac{i+1}{2} \rfloor + \lceil \frac{n-i-1}{2} \rceil = \lfloor \frac{n}{2} \rfloor$ unless n is odd and $i+1$ is even (where again the value is $\lceil \frac{n}{2} \rceil$).

In any case one of the expressions $\lfloor \frac{i}{2} \rfloor + \lceil \frac{n-i}{2} \rceil$ and $\lfloor \frac{i+1}{2} \rfloor + \lceil \frac{n-i-1}{2} \rceil$ equals $\lfloor \frac{n}{2} \rfloor$ and the other equals $\lceil \frac{n}{2} \rceil$. Hence finally

$$T(i) + T(n-i) + i \leq T\left(\left\lfloor \tfrac{n}{2} \right\rfloor\right) + T\left(\left\lceil \tfrac{n}{2} \right\rceil\right) + \left\lfloor \tfrac{n}{2} \right\rfloor$$

because $-\lfloor \frac{i+1}{2} \rfloor + \lfloor \frac{n+i}{2} \rfloor \leq \lfloor \frac{n}{2} \rfloor$. □

4.8 Multiple Simultaneous Joins

In queries like `SELECT a.b, c.d, e.f FROM a, c, e WHERE a.x = c.z AND b.y = c.z` we need to re-encrypt multiple columns. As already stated we simulate this by joining pairs of columns one after the other. Given a set of k columns that need to be compared for the join, we chose a column corresponding to a key with the highest occuring rank. Then we form pairs consisting of this column and all the other columns successively. Since both our algorithms encrypt "towards the higher rank" this ensures that all columns have the same key as the chosen column after $k-1$ executions of the algorithm. The number of proxy re-encryptions thereby is bounded by $k-1$ for the Algorithm 3 and by k for the Algorithm 4 (since each of the columns is re-encrypted at most once).

4.9 Non-uniform Costs

So far we have assumed uniform costs of 1 for each column, but some columns may be easier to re-encrypt than others. Particularly, the re-encryption cost is linear in the number of elements per column. This means, that it is easier to re-encrypt two columns of size 1 and 2, respectively, than one column of size 4.

We now consider element-wise re-encryption costs by incorporating *non-uniform column costs,* e.g. the size of the column, in our algorithms. Simply, initialize the columns with their respective costs in Algorithm 1. This may significantly reduce the overall costs. Following the example above, consider columns A, B and C of respective sizes 2, 1 and 4 and the join schedule

$\textsc{Join}(A, B)$
$\textsc{Join}(C, A)$

Uniform costs may suggest the following re-encryptions: $A \leftarrow B$, $C \leftarrow A$. This results in 6 element re-encryptions. The worst possible performance for any set of re-encryptions. Instead non-uniform costs using column sizes dictate these re-encryptions: $B \leftarrow A$, $A \leftarrow C$. The result are 3 element re-encryptions. Furthermore, the maximum number of element re-encryptions using either of our algorithms in this example is 4. This is also the minimum worst-case cost under any schedule of join operations.

It is therefore important to note that the analysis of minimum worst-case cost of re-encryption of Section 4 in the general case remains intact. We always achieve the best worst-case cost assuming any future schedule of join operations. To see this, view a column with non-uniform cost c as a group of c columns with uniform cost 1 that always operate successively. Let N be the sum of the costs of all columns, then our algorithm incurs costs of at most $O(N \log N)$.

Nevertheless, in some cases of non-uniform costs we may perform too many re-encryptions for a specific schedule resulting in sub-optimal costs, since the future schedule is unknown. Consider columns A, B, C and D of sizes 1, 5, 2 and 3 and the following join schedule

JOIN(A, B): $A \leftarrow B$
JOIN(C, D): $C \leftarrow D$
JOIN(A, C)

In the third join our algorithm dictates $C \leftarrow A$ leading to 5 element re-encryptions. This clearly leads to the minimal costs of also 5 for a future

JOIN(B, C)

but in case there is no future join, costs are not optimal. It would be more efficient to re-encrypt as $A \leftarrow C$ resulting in a cost of 4 element re-encryptions. Yet, choosing to re-encrypt as $A \leftarrow C$ will increase the worst-case cost under many future join schedules. We therefore choose to optimize the worst-case cost where our bound is tight.

The number of elements in a column may vary, because rows may be inserted or deleted. This further complicates the analysis and possible algorithms also have to account for these future operations. We leave this as future work and currently assume fixed non-uniform costs.

5 Experiments

We performed a number of experiments in order to measure the difference between the best re-encryption cost and our Algorithm 3. We chose $n = 8$ columns and a join schedule of length $m = 16$. Note that we need to find the optimum schedule in $2^m = 65536$ options.

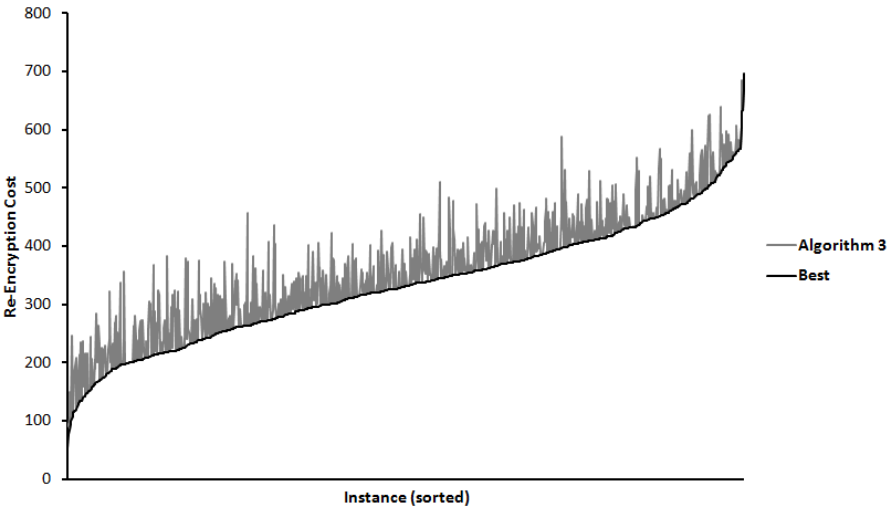


Fig. 1. Cost difference between optimum and Algorithm 3

We chose the schedule uniformly random among all possible pairs of different columns. Pairs could occur repeatedly. We chose the cost for each column uniformly random between 1 and 100.

We performed 1000 experiments. In each we recorded the optimum cost and the cost of our Algorithm 3. In 41% of the experiments our algorithm delivered the optimum schedule. The mean difference to the optimum was 26, i.e. roughly half an average column cost. The maximum difference was 193 and the median difference was 9, such that few large sub-optimal cases account for the majority of the difference.

In Figure 1 we depict our results. We have sorted all experimental results in increasing optimal cost (black line). The gray line depicts the corresponding cost of our algorithm.

6 Conclusion

In this paper we have considered the problem of selecting a column for re-encryption in a deterministically, adjustably encrypted database. To the best of our knowledge this is the first paper considering this problem. We have provided an algorithm that achieves the best possible worst-case bound and a simpler algorithm that performs very well in experimental settings.

References

1. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)
2. Binnig, C., Hildenbrand, S., Färber, F.: Dictionary-based order-preserving string compression for main memory column stores. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD) (2009)
3. Blaze, M., Bleumer, G., Strauss, M.J.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)
4. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 224–241. Springer, Heidelberg (2009)
5. James, G.: The representation theory of the symmetric groups. LNM 682. Springer (1978)
6. Galler, B., Fischer, M.: An improved equivalence algorithm. Communications of the ACM 7(5) (1964)
7. Hacigümüs, H., Iyer, B., Li, C., Mehrotra, S.: Executing SQL over encrypted data in the database-service-provider model. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD) (2002)
8. Pohlig, S., Hellman, M.: An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. IEEE Transactions on Information Theory 24 (1978)
9. Popa, R., Redfield, C., Zeldovich, N., Balakrishnan, H.: CryptDB: Protecting confidentiality with encrypted query processing. In: Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP) (2011)
10. Popa, R., Zeldovich, N.: Cryptographic treatment of CryptDB's adjustable join. Technical Report MIT-CSAIL-TR-2012-006 (2012)

Access Control and Query Verification for Untrusted Databases

Rohit Jain and Sunil Prabhakar

Department of Computer Sciences, Purdue University
West Lafayette, IN, USA
{jain29,sunil}@cs.purdue.edu

Abstract. With the advent of Cloud Computing, data are increasingly being stored and processed by untrusted third-party servers on the Internet. Since the data owner lacks direct control over the hardware and the software running at the server, there is a need to ensure that the data are not read or modified by unauthorized entities. Even though a simple encryption of the data before transferring it to the server ensures that only authorized entities who have the private key can access the data, it has many drawbacks. Encryption alone does not ensure that the retrieved query results are trustworthy (*e.g.*, retrieved values are the latest values and not stale). A simple encryption can not enforce access control policies where each entity has access rights to only a certain part of the database. In this paper, we provide a solution to enforce access control policies while ensuring the trustworthiness of the data. Our solution ensures that a particular data item is read and modified by only those entities who have been authorized by the data owner to access that data item. It provides privacy against malicious entities that somehow get access to the data stored at the server. Our solutions allow easy change in access control policies under the lazy revocation model under which a user's access to a subset of the data can be revoked so that the user can not read any new values in that subset of the data. Our solution also provides correctness and completeness verification of query results in the presence of access control policies. We implement our solution in a prototype system built on top of Oracle with no modifications to the database internals. We also provide an empirical evaluation of the proposed solutions and establish their feasibility.

Keywords: Access Control, Cloud Computing, Query Verification, Private Outsourcing.

1 Introduction

Access control mechanisms are an important part of a database system with which the data owner limits a user's access to a subset of the data. In a typical setting, the database server enforces access control policies by rewriting user queries to limit access to the authorized subset. When the data owner wants to revoke or grant a user, access to a certain part of the data, the data owner does that by informing the server. Traditionally, the server is assumed to be trustworthy and the data owner assumes that the access control policies will be faithfully enforced by the server. However, this assumption is not reasonable when the database is hosted at a third-party server, *e.g.*, cloud, as the

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 211–225, 2013.

data owner lacks control over the hardware and software running at the server. Even when the server is trusted, there is a threat from a malicious insider or an intruder.

Another important problem that arises when the database systems are hosted at an untrusted server is to verify the trustworthiness of query execution. Much work has been done [1,2,3,4] towards verifying correctness and completeness of query results. However, most of these solutions do not work in the presence of access control rules, as they leak information that is outside the query range and outside the scope of the user's authorization.

In this paper, we provide solutions that ensure that a data item in the database is read and modified only by authorized users, and none other (including the server). The data encrypted by our solution is still queriable. Our solution provides mechanisms to verify the trustworthiness of query results in the presence of access control rules. For this, we extend our previous work [1] on ensuring the trustworthiness of data retrieved from an untrusted database that can be modified by multiple entities. The contributions of this work are:

- A novel mechanism to enforce access control rules without trusting the server
- Solutions that allow users to verify the correctness and completeness of query results in the presence of access control rules
- A demonstration of the feasibility of the solution through a *prototype in Oracle*, and its evaluation

The rest of this paper is organized as follows. Section 5 discusses some related work. Section 2 describes our model and presents some preliminary tools that are necessary for this work. Section 3 presents our solutions. A discussion of the implementation of the solution and an empirical evaluation is presented in Section 4. Finally, Section 6 concludes the paper.

2 Preliminaries

In this section, we start by explaining the different entities involved in our model. Then, we explain Merkle Hash Trees and Merkle B+ Tree which we use for building our solutions, and also discuss their use to verify the correctness and completeness of query results.

2.1 Model

There are three main entities involved: **Alice**, the database *owner*; **Bob**, the *(untrusted) database server* that will host the database; and **Carol**, the user(s) that will access this data (may include Alice) from the server. Users are authorized by Alice and can independently authenticate themselves with the server. A user can read or write data to the parts of the database she is authorized to. Figure 1 shows the various entities in this model.

Alice wants to ensure that the data are accessed by only those entities that were authorized by her. Alice and Carol want to ensure that the query results were indeed correct and complete in presence of access control policies.

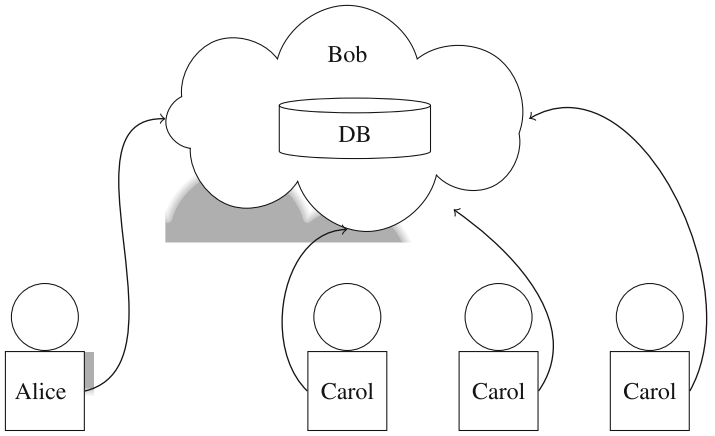


Fig. 1. The various entities involved: The database owner (Alice); The database server(Bob); and Authorized users (Carol)

An acceptable solution should allow Alice to grant or revoke access to a user at any point in time, without much work. The solution will disable Bob from being able to read the encrypted data. However, Alice and Carol should still be able to execute queries and run updates on the encrypted data.

Note that our assumptions about Bob are minimal. In most settings, the server is likely to be at least semi-honest – *i.e.*, it will not maliciously compromise data privacy by not following access control rules, or compromize data integrity by maliciously modifying the data or query results. However, due to poor implementation, failures, over commitment of resources, or other reasons, some loss of data or breach of privacy may occur. Given the lack of direct control over the server, Alice should not assume that Bob is infallible.

Lazy Revocation Model: As mentioned before, simple encryption can ensure that only authorized users can read or write the data. However, this introduces many problems. One such problem is related to dynamic access control rules. In a simple encryption method, when a user's access is revoked from a subset of the data, the data have to be re-encrypted. This can be a very costly process due to network usage and computation for encryption. To alleviate this burden, we consider the *Lazy Revocation Model*. Under this model, when a user is granted access to a subset of the database, the user can read or write to that subset. If the user's access is revoked from that subset, the data are not re-encrypted immediately. Instead, the new values in that subset are encrypted with a new version of the key so that the evicted user can no longer read the new values in that subset. Since the user had access to the old data before eviction, it can be assumed that the user had cached that data, hence it is not important to re-encrypt old values. We will consider the lazy revocation model for access control policies.

Table 1. Symbol Table

Symbol	Description
t_i	the i^{th} tuple of a relation
$h(x)$	the value of a one way hash function over x
$\Phi(n)$	label of node n in MB-tree or MHT
H_i	label of the i^{th} node in the MB-tree
$a\|\|b$	concatenation of a and b
VO	a verification object
$Proof$	root label of the MB-tree
R	a set of ranges that partitions the data
r_i	$r_i \in R$
S_i, K_i	State and key for range r_i
$Enc_k(x)$	encryption of x using symmetric-key k
B_n	access control bitmap for node n

2.2 Correctness and Completeness

We begin by discussing the use of Merkle Hash Trees (MHT) to prove correctness. And then further discuss a variant, the MB-tree, which we use to prove completeness. We will use an MB-tree as a building block for our overall solution. Correctness requires that any data item in the query result are indeed part of the database and is not a fabricated value. An MHT can be used to establish the correctness of query results. An MHT is a binary tree with labeled nodes. We represent the label for node n as $\Phi(n)$. For an internal node, n, with children n_{left} and n_{right}, $\Phi(n)$ is defined as:

$$\Phi(n) = h(\Phi(n_{left})||\Phi(n_{right})) \tag{1}$$

where $||$ is concatenation and h is a one-way hash function. Table 1 explains the symbols used in this paper. Labels for leaf nodes are computed as the hash of the tuple value represented by that leaf. The root label is called 'Proof'.

Initially, an MHT is created on top of the database table. Alice stores only the root hash value ($Proof$) to authenticate future query results. To prove the correctness of a tuple, *i.e.*, to verify that a tuple existed in the database, Alice can ask Bob for some extra data (called *Verification Object (VO)*) from the MHT and recompute the root hash label. If the computed root hash label is the same as that she stored initially, she is convinced about the correctness of the tuple.

Completeness requires that all data items that should have been part of the query result are indeed present in the query result. Correctness and completeness combined establish the correntness of read-only queries. MHT can be extended to use B+ trees instead of a binary tree [2]. MB-trees can be used to verify both correctness and completeness. To prove completeness of a range query, Bob provides extra data with which Alice can verify that tuple values just preceeding, and just following (in sorted order) the query results were indeed outside the query range. Alice can also verify that no data is missing from the query result and the returned values are indeed part of the database. For more details, please refer to [1,2]. Figure 2 shows a sample MB-tree structure built on the attribute A of Table 2.

Table 2. Sample Data Table

tupleID	A
1	23
2	29
3	35
4	48
5	59
6	63
7	65
8	70

Table 3. Bucketized Data Table

tupleID	A^*	Enc(A)
1	[20-30)	$Enc_{K_1}(23)$
2	[20-30)	$Enc_{K_1}(29)$
3	[30-40)	$Enc_{K_1}(35)$
4	[40-50)	$Enc_{K_2}(48)$
5	[50-60)	$Enc_{K_2}(59)$
6	[60-70)	$Enc_{K_2}(63)$
7	[60-70)	$Enc_{K_3}(65)$
8	[70-80)	$Enc_{K_3}(70)$

As an example, consider a query $\sigma_{30<A<60}$. The result for this query would include t_3, t_4, and t_5. To verify the correctness and completeness of the query results, the server sends *VO* to the user which includes the tuples just preceeding and just following the query ranges (*i.e.*, t_2 and t_6). The *VO* also includes any node labels that are required to compute the root label (*i.e.*, $h(t_1)$ and H_7). Using *VO*, the user can generate the $Proof$. If the computed proof matches with the proof value computed by Alice, the user is assured that the query results were correct and complete.

Access Control: In the presence of access control rules, traditionally, the query range is divided into multiple parts to ensure that each sub range is accessible to the user. In that case, each sub range can be verified individually. However, for verification, the server has to reveal the tuples bordering each sub-range. These bordering tuples may not be accessible to the user, leading to information leakage. In the next section, we will discuss our proposed solutions to enforce access control rules while still allowing query verification and privacy from the server or hackers.

Updates: MHT or MB-tree work for static databases. When the data can be modified by users without prior knowledge of the data owner, as is the case in our model, MB-tree cannot be used directly. [1] proposes solutions with which authorized users can executed transactions at the server without being vetted by the data owner. These transactions can read or write data. This is done by engaging the server in a protocol that requires

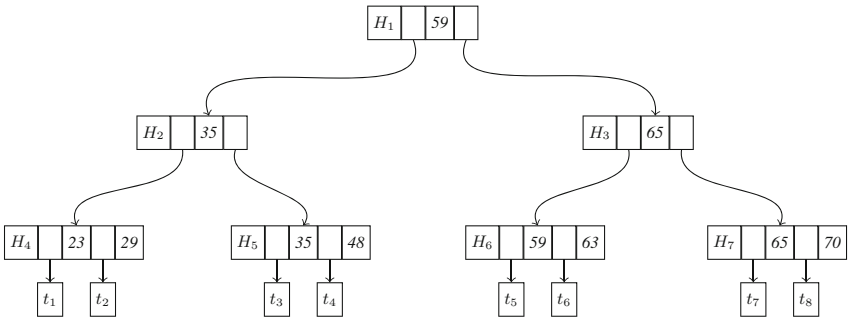


Fig. 2. An MB-tree on attribute A of Table 2

the server to declare the database state on which the transaction was executed and the database state that the transaction produced. The user can verify that the transaction read values from the declared consistent state to produce the next consistent state. However, this solution does not enforce access control rules. In the next section, we provide a solution that allows the user run and verify transactions in the presence of access control rules.

3 Access Control

As mentioned before, access control rules allow the data owner to restrict a user's access to a certain part of the database. The database owner may also want to hide the data from the server as well, while still allowing the users to read and query the data. The difficulty introduced by using access control rules is two fold. Firstly, verification algorithms have to be modified to assure the user that the partial database table visible to the user is indeed correct and complete. Secondly, the data have to be encrypted so the user sees only the allowed data, and, the data remain private from the server or an intruder. The server should still be able to run queries on this data. In this section, we provide our solutions to these problems.

In this paper, we consider fine-grained access control policies. We assume that the access control policies expose a user to a subset of each database table (this is the approach adopted by some commercial systems like Oracle VPD). In particular, we consider the following system for defining access control rules: $R = \{r_i | 0 \leq i \leq k\}$ is a set of ranges on an attribute that partitions the data into k disjoint subsets. Each user is allowed access (read and write) to a part of the database table defined by a subset of R, *i.e.*, the user, Carol, can access tuples $\{t_i | t_i \in \cup r_{i_l}\}$, where $\{r_{i_l}\} \subset R$ is the set of ranges accessible to Carol.

3.1 Verification in Presence of Access Control

Given a range query, all tuples that satisfy the range query may not be accessible to the user. In such case, the verification using the regular MB-tree *VO* will not work. Also, verification of query results usually involves reading extra tuple values [2,4]. These tuples may not be accessible to the verifier due to the access control rules. In such case, the verifier will not be able to verify a query or a transaction. Suitable adjustments to the authentication data structures are required to enable the verification of a query in presence of access control rules.

To solve this problem, we modify the MB-tree as follows: Each node, n, is extended with an access control bitmap, B_n, in which the i^{th} bit is "on" if there is a tuple in the subtree that belongs to the range r_i. Node labels are computed using Equation 2.

$$\Phi(n) = h(B_{child\_1} || \Phi(n_{child\_1}) || \ldots || B_{child\_k} || \Phi(n_{child\_k})) \quad (2)$$

The *VO* now contains the nearest tuple value just preceding and the nearest tuple value just following the query range such that these tuples are accessible to the user. *VO* also contains all the tree nodes required to prove the correctness of these tuples and to prove

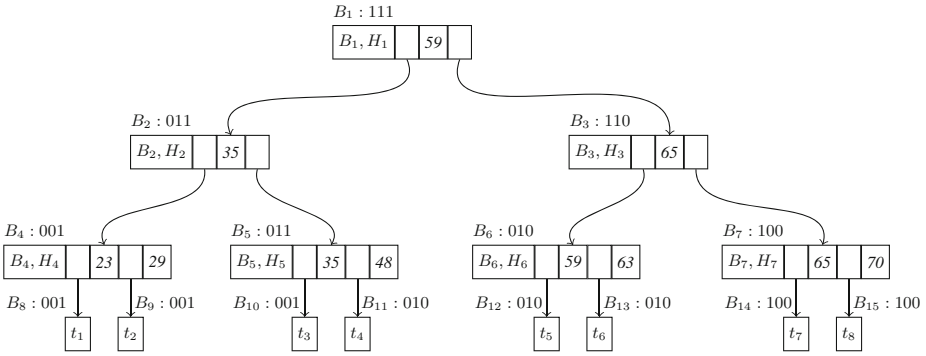


Fig. 3. Augmented MB-tree to allow Access Control

that the tuples that were left out of the query results were indeed inaccessible to the user.

As an example, consider the following access control ranges on attribute A:

r_1 : [0, 35]
r_2 : [36, 64]
r_3 : [65, 100]

Under these accesss control ranges, each access control bitmap will have three bits, one for each access control range. Figure 3 shows an augmented MB-tree, as described above, built on Table 2.

When a user who is authorized to access r_1 and r_3 executes a range query $\sigma_{25<A<50}$, tuple t_2 and t_3 will form the query result. Tuple t_4 will not be part of the query result as it is not accessible to the user. To verify the correctness and completeness of the query result, the user has to verify that the missing tuples were indeed inaccessible to her. The user will also have to verify that the nearest tuple just before the query range was indeed t_1 and the nearest tuple just following the query range (and also accessible to the user) is indeed t_7.

To prove the completeness of the query result, the *VO* of this query will include t_1 and t_7. To prove that the ommitted tuples (*i.e.*, t_4, t_5, and t_6) were indeed inaccessible to the user, the *VO* will also include the bitmaps B_6 and B_{11}. Using B_6, the user can be convinced that tuples t_5, and, t_6 were indeed inaccessible to her. Similarly, using B_{11} the user can be assured that Tuple t_4 was inaccessible to her. As in the case of regular MB-tree, the *VO* will include all other necessary labels required to calculate the root label.

3.2 Enforcing Privacy for Access Control

In this subsection, we present our solutions to encrypt the database, so that a user can read/write only the subset of the data that she has been authorized to access. The server (or any intruder) cannot read the data. As mentioned before, in this work we consider

the *lazy revocation model*. Under this model, once a range, r_i, is removed from a user's accessible ranges, the future tuples in r_i are encrypted using a new key. All remaining and future users who can access r_i will be distributed the new key. Any pre-existing tuples in r_i are not necessarily re-encrypted with the new key. To decrypt the data in the range r_i, the user may need the current or previous keys of that range. Only the data owner decides which ranges are accessibe to the user.

The Key Regression scheme [5] provides a mechanism for versioning encryption keys for symmetric-key encryption. Given a version of the key, the user can compute all previous versions of the key. However, future versions of the key can not be derived using the current key. In the start, all data items in an access control range are encrypted using the first version of the key. Each user authorized to access the range is given that key. When a user is evicted from the range, the key is updated to a newer version. All future data items in the range are now encrypted using the new version of the key. Since the users cannot generate the new version of the key, the evicted user cannot read future tuples in the range. A Key Regression scheme is defined using four algorithms. Algorithm *setup* is used by the data owner to setup the initial state. Algorithm *wind* is used to generate the next state. Algorithm *unwind* is used to derive the previous state, and *keyder* is used to generate the symmetric key for a given state. The tuples are encrypted using the symmetric key. We consider a particular key regression scheme that uses RSA to generate states.

Consider an RSA scheme with private key $< p, q, d >$, public key $< N, e >$, and security parameter k, such that p and q are two k-bit prime numbers, $N = pq$, and $ed = 1(mod\varphi(N)$ where $\varphi(N) = (p-1)(q-1)$. For each range r_i, a secrent random number $S_i \in Z_N^*$ is selected as the initial state.

Algorithm *wind*, *unwind* and *keyder* are defined in Algorithms 1, 2, and, 3 respectively.

Algorithm 1. wind (N, e, d, S_i)

$nextS_i = S_i^d \ (mod\ N)$
return $nextS_i$

Algorithm 2. unwind (N, e, S_i)

$prevS_i = S_i^e \ (mod\ N)$
return $prevS_i$

Algorithm 3. keyder(S_i)

$K_i = SHA1(S_i)$
return K_i

For each range, r_i, in range set R, the data owner generates a secret state $S_i \in Z_N^*$. The user stores the current states for each range it has access to. Using the current state of a range, the user can compute the corresponding symmetric-key to encrypt or decrypt the data in that range. Whenever, a range is added or removed from a user's accessible

ranges, the state corresponding to that range is moved to the next state and all users who still have access to that range are informed about the new version of the state. If a tuple in a range is encrypted using a newer key, the user requests the new state from the data owner.

Due to encryption, the server cannot execute range queries. To be able to execute range queries on data, we use bucketization to divide the data into multiple buckets. Range queries are then suitably modified to search data among these bucket ranges.

Bucketization: Bucketization involves partitioning the attribute domain into multiple equi-width or equi-depth partitions. Attribute values are then converted from a specific value in the domain to the bucket labels. Table 3 is an example of equi-width bucketization of Table 2 where each partition width is 10. Using equi-width bucketization reveals the density in each bucket. Equi-depth, on the other hand, requires frequent adjustments (which requires communication with the user) when database is updated frequently. [6,7] show that only limited information can be deduced due to bucketization.

As shown in Table 3, the attribute A^* represents the bucket labels after bucketization, and the encrypted tuple value is kept in a separate attribute. User queries are now executed on A^*. To verify the correctness and completeness of query results, our augmented MB-tree can be built on top of the bucketized field, A^*. The verification process will remain the same, except now tuples will be inserted in the tree according to A^*.

Thus, combining the solutions proposed in Subsections 3.1 and 3.2, the data owner and the users can be convinced that the data were not maliciously modified, and the data were accessed by the user that had appropriate authorizations.

4 Experiments

To demonstrate the feasibility and evaluate the efficiency of the proposed solutions, we implement our solutions on top of Oracle. The solutions are implemented in the form of database procedures using Pl/SQL and no internal modifications were done on the database. While we expect that the ability to modify the database internals or to exploit the index system will lead to a much more efficient implementation, our current goal is to establish the feasibility of our approach and to demonstrate the ease with which our solution can be adopted for any generic DBMS. Users are implemented using Python.

Setup: We create a synthetic database with one table *uTable* containing one million tuples of application data. *uTable* is composed of a table with two attributes ($TupleID$ and A). The table is populated with random values of A between -10^7 and 10^7. When tuples are encrypted, the ciphertext is stored in attribute $EncA$. Table 4 describes the different tables and indexes used in our prototype. An MB-tree is created on attribute A (integer). We consider three transactions implemented as stored procedures, namely *Insert*, *Delete*, and *Select*. *Insert* creates a new tuple with a given value of attribute A. *Delete* deletes the tuples which have a given value of attribute *A* and *Select* is a range query over attribute A. The experiments were run on an Intel Xeon 2.4GHz machine with 12GB RAM and a 7200RPM disk with a transfer rate of 3Gb/s, running Oracle 11g on Linux. We run Oracle with a strict serializable isolation level. We use a standard block size of 8KB.

Table 4. Relations and Indexes in the database

Table	Attributes	Indexes
uTable	TupleID, A, EncA[2], keyVersion[2]	A
uTableMBT	id, level, Label, keys, children, childLabels, key_min, key_max, accessBitmap[1], childAccessBitmap[1]	id, (key_min, key_max, level)
AccessControlRanges	id, key_min, key_max	
AccessControlRules	AcessControlRule_id, Use_id	

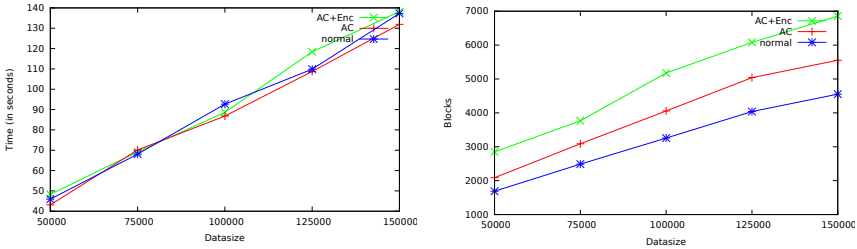
Implementation Details: The MB-tree has been implemented in the form of a database table – each node in the MB-tree is represented by a tuple in the MB-tree table (*uTableMBT*). Ideally, the MB-tree should be maintained as a B+ index trees of the database. However, that requires internal modifications to the index system of the database. We leave that for future work. Each MB-tree node, identified by a unique id, stores *uTable* tuples in the range $[key\_min, key\_max)$. $level$ denotes the height of the node from the leaf level, *i.e.*, leaf nodes have $level = 0$, and the root has the highest level. The $keys$ field stores the keys of the node, and the $children$ and $childLabels$ fields store the corresponding child ids and labels respectively. $Label$ stores the label of the node. When access control mechanisms are in place, two more attributes, $accessBitmap$ and $childAcessBitmaps$, are added to store the access control bitmap of the node and access control bitmaps of the child nodes respectively.

4.1 Results

We now present the results of our experiments. To provide a base case for comparison, we compare the performance of our solutions with a regular MB-tree based solution [2], where access control rules are not supported. This solution leaks information for transaction verification. Furthermore, this solution does not provide privacy against a malicious server. We analyze the costs of construction for the authentication data structures, execution of a transaction, and verification of a transaction.

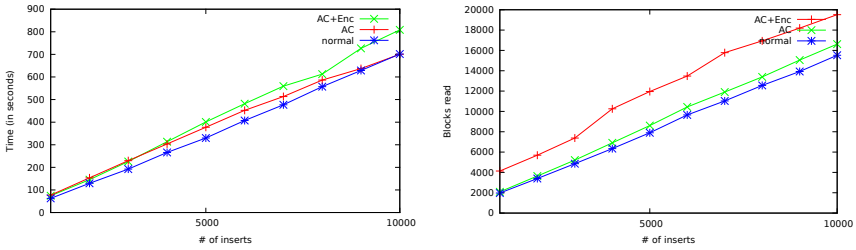
The fanout for the authentication structure is chosen so as to ensure that each tree node is contained within a single disk block. In each experiment, time is measured in seconds, storage and IO is measured as the number of blocks read or written as reported by Oracle.The reported times and IO are the total time and IO for the entire workload. Each experiment was executed 3 times to reduce the error – average values are reported. In the plots, $Normal$ represents the solution from [2], AC represents our solution where access control bitmaps are added to the nodes to support access control rules, and $AC + Enc$ represents our solution that encrypts the tuple values and uses bucketization. AC and $AC + Enc$ both allow query verification in the presence of access control rules. $AC + Enc$ also provides privacy against the server or an intruder.

[1] Used when supporting Access Controls.
[2] Used when supporting Access Controls with Encryption.



(a) Time vs data size (b) Blocks used vs data size

Fig. 4. Construction time and storage overhead



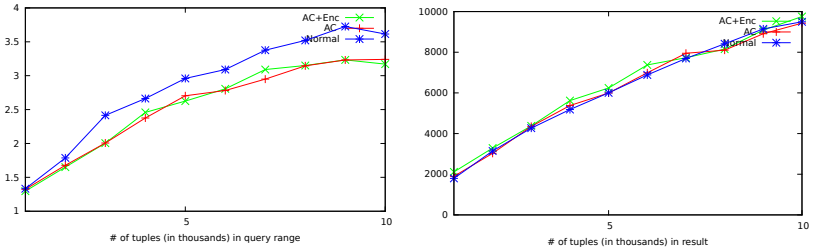
(a) Time vs # of inserts (b) I/O vs # of inserts

Fig. 5. Insert time and I/O overhead

When bucketization is used, we divide the data into 1000 buckets. We use 200 access control ranges.

Construction Cost: First, we consider the overhead of constructing (bulk loading) the proposed data structures. To support access control rules, our solution requires augmenting MB-tree nodes with additional values that store the access control bitmaps. To provide privacy from the server, key regression is used that allows different versions of the encryption key. This requires storing additional attibutes to store the ciphertext and the key version. Figures 4(a) and 4(b) show the effect of data size on construction time and storage overhead, respectively. As expected, the storage cost is higher for our solutions. However, the construction time does not change significantly as the additional computation required for encryption is done by the user, keeping the computation cost for the server similar to just maintaining the MB-tree.

Insert Cost: We study the performance as the number of *Insert* transactions is increased. For this experiment no verification is performed. Figures 5(a) and 5(b) show the results. As expected, our solution incurs a higher overhead for IO as it requires keeping additional data. These costs increase linearly with the number of transactions. Surprisingly, this does not translate into a significant increase in the running time. This represents the computational overhead of hashing and concatenations which dominates the cost. Delete operation shows similar costs (not presented due to lack of space).



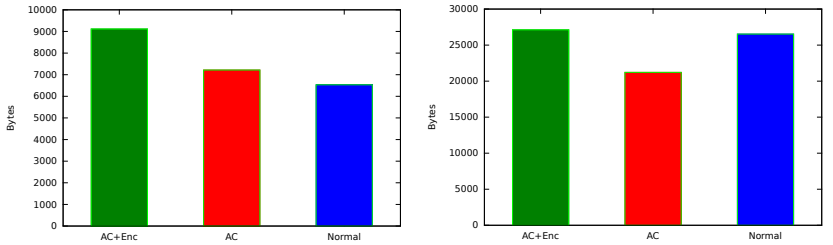
(a) Time vs # of tuples (b) IO vs # of tuples

Fig. 6. Cost of search+verification vs number of tuples in the result

Search Cost: Search cost is influenced by both the size of the result (larger results will be more expensive to verify) and number of access control rules as that requires verifying that the tuples that were dropped from the query result were indeed not accessible to the user. To evaluate the performance of our solution for range queries (*Search*), we run 100 *Search* transactions for different ranges (thereby with different result set size) and verify all transactions. Figures 6(a) and 6(b) show the results. As the result set size increases, execution time and the amount of IO increase. For the regular MB-tree solution, the query range is divided into multiple sub-ranges based on access control rules. Each sub-range that is accessible to the user is returned as query result. For verification, the server has to return the right and left most paths of each sub-range. However, in our solution, an access control bitmap is enough to verify that the sub-range is not accessible. This decreases the *VO* size and computation cost. As shown in the figure 6(a), our solution performs slightly better than MB-tree as our solution requires lesser *VO* size. As the result set size increases, the verification object size increases which results in an increase in verification time. The performance of our solution is comparable to that of an MB-tree alone.

Verification Cost: Our solution changes the *Verification Object* significant as our solution does not require bordering tuples outside the accessible range. However, since the node labels now include access control bitmaps, it increases the *VO* size. We now demonstrate the change in *VO* size in our solutions. To demonstrate the overhead of insert verification, we run 1000 *Insert* transactions and verify them. Average *VO* size is reported in figure 7(a). As expected, the *VO* size is higher for our solutions as it requires additional information, like access control bitmaps and key versions.

To demonstrate the overhead of search query verification on the system, we run 1000 *Search* transactions with varying ranges and varying access control ranges. The average *VO* size is reported in Figure 7(b). As discussed before, in a normal MB-tree, to support access control, a query range has to be divided into multiple sub-ranges so that the query accesses only the part of the data that are accessible to the user. For each sub-range, the *VO* includes the tuple just before and just following the sub-range. *VO* also includes all necessary nodes that are required to verify that the bordering tuples indeed existed the database. However, in our solution, this is not necessary. Each node contains information if the descendant tuples are accessible or not. Hence, *VO* does not always require the bordering tuples. Figure 7(b), that shows the effects of our solution on the



(a) VO Size for insert (b) VO Size for search

Fig. 7. Verification Object Size

VO size validates this. *VO* size for AC is smaller than the normal MB-tree. *VO* size for $AC + Enc$ is comparable to MB-tree. This is due to the ciphertexts.

Overall, we observe that our solutions are efficient and provide mechanisms for access control with reasonable overheads and perform better than current solutions in some cases.

5 Related Work

Much work has been done towards providing mechanisms to verify the correctness and completeness of query results from an untrusted database server [1,2,3,4,8,9]. While some of the earlier work only considered correctness of query results [8,10], later work consider both correctness and completeness [2,4]. Some of these work have also considered data updates from multiple sources [1,4]. [1] proposes a solution that uses Merkle B-Trees as authentication data structure, and allows multiple entities to independently run transactions on the untrusted database. Most of these works do not consider issues related to data privacy and access control. In these works, the user requires additional data items for verification, leading to information leakage.

Some work has been done towards providing verification for correctness and completeness in the presence of access control rules [11,12,13,14,15]. While [12] supports one-dimensional range queries and data updates, [11] supports multi-dimensional range queries and does not handle updates. Both these solutions do not provide privacy against the server. [13] provides a tree based solution for verifying correctness of query results without information leakage. However, this solution does not provide mechanisms for verifying completeness. [14,15] focus on the access control problems with data authenticity for XML data. These solutions provide solutions for data privacy against users but not the server or an intruder. The server or any intruder would have full access to the data leading to breach of privacy.

[16] proposes solutions to provide privacy against the server. Data are encrypted before sending it to the server. The data are encrypted in such a way that user queries can still be executed on the encrypted data. However, this solution does not provide access control mechanisms. Much work has been done towards key management [17,18,19,5,20]. [5,20] consider the *lazy revocation model* under which following the

revocation of user membership from a group, the content publisher encrypts future content in that group with a new cryptographic key and the new key is distributed to only current group members. The content publisher does not immediately re-encrypt all pre-existing content since the evicted member could have already cached that content. [5] proposes a key derivation mechanism with which a user can derive old encryption keys using the current keys, however, it does not allow a user to derive future keys. When a user is evicted from the group, all future updates are encrypted using a newer version of the key. This saves a lot of computation and I/O cost whenever access control rules are changed. [17,18,19] propose key management solutions for access hierarchies. [17] proposes a solution to not only restrict a user's access to a subset of the data, but also restricts the user's access to a limited time.

In this paper, we propose solutions to solve both problems collectively – our solutions provide mechanisms to ensure trustworthiness of query results while ensuring that access control policies are enforced, and it also provides mechanisms for encrypting the data that ensures that a data item is accessed (read and/or write) by only those entities that were authorized to access it.

6 Conclusion

In this paper, we considered the problem of implementing access control policies on an untrusted database server, while ensuring that the query results are trustworthy. With our solution, the data owner can be assured that the data will be read by only those users that were authorized by her apriori. Furthermore, the data owner and the users can be assured of the trustworthiness of the query results without violating the access control policies. We demonstrate that the solutions can be implemented over an existing database system (Oracle) without making any changes to the internals of the DBMS. Our results show that the solutions do not incur heavy costs and are comparable to current solutions for query verification (that do not support access control rules). We believe that the efficiency of the solutions can be further improved by modifying the internals and exploiting the index structures to get better disk performance. We plan to explore these issues in future work.

Acknowledgements. We thank Walid Aref for many discussions and valuable comments. The work in this paper is supported by National Science Foundation grants IIS-1017990 and IIS-09168724.

References

1. Jain, R., Prabhakar, S.: Trustworthy data from untrusted databases. In: ICDE (2013)
2. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L.: Dynamic authenticated index structures for outsourced databases. In: SIGMOD (2006)
3. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and integrity in outsourced databases. In: NDSS (2004)
4. Narasimha, M., Tsudik, G.: Authentication of outsourced databases using signature aggregation and chaining. In: Li Lee, M., Tan, K.-L., Wuwongse, V. (eds.) DASFAA 2006. LNCS, vol. 3882, pp. 420–436. Springer, Heidelberg (2006)

5. Fu, K., Kamara, S., Kohno, T.: Key regression: Enabling efficient key distribution for secure distributed storage. In: NDSS (2006)
6. Ceselli, A., Damiani, E., Vimercati, S.D.C.D., Jajodia, S., Paraboschi, S., Samarati, P.: Modeling and assessing inference exposure in encrypted databases. TISSEC 8(1) (February 2005)
7. Hore, B., Mehrotra, S., Tsudik, G.: A privacy-preserving index for range queries. In: VLDB (2004)
8. Devanbu, P.T., Gertz, M., Martel, C., Stubblebine, S.G.: Authentic third-party data publication. In: DBSec (2000)
9. Pang, H., Jain, A., Ramamritham, K., Lee Tan, K.: Verifying completeness of relational query results in data publishing. In: SIGMOD (2005)
10. Pang, H., Tan, K.: Authenticating query results in edge computing. In: ICDE (2004)
11. Chen, H., Ma, X., Hsu, W., Li, N., Wang, Q.: Access control friendly query verification for outsourced data publishing. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 177–191. Springer, Heidelberg (2008)
12. Pang, H., Jain, A., Ramamritham, K., Tan, K.L.: Verifying completeness of relational query results in data publishing. In: SIGMOD (2005)
13. Kundu, A., Bertino, E.: Structural signatures for tree data structures. In: VLDB (2008)
14. Bertino, E., Carminati, B., Ferrari, E., Thuraisingham, B., Gupta, A.: Selective and authentic third-party distribution of xml documents. TKDE 16(10) (October 2004)
15. Miklau, G., Suciu, D.: Controlling access to published data using cryptography. In: VLDB (2003)
16. Hacigümüs, H., Iyer, B.R., Mehrotra, S.: Providing database as a service. In: ICDE (2002)
17. Tzeng, W.G.: A time-bound cryptographic key assignment scheme for access control in a hierarchy. TKDE 14(1) (January 2002)
18. Akl, S.G., Taylor, P.D.: Cryptographic solution to a problem of access control in a hierarchy. TOCS 1(3) (August 1983)
19. Atallah, M.J., Frikken, K.B., Blanton, M.: Dynamic and efficient key management for access hierarchies. In: CCS (2005)
20. Kallahalla, M., Riedel, E., Swaminathan, R., Wang, Q., Fu, K.: Plutus: Scalable secure file sharing on untrusted storage. In: FAST (2003)

Quantitative Security Risk Assessment of Android Permissions and Applications

Yang Wang[1], Jun Zheng[1], Chen Sun[1], and Srinivas Mukkamala[2,3]

[1] Department of Computer Science and Engineering,
New Mexico Institute of Mining and Technology, Socorro, NM 87801
{yangwang,zheng,chen}@nmt.edu
[2] The Institute for Complex Additive Systems Analysis,
New Mexico Institute of Mining and Technology, Socorro, NM 87801
srinivas@cs.nmt.edu
[3] CAaNES, LLC, Albuquerque, NM 87109

Abstract. The booming of the Android platform in recent years has attracted the attention of malware developers. However, the permissions-based model used in Android system to prevent the spread of malware, has shown to be ineffective. In this paper, we propose DroidRisk, a framework for quantitative security risk assessment of both Android permissions and applications (apps) based on permission request patterns from benign apps and malware, which aims to improve the efficiency of Android permission system. Two data sets with 27,274 benign apps from Google Play and 1,260 Android malware samples were used to evaluate the effectiveness of DroidRisk. The results demonstrate that DroidRisk can generate more reliable risk signal for warning the potential malicious activities compared with existing methods. We show that DroidRisk can also be used to alleviate the overprivilege problem and improve the user attention to the risks of Android permissions and apps.

Keywords: Android App, Android Permission, Malware, Risk Assessment, DroidRisk.

1 Introduction

The use of touchscreen based mobile devices like smartphones has seen unprecedented growth in recent years due to their portability and real-time information access. According to a recent study from research firm Ovum [10], smartphones will dominate the mobile phone market with a compound annual growth rate of 24.9% from 2011 to 2017 and 1.7 billion devices are estimated to be shipped by 2017. The dominance of smartphones is largely attributed to the Google Android mobile OS which took 68% of the global market in the second quarter of 2012 [3]. This is mainly due to it being open source and the large collection of mobile applications (apps) in the unrestricted official and third-party Android app markets. In September 2012, the Google's official app store, Google Play (previously known as Android Market [6]), hit total 25 billion app downloads with more than 675,000 apps [7].

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 226–241, 2013.

However, the popularity of Android platform has also attracted the attention of malware developers. It was reported that more than 260,000 Android devices were affected by a mobile virus called `DroidDream` in the official Android Market within 48 hours in 2011 [5]. Felt et al. did a survey of mobile malware on three different mobile platform including Android [20]. They found that the most common malicious activities are collecting user information and sending premium-rate SMS message. Zhou et al. collected 204,040 apps from five different Android markets between May to June in 2011 [27]. 211 malware were found in the collected apps, where 32 from the official Android Market and 179 from the third-party markets.

The security mechanism used in Android platform to prevent the spread of malware is the permission-based model, which protects the access to sensitive privacy data (contact list, emails, phone call logs, location etc.) and system resources (GPS, camera, WiFi etc.). An Android app needs the user's approval of the requested permissions to be installed in the user's device. Felt et al. conducted two usability studies on the effectiveness of Android permission system [21]: one Internet survey of 308 Android users and a laboratory survey of 25 Android users. The studies showed that only 17% of participants paid attention to permissions during app installation, which indicates that the current Android permission system fails to protect most users from malware.

Several recommendations were made in [21] to improve the low attention and comprehension rate of Android permission system such as re-organizing and re-naming categories, description of permissions focusing on risk instead of resources, smaller permission list etc.. However, based on our own experiences of installing Android apps, the main reason causing the ineffectiveness problem is the text-based permission warning interface for app installation, which can be easily ignored by the users [24]. Our solution for this problem is to have a quantitative assessment of the risk levels of Android permissions and apps. With the quantitative risk information, users can easily understand the risk of an app to be malicious and pay more attention to those permissions with high risk levels.

This paper has made the following contributions:

- We propose DroidRisk, a framework for quantitative security risk assessment of Android permissions and apps based on permission request patterns, which follows the U.S. National Institute of Standards and Technology (NIST) guide for IT security risk management [11]. To the best of our knowledge, this is the first attempt to quantitatively assess the risk levels of both Android permissions and apps.
- We evaluate the effectiveness of DroidRisk with two datasets. The benign app dataset has 27,274 popular apps collected from Google Play in July 2012. The malware dataset consists of 1,260 Android malware samples from the Android Malware Genome Project [26]. We show that reliable risk signal can be generated with the quantitative risk levels of apps for warning the potential malicious activities.

- We show that DroidRisk can be applied to alleviate the overprivilege problem.
- We have implemented two applications of DroidRisk to improve the user attention to the risks of Android permissions and apps: a modified App web page in Android market with the quantitative risk information of the app and its requested permissions, and an Android app to evaluate the risk levels of apps installed in local device.

2 Related Work

Due to its importance for Android security and user privacy, the permission system has attracted lots of research interests. There are several studies on how the permissions are used by Android apps. In [13], Barrera et al. did an empirical analysis of the permission-based security models by analyzing 1,100 most popular Android apps using the Self-Organizing Map (SOM) algorithm. They found that among the defined permissions only a small portion of them are actively used by developers. Another finding of their study is that the requested permissions are not strongly correlated with application categories. Felt et al. did a survey of 100 paid and 856 free apps from Android Market in [20]. It was observed that 93% of free and 82% of paid apps request at least one dangerous permission. They also built a tool called Stowaway that can detect whether a compiled Android app requests more permissions than necessary, i.e. overprivileged [18]. Among the apps they investigated, about one-third were actually overprivileged. In [25], Wei et al. studied the permission evolution in the Android ecosystem. One of their key observations is that the set of dangerous-level permissions always outnumbers other permission types in all versions of the Android platform and it is still growing. Frank et al. studied the permission request patterns of Android apps using pattern mining technique [22]. They tried to relate the permission request pattern with the app's reputation which can be served as an indicator of app quality. Although all these works revealed something about the permission request patterns of Android apps, they didn't attempt to identify potential malware.

Recently, the app's permission request pattern has been used to generate risk signal for warning potential malicious activities. Enck et al. proposed a light weight application certification service called Kirin that uses a rule-based strategy to identify suspicious apps based on their requested permissions [16]. However, because the rules were defined manually, they can't adapt to the changing characteristics of current permissions and apps. For example, the 9th rule of Kirin is no longer valid because the permission `SET_PREFERRED_APPLICATION` has been deprecated since Android API level 7. In [27], Zhou et al. proposed a system called DroidRanger to detect malicious apps in official and alternative Android markets. The first component in DroidRanger is *permission-based filtering* which uses some dangerous permissions such as `RECEIVE_SMS` and `SEND_SMS` to find potential malicious apps. It was shown that only 0.66% of apps needed further analysis after the permission-based filtering step. Chia et al. studied permissions systems of Facebook apps, Chrome extensions and Android apps to

find the reliable signals to identify potential harmful and inappropriate apps [15]. They investigated several signals including the adjusted community rating, the availability of the developer's website and the number of apps published by the developer. However, none of those signals was found to be reliable. In [24], Sarma et al. proposed a set of risk signals by examining the permission request patterns from apps in Android Market and the collected malicious apps. The proposed risk signals include rare critical permissions (RCP), rare pairs of critical permissions (RPCP), combination of RCP and RPCP, and category-based RCP (CRCP). The RCP signal is triggered if at least one of the critical permissions is requested by less than certain percentage of the Android Market apps. The RPCP signal is triggered if for a pair of critical permissions, any individual permission occurs more frequent than they occur as a pair. The CRCP signal is the combination of category information with RCP. Though RCP has shown superior performance compared with Kirin in terms of warning and detection rates [24], the users do not have any idea about the risk levels of requested permissions by an app and the app itself as there is no quantitative assessment of the risk levels.

3 Application Dataset

In this section, we describe the benign app and malware datasets collected for our study. We also provide the statistics of requested permissions of these two datasets.

3.1 Data Collection

Benign App Dataset. Among the large number of Android markets available worldwide, Google Play is the largest and most reliable one. An antivirus system, called Google Bouncer, is deployed to detect the malicious apps uploaded by developers [1]. Any malicious app found by Google Bouncer, that may be harmful to users or tries to steal privacy information, will be removed from Google Play. Thus, it is reasonable to assume that the apps from Google Play perform no malicious activities and can be used to construct the benign app dataset.

To collect the information of apps from Google Play, we developed a crawler to automatically extract the name, category and the requested permissions of each app from its corresponding web page. In total we collected the information of $27,274$ popular apps from Google Play in the middle of July, 2012. The collected apps belong to 26 subcategories under "Applications" category and 8 subcategories under "Games" category.

Malware Dataset. The Android malware dataset used in our study is from the Andriod Malware Genome Project [26]. This dataset consists of $1,260$ Android malware samples in 49 families from different markets which has a much larger size compared with the malware dataset of 121 samples used in [24]. 86% of the samples are repackaged versions of normal apps. 36.7% of them leverage

root-level exploits. More than 90% try to remote control the device. 45.3% and 51.1% of them stealthily send short messages or make calls and collect privacy user information, respectively. Unfortunately, there is no category information for this malware dataset.

3.2 Statistics of Requested Permissions

Many malware request more permissions than the need of their claimed functions to perform malicious activities. For example, a game-like malware may request the permission to send short messages, which could result in a financial loss to the user. In this section, we provide the statistics of permissions requested by benign apps and malware, which inspire our work of DroidRisk.

Frequently Requested Permissions. Figure 1 shows the top 20 requested permissions in the benign app and malware datasets. `INTERNET` is the most frequently used permission by both the benign apps and malware. There are many reasons to request permission for internet access: some of the apps need to log in; some are designed to use internet like browsers and email clients; some need to load advertisement etc. As a result, Internet-related permissions, such as `ACCESS_NETWORK_STATE` and `ACCESS_WIFI_STATE`, become very popular. Another set of widely used permissions are location related ones such as `ACCESS_FINE_LOCATION` and `ACCESS_COARSE_LOCATION` for location based services. The most significant differences between benign apps and malware observed from Figure 1 are: malware are more favor of changing the settings and use money-related services such as short message service (SMS). Changing settings, especially changing the network settings, generally is the first step before a malware performs any malicious activity. Sometimes malware even try to kill background processes, which could help them avoid being detected by anti-virus apps. It can be seen that SMS is a popular service among malware as four SMS-realted permissions are much more popular in malware than benign apps. In [26], Zhou and Jiang introduced a family of malware targeting on the financial charging. This kind of malware may stealthily edit and send out SMS to waste the user's money. They may subscribe premium-rate services without the user's consent for profit.

Number of Requested Permissions. Figures 2(a) and 2(b) show the percentages of malware and benign apps requesting certain number of permissions, respectively. As can be seen, malware are likely to request more permissions than benign apps. It is shown that 58.8% and 92.7% of the benign apps request no more than 4 and 11 permissions respectively, while the numbers for malware are 7.5% and 49.3%. Figures 2(c) and 2(d) show the percentages of malware and benign apps requesting certain number of dangerous permissions, respectively. It can be easily seen that malware also request more dangerous permission than benign apps. More than half of the benign apps request 2 or less dangerous permissions and 91.8% of them have no more than 7 dangerous permissions. For

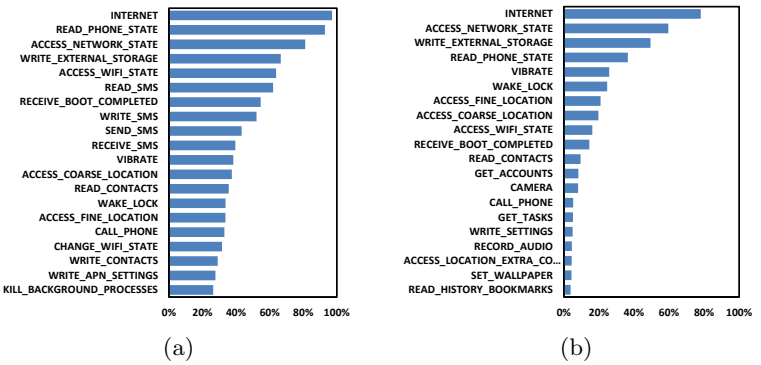


Fig. 1. Top 20 most popular permissions in: (a) malware dataset (b) benign app dataset

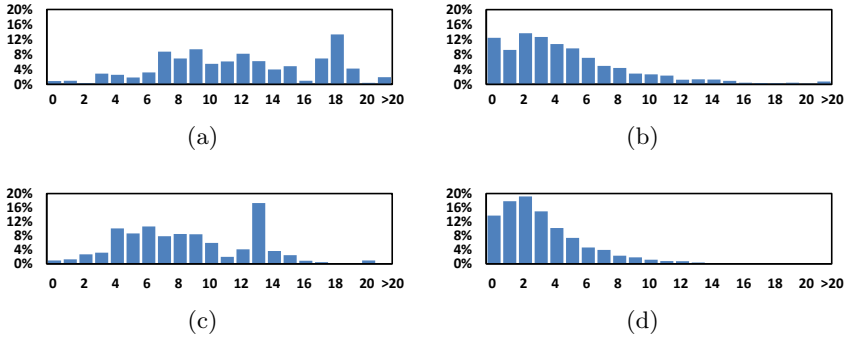


Fig. 2. Percentage of apps requesting certain number of permissions: (a) malware (b) benign apps; Percentage of apps requesting certain number of dangerous permissions: (c) malware, (d) benign apps

malware, 62.5% of them have at least 7 dangerous permissions, and around a quarter of them request 13 or more dangerous permissions. Clearly, malware are more interested in dangerous permissions for their malicious activities.

4 Methods

4.1 Quantitative Security Risk Assessment

Quantitative methods have been used to assess the financial risk for a long time [12] but they are still relatively new for security risk assessment. To guide the risk management for IT systems, NIST published a set of IT security risk management best practices in 2002 [11]. According to the NIST guide, risk assessment is the first step of IT risk management. To quantitatively assess the risk level

of an system, it needs to find the risk of each potential adverse event for the system, which is defined as [11,23]:

$$R(E_i) = L(E_i) \times I(E_i) \tag{1}$$

where E_i is the ith potential adverse event, $R(E_i)$ is the risk of E_i, $L(E_i)$ and $I(E_i)$ are the likelihood and the impact of E_i, respectively. The likelihood represents the probability that a weakness is exploited by the attacker, while the impact refers to the magnitude of harm caused by this weakness being exploited [23]. Assume the adverse events are independent, the system risk R_{sys} can be obtained by summing the risk values of individual adverse events as shown in Equation (2):

$$R_{sys} = \sum_i R(E_i) = \sum_i L(E_i) \times I(E_i) \tag{2}$$

where $i = 1, 2, ..., n$ and n is the total number of potential adverse events.

4.2 DroidRisk – Quantitative Security Risk Assessment of Android Permissions and Apps

The DroidRisk framework for quantitative risk assessment of Android permissions and apps follows the NIST guide. We consider an Android app A as the system and each permission p_i requested by A as the individual adverse event. By assuming that the permissions requested by A are independent, the risk level of the app A, R_A, can be defined as:

$$R_A = \sum_i R(p_i) = \sum_i L(p_i) \times I(p_i) \tag{3}$$

where $R(p_i)$ is the risk level of permission p_i, $L(p_i)$ and $I(p_i)$ are the likelihood and the impact of permission p_i respectively, $i = 1, 2, ..., n$ and n is the total number of requested permissions by A.

The key problem to be solved in the DroidRisk framework is to calculate the likelihood $L(p_i)$ and the impact $I(p_i)$ for a requested permission p_i. We define the likelihood $L(p_i)$ as the probability that the app A is malware if p_i is requested by A, i.e. $P(A \text{ is malware} \mid p_i)$. This *posteriori* conditional probability can be calculated using Bayes' rule as show in Equation (4):

$$P(A \text{ is malware} \mid p_i) = \frac{P(p_i \mid A \text{ is malware}) \times P(A \text{ is malware})}{P(p_i)} \tag{4}$$

where $P(p_i \mid A \text{ is malware})$ is the priori probability that a malware requests permission p_i, $P(A \text{ is malware})$ is the priori probability that an app is malware for all collected apps, and $P(p_i)$ is the priori probability that any collected app, benign or malicious, requests permission p_i.

For estimating impact levels of permissions, we consider two classes of permissions, normal and dangerous[1]. Although it is hard to evaluate the exact level of harm caused by a permission if it is requested by a malware, it is certain that dangerous permissions are more harmful than normal permissions. Thus, we set the impact level of normal permissions, I_{np} as 1 while give dangerous permissions a higher impact level. In Section 5.2, we used an empirical method to determine the impact level of dangerous permissions, I_{dp}.

5 Results

In this section, we first present the likelihood values calculated for Android permissions based on two datasets we collected. We then show how to determine the impact level for dangerous permissions with an empirical method followed by the presenting of risk levels of Android permissions and apps from the two datasets. Finally we demonstrate that DroidRisk can be used to assess the risks of apps in third-party markets.

5.1 Likelihood of Android Permissions

With the collected benign app dataset and malware dataset, we can calculate the likelihood of each Android permission using Equation (4). Figure 3 shows the top 20 permissions with highest likelihood values. The blue bar represents the likelihood of a normal permission while the red bar is for the likelihood of a dangerous permission.

There are 14 dangerous permissions on the list. `WRITE_APN_SETTINGS` has the highest likelihood among all permissions, which can change the network setting, thus to intercept and inspect the network traffic without the user's awareness. There are 5 SMS and MMS related dangerous permissions on the list, which means that SMS and MMS services are the major target of malware developers. Unlike other systems which don't pay attention to normal class permissions, we have 6 of them on the list. These permissions may be used by malware to facilitate some malicious activities. For example, `INSTALL_PACKAGES` and `DELETE_PACKAGES` are used by several malware families such as `JSMSHider` and `GoldDream` to perform update attack and `KILL_BACKGROUND_PROCESSES` is used by an Android Trojan named `AnserverBot` to avoid being detected by certain anti-virus apps [26][2].

[1] Android characterizes the potential risk of the permissions using 4 protection levels – normal, dangerous, signature and signatureOrSystem. Since signature or signatureOrSystem permissions can't be granted to third party apps, we category them into normal class. The protection level for a given permission in this study is obtained from Android API level 9.

[2] `KILL_BACKGROUND_PROCESSES` has been used to replace the old name `RESTART_PACKAGES` for permission - 'kill background processes' since the release of Android API level 8.

5.2 Impact Level of Dangerous Permissions

Based on the discussion in Section 4.2, the impact level of dangerous permissions, I_{dp} should be higher than the impact level of normal permissions, I_{np}. Since there is no way to obtain the actual level of harm caused by normal permissions or dangerous permissions, we solved the problem with an empirical method using ROC (Receiver Operating Characteristic) curve. ROC curve is an efficient tool to evaluate the binary classification performance and select the optimal threshold setting [17]. In our case, once I_{dp} is determined, we can obtain the risk level of each Android permission, and then compute the risk levels of benign apps and malware according to their requested permissions. As one of our design goals is to classify benign apps and malware with a single threshold of risk level, ROC curve is the appropriate choice to find the value of I_{dp} which gives the best classification performance.

ROC curve is plotted with the true positive rate (TPR) versus the false positive rate (FPR) under various threshold settings. Figure 4 shows the ROC curve generated for $I_{dp} = 2$. In Figure 4, the ideal classification performance is obtained at the upper left corner, which means no classification error. The closer the ROC curve to the upper left corner, the better the classification performance. Therefore, a popular metric used to measure the classification performance is the area under curve (AUC) [14], which is the area between the ROC curve and the x-axis. Usually a AUC greater than 0.9 is considered as excellent classification performance.The value of I_{dp} is determined as 1.5 which gives the highest AUC value (0.9313).

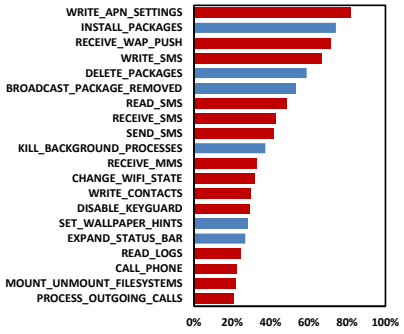


Fig. 3. Top 20 permissions with highest likelihood value

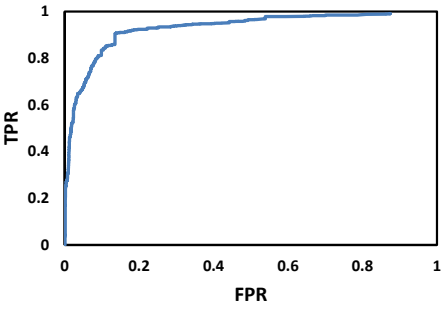


Fig. 4. ROC curve with $I_{dp} = 2$

5.3 Risk Levels of Android Permissions and Apps

Once the impact value of dangerous permissions is obtained, we are able to compute the risk levels of all Android permissions. Figure 5 shows the top 20 permissions with highest risk levels. Compared with the top 20 list for likelihood, there is no new permission on the list but the rankings of most dangerous

permissions are promoted because of higher impact level. `WRITE_APN_SETTINGS` is still on the top of the list with highest risk level among all permissions.

We also compute the risk levels for all benign apps and malware in our datasets by simply summing the risk levels of requested permissions as shown in Equation (3). Figure 6 shows the histograms of the risk levels of benign apps and malware. It can be easily observed that majority of benign apps have risk levels less than 2, while malware typically have higher risk levels.

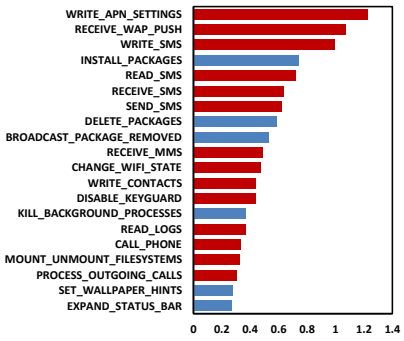


Fig. 5. Top 20 permissions with highest risk levels

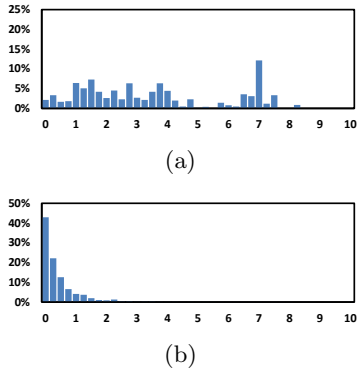


Fig. 6. Risk level histograms, (a) malware, (b) benign apps

As seen in Figure 6(b), there are few benign apps with very high risk levels. We did an investigation on top 20 benign apps with highest risk levels which are shown in Figure 7. The top 6 apps on this list are solely used for in-store demonstration of certain mobile devices. Some of them are warned by the vendors not for public use. The possible reason is that such apps may need lots of permissions to demonstrate different features of the device. It's not surprising to see 6 security apps on this list since they usually require lots of permissions to monitor the whole system and may block, intercept or change the behaviors of the system or other apps. Therefore, it is highly recommended that users should carefully choose the security apps for their devices since there are malware pretending to be anti-virus apps [2]. The rest of the apps on this list are communication tools used for SMS, calls, email exchange etc. Since SMS, calls and other communications may leak privacy information or result in unexpected financial charge, users need to pay special attention to those apps.

5.4 Third Party Android Market

Besides Google Play, there are lots of third party android markets available worldwide, which are considered as the major source of malware [27]. Many of such markets are not capable to do a malware detection before they put a new app on the shelf. Some of them even solely depend on the feedback from

users to decide whether an app needs to be removed from their markets or not. However, third party markets are more preferred than Google Play by many users. One reason is that these users need free apps which cannot be found in Google Play, e.g. free app of the day in Amazon appstore for Android. Another reason is that some users need a localized version of the original app, especially for users whose native language is not English. We arbitrarily selected a third party Android market in China, mumayi [9], to investigate the risk of third party Android markets.

We downloaded 602 popular apps from mumayi using a crawler and collected their requested permission information. We then computed the risk level of each collected app using DroidRisk. Figure 8 shows the boxplots of the risk levels of apps from Google Play, apps from mumayi and malware. The median risk levels for apps from Google Play, apps from mumayi and malware are 0.32, 1.35 and 3.22, respectively. This indicates that to install an app from the third party market is more risky than the official market, Google Play.

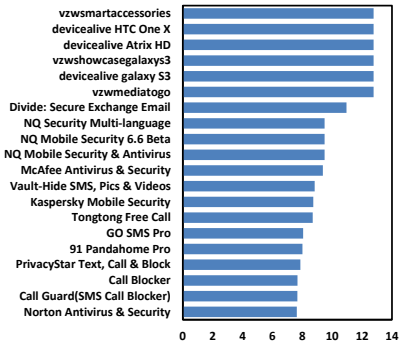


Fig. 7. Top 20 benign apps with highest risk levels

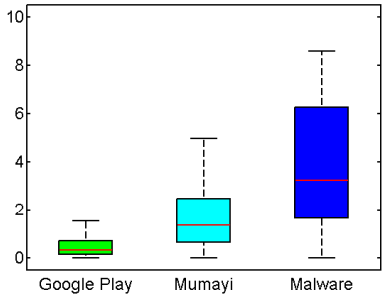


Fig. 8. Boxplots of the risk levels of apps from Google Play, apps from mumayi and malware

6 Applications of DroidRisk

In this section, we show that reliable risk signal to identify potential malware can be generated with the quantitative risk information obtained by DroidRisk. The applications of DriodRisk for alleviating the overprivilege problem and improving the user attention to the risks of Android permissions and apps are also presented.

6.1 Reliable Risk Signal

Risk signals are used by users to identify potential malware. A reliable risk signal should be triggered by as many malware as possible, and rarely be triggered by

benign apps. We compare the performance of DroidRisk with two state-of-art methods: Kirin from [16], RCP and $RCP + RPCP$ from [24] using 10-fold cross-validation. The risk signal for RCP is generated with rule $\#RCP(\theta) \geq 1$ which is the simplest one in [24]. The risk signal for $RCP + RPCP$ is generated with rule $\#RCP(2) + \#RPCP(1) \geq \theta$ which is the best performing one in [24]. The performance of Kirin is obtained with 7 rules as described in [24]. Figure 9 presents the ROC curves of different method for fold-1. For other folds, we obtained similar results. From Figure 9, we can observe that DroidRisk has significant better classification performance than other methods. One possible reason is that DroidRisk tries to capture the request patterns of both normal and dangerous permissions while there are only 9 manually defined rules in Kirin and only 24 or 26 critical permissions are used for RCP and $RCP + RPCP$.

Figure 9 also shows that the risk signal of Kirin is not tunable as the rules are predefined. Although the risk signals generated by RCP and $RCP + RPCP$ are adjustable, it is not easy to find a reliable one since there are several parameters to be tuned. With the quantitative risk information generated by DroidRisk, the risk signal is generated from a single threshold of risk level which can be easily found using ROC curve. The simple thresholding operation also makes it suitable to be implemented in resource-constrained mobile devices.

The results of the 10-fold cross validation are shown in Figure 10, where F-score is used as the performance measure. The parameters of DroidRisk, RCP and $RCP + RPCP$ are chosen as the best performing ones for the training sets. It can be seen that DroidRisk has the best performance among all the methods.

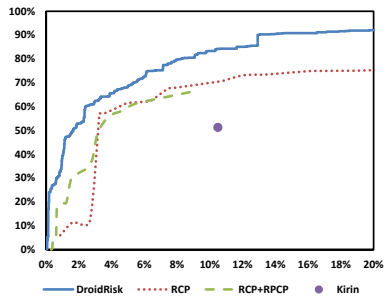


Fig. 9. ROC curves of different methods for fold-1

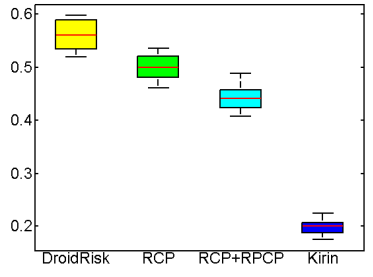


Fig. 10. Boxplots of the F-scores of different methods for 10-fold cross validation

6.2 Overprivilege Problem

Android permission system gives the app developers the ability of requesting any permission no matter they actually use it or not. Many developers and users rarely pay attention to whether the requested permissions are useful for them or not. As a result, Android apps tend to be overprivileged, which may raise security problems. We randomly selected 50 popular apps from both the third

party Android market, mumayi, and Google Play. We used the Stowaway [18] to detect unnecessary permissions in each app. Table 1 shows the statistics of the collected apps due to the overprivilege problem. The boxplots in Figure 11 show how the unnecessary permission affect the risk levels of apps from mumayi and Google play. It can be seen that the overprivilege problem is common in both the official and third party markets although the problem is much severe in third party market. The median risk level of apps from mumayi has increased from 1.04 to 1.43 due to overprivilege. Although the median risk level of apps from Google Play only increases from 0.64 to 0.66, we did find 7 apps whose risk levels raise above the threshold set by DroidRisk because of those unnecessary permissions, which makes them suspicious to be malware. Since the app developers always try to get more downloads, declaring unnecessary permissions to result in a high risk level for the app will make the user more likely to cancel the download because of the warning of our DroidRisk system. Thus, the overprivilege problem can be alleviated since the developers need to be more careful about requesting permissions for their apps, especially those related to possible malicious activities.

Table 1. Statistics of the collected apps from mumayi and Google Play due to overprivilege problem

	mumayi	Google Play
Percentage of overprivileged apps	64%	44%
Average # of unnecessary permissions	1.92	0.84

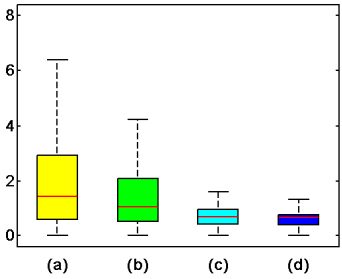


Fig. 11. Risk levels with and without unnecessary permissions for apps from, (a-b): mumayi, (c-d): Google Play

6.3 Improving User Attention to Risks of Android Permissions and Apps

One problem with current Android permission system is that all information related to the risks of permissions are text descriptions while users often do not pay attention to the text information due to various reasons [21]. One of the design goals of DroidRisk is to improve the user attention of the risks of Android permissions and apps. In the following, we show two applications that utilize the quantitative risk information obtained by DroidRisk to reach this design goal.

App Web Page with Quantitative Risk Information: Each Android app has a web page after it is submitted to an Android market, official or third party. Users often browse the market to find out interesting apps for their devices.

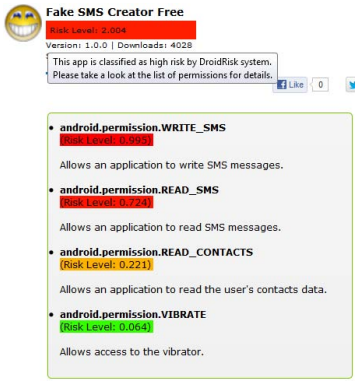


Fig. 12. Screenshot of the Fake SMS Creator Free app's web page in Aptoide with quantitative risk information

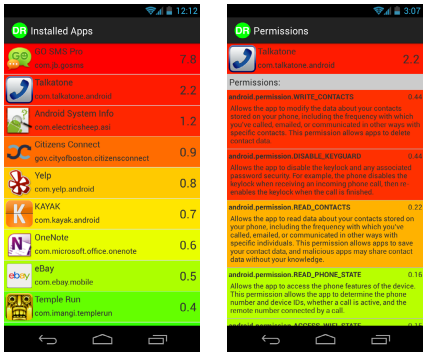


Fig. 13. DroidRisk Android App: (a) view of the list of installed apps with their corresponding risk levels, (b) view of the list of permissions requested by an app and their risk levels

Since the text-based permission information shown in the app's web page is not effective in informing the user about the risk of the app, we developed a solution to present the quantitative risk information of the app and its requested permissions in the web page. The solution was implemented in FireFox browser by creating a GreaseMonkey[3] script to change the display of the app's web page. Figure 12 uses the Fake SMS Creator Free app's web page in Aptoide, a third party market[4], as an example to show how the risk information of the app and its requested permissions are displayed in the web page. The current design uses a colored box that displays the risk value of an app or a requested permission. The color scheme of the design corresponds to the risk level. When the user's mouse moves above the box of the app's risk level, a textbox will be shown to alert the potential risk and ask the user to see the list of permissions. It is expected that this design with the quantitative risk information and the color-based visual cue can significantly improve the attention rate of Android permission system.

DroidRisk Android App: To let the users evaluate the risks of apps installed in their local devices, we developed an Android App of the same name as the framework. This app extracts the permissions requested by each installed app and computes the risk level of the app. Figure 13(a) shows the app's interface which shows the list of installed apps with their corresponding risk levels. The user can tap any app to see the list of permissions requested by the app and their risk levels (Figure 13(b)). The app also uses a color scheme to give the users a direct visual cue of the risk level. We plan to expand the functionalities of DroidRisk app in future to report the risk level of any app in any Android market once the user provides the corresponding information.

[3] GreaseMonkey is a Firefox add-on that allows the user to customize the display or behavior of a web page using a user script [8].

7 Conclusion

In this paper, we demonstrate that the proposed DroidRisk framework can be used to improve the efficiency of Android permission system for informing the user about the risks of Android permissions and apps. It can be easily incorporated into existing Android malware detection systems as the first barrier to prevent the spread of malware. It will be especially useful for those third party markets without malware detection capability. In our future work, we will investigate alternative ways to find the impact levels of Android permissions, e.g. based on the study of [19] which ranks the risks of permissions according to the users' ratings. We will also explore security rules involving multiple permissions and evaluate their corresponding risks which may be underestimated by DroidRisk.

References

1. Android and security, http://googlemobile.blogspot.com/2012/02/android-and-security.html
2. Android enesoluty - fake antivirus, spyware, http://contagiominidump.blogspot.com/2012/11/android-enesoluty-fake-antivirus-spyware.html
3. Android races past apple in smartphone market share, http://money.cnn.com/2012/08/08/technology/smartphone-market-share/index.html
4. Aptoide, http://www.aptoide.com
5. Drioddream malware has now claimed 260,000 devices, http://androidjournalist.wordpress.com/2011/03/10/droiddream-malware-has-now-claimed-260000-devices/
6. Google play, https://play.google.com
7. Google play hits 25 billion downloads, http://officialandroid.blogspot.com/2012/09/google-play-hits-25-billion-downloads.html
8. Greasemonkey, http://www.greasespot.net
9. Mumayi, http://www.mumayi.com/
10. Smartphone sales to top 1.7 billion by 2017, dominate mobile phone market, http://www.inquisitr.com/230416/smartphone-sales-to-top-1-7-billion-by-2017-dominate-mobile-phone-market/
11. National Institue of Standards and Technology (NIST), Risk management guide for information technology systems (2002)
12. Alexander, C.: Market risk analysis: quantitative methods in finance. Wiley (2008)
13. Barrera, D., Kayacik, H.G., van Oorschot, P.C., Somayaji, A.: A methodology for empirical analysis of permission-based security models and its application to android. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, pp. 73–84 (2010)
14. Bradley, A.P.: The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recogn. 30(7), 1145–1159 (1997)

15. Chia, P.H., Yamamoto, Y., Asokan, N.: Is this app safe?: a large scale study on application permissions and risk signals. In: Proceedings of the 21st International Conference on World Wide Web, WWW 2012, pp. 311–320 (2012)
16. Enck, W., Ongtang, M., McDaniel, P.: On lightweight mobile phone application certification. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS 2009, pp. 235–245 (2009)
17. Fawcett, T.: An introduction to roc analysis. Pattern Recognition Letters 27(8), 861–874 (2006)
18. Felt, A.P., Chin, E., Hanna, S., Song, D., Wagner, D.: Android permissions demystified. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, pp. 627–638 (2011)
19. Felt, A.P., Egelman, S., Wagner, D.: I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. In: Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM 2012, pp. 33–44 (2012)
20. Felt, A.P., Greenwood, K., Wagner, D.: The effectiveness of application permissions. In: Proceedings of the 2nd USENIX Conference on Web Application Development, WebApps 2011, p. 7 (2011)
21. Felt, A.P., Ha, E., Egelman, S., Haney, A., Chin, E., Wagner, D.: Android permissions: user attention, comprehension, and behavior. In: Proceedings of the Eighth Symposium on Usable Privacy and Security, SOUPS 2012, pp. 3:1–3:14 (2012)
22. Frank, M., Dong, B., Felt, A.P., Song, D.: Mining permission request patterns from android and facebook applications. In: Proceedings of the IEEE International Conference on Data Mining, ICDM 2012 (2012)
23. Joh, H., Malaiya, Y.K.: Defining and assessing quantitative security risk measures using vulnerability lifecycle and cvss metrics. In: Proceedings of the 2011 International Conference on Security and Management, SAM 2011, pp. 10–16 (2011)
24. Sarma, B.P., Li, N., Gates, C., Potharaju, R., Nita-Rotaru, C., Molloy, I.: Android permissions: a perspective combining risks and benefits. In: Proceedings of the 17th ACM Symposium on Access Control Models and Technologies, SACMAT 2012, pp. 13–22 (2012)
25. Wei, X., Gomez, L., Neamtiu, L., Faloutsos, M.: Permission evolution in the android ecosystem. In: Proceedings of the 2012 Annual Computer Security Applications Conference, ACSAC 2012 (2012)
26. Zhou, Y., Jiang, X.: Dissecting android malware: characterization and evolution. In: Proceedings of the 33rd IEEE Symposium on Security and Privacy, Oakland 2012, pp. 95–109 (2012)
27. Zhou, Y., Wang, Z., Zhou, W., Jiang, X.: Hey, you, get off my market: detecting malicious apps in official and alternative android markets. In: Proceedings of the 19th Network and Distributed System Security Symposium, NDSS 2012 (2012)

A Model for Trust-Based Access Control and Delegation in Mobile Clouds*

Indrajit Ray[1], Dieudonne Mulamba[1], Indrakshi Ray[1], and Keesook J. Han[2]

[1] Dept. of Computer Science, Colorado State University, Fort Collins, CO 80523
{indrajit,mulamba,iray}@cs.colostate.edu
[2] Air Force Research Laboratory/RIGA, 525 Brooks Road, Rome, NY 13441
Keesook.Han@rl.af.mil

Abstract. Multi-tenancy, elasticity and dynamicity pose several novel challenges for access control in mobile smartphone clouds such as the Android™ cloud. Accessing subjects may dynamically change, resources requiring protection may be created or modified, and a subject's access requirements to resources may change during the course of the application execution. Cloud tenants may need to acquire permissions from different administrative domains based on the services they require. Moreover, all the entities participating in a cloud may not be trusted to the same degree. Traditional access control models are not adequate for mobile clouds. In this work, we propose a new access control framework for mobile smartphone clouds. We formalize a trust-based access control model with delegation for providing fine-grained access control. Our model incorporates the notion of trust in the Role-Based Access Control (RBAC) model and also formalizes the concept of trustworthy delegation.

Keywords: access control model, delegation, mobile cloud security, trust.

1 Introduction

Smartphones and other mobile devices are increasingly shifting the personal computing model away from traditional desktops and laptops to *mobile cloud computing*. In this model, cloud computing, mobile devices and networks seamlessly interact with each other to provide newer types of services that were previously not possible (such as location based services). The unique characteristics of mobile cloud computing – namely, multi-tenancy, elasticity, massive scalability [15], and mobility – introduce novel challenges to authorization and access control. To begin with, multi-tenancy results in the co-residency of machines (virtual machines, database engines etc.) and other resources owned by different clients or tenants at the same privileged position in the cloud with respect to one another. As a result, a guest operating system can exploit vulnerabilities in the hypervisor and run processes on other guests or the host, and security breaches can arise in one smartphone client and propagate to another easily via the cloud. Proper authorization and access control techniques should therefore not only protect tenant resources from un-authorized disclosure and modification from attackers, but also should

* Approved for Public Release; Distribution Unlimited:88ABW-2013-2127 dated 02 May 2013.

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 242–257, 2013.

allow *segregation* of tenants from one another, and *isolation* of computation, storage and network resources of the mobile cloud provider from tenants.

A mobile cloud environment is inherently very dynamic. The accessing entities may change, resources requiring protection may be created or modified, and an entity's access to resources may change during the course of an application execution. Users need to dynamically acquire permissions from different domains based on the services they need. Interactions among entities may occur in ad hoc manners and where the access-requesting entity may not be known in advance by the access-granting entity. In such situations, traditional identity-based access control models such as Discretionary Access Control (DAC), Mandatory Access Control (MAC) [21], or Role-based Access Control (RBAC) [17,12], that rely on the access-granter knowing the identity of access requester beforehand and authenticating the requester, can no longer be applied.

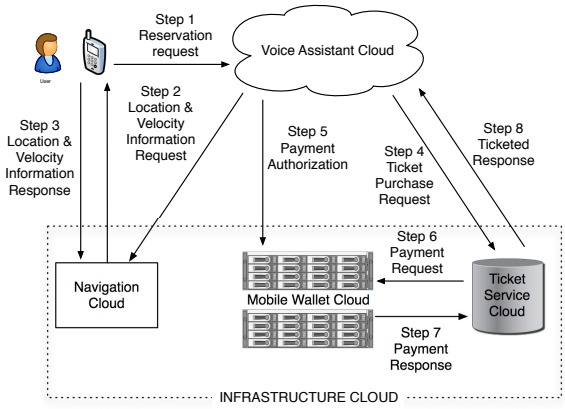


Fig. 1. Mobile smartphone cloud application illustrating need for delegation

Last but not least, in a mobile cloud environment resources are often distributed and managed by different service providers and clients move from one service provider to another to obtain the needed service. To support a specific service, there is frequently the need for coordination and interaction between these different providers. A tenant of one service provider may need to access other providers to obtain the relevant service. This is illustrated by the application scenario in Fig. 1. A mobile smartphone user invokes the voice assistant application (VAC) on her smartphone and instructs it to make a reservation for that evening's show at an en-route theater closest to her destination. VAC computes the time to reach the destination by accessing a navigation service (NavC), identifies candidate theaters by consulting a location service (LocC) and selects one from the list, contacts a ticket service provider cloud (TktC) for a reservation and contacts the user's mobile wallet provider (MobWC) to purchase the ticket from TktC.

For accessing different services the authorization sequence can potentially be as follows. The user initially authorizes the VAC for the ticket purchase task; however the VAC cannot carry out the task on its own and therefore needs to pass on the authorization (or portions thereof) to various other cloud providers. Moreover, if the different

service providers are all tenants of one or more infrastructure cloud(s) (as shown in Fig. 1), then each service provider needs to rely on the infrastructure to manage access control to its resources. *Delegation* of authorization is the principle that allows one service provider to act on behalf of the user to make the user's access rights available to other service providers. There is a need to manage and mediate interactions with distributed resources having distributed administrators. However, authentication of the requesting client (required in conventional access control) may be difficult in mobile smartphone cloud systems. For example, it may not be possible for the VAC to be registered with both the MobWC and the TktC. Hence, delegation may need to proceed without associated authentication. This is further complicated by the fact that often the privilege to delegate may itself need to be delegated.

In this work, we propose a new trust-based access control model that supports complex delegation across different members of the mobile cloud for providing fine-grained access controls. This model is based on extensions to the RBAC model, and adapts concepts from the trust-based access control models proposed earlier by us [7,19]. We assume an existing context sensitive non-binary trust model such as the one in [16]. For this work, we identify the different trust-based access control model elements that are needed for addressing specific challenges in smartphone clouds (Section 3.1), and the relationships among those elements (Section 3.2). We formalize the model and give the rules of access (Sections 3.3 and 3.4). The smartphone cloud system is very dynamic and allows frequent updates to its RBAC relations. Using traditional RBAC relations to control delegation in such environments is of limited advantage because of the resulting inconsistencies in role hierarchy. We adapt the notion of *administrative scope* to resolve dynamically any inconsistency involved in controlling delegations (Section 3.5). Finally, we propose algorithms to perform trust-based delegation, including chained delegation (Section 4).

2 Related Work

Role-Based Access Control (RBAC) is used extensively within organizations for administering and managing privileges and is often considered the de-facto standard for access control.

While the advantages of RBAC are numerous, researchers are increasingly identifying limitations in the model for newer and emerging applications. The biggest limitation is the lack of support for delegation in the standard RBAC model and the failure to support dynamic adaptation of access control policies based on changing needs. Many researchers have extended RBAC to support delegation [3,2,10,18,22,24,25]; however these models fail to support the need for ad hoc authorization and ad hoc delegation and thus cannot readily be adapted to the cloud environment. Researchers have also proposed Credential-based or Attribute-based access control models [6,5] to address the challenges of unknown users in access control. Unfortunately, these models do not allow access control decisions to be dynamically updated or revoked based on the history of the requester, or based on changing requirements of the requester.

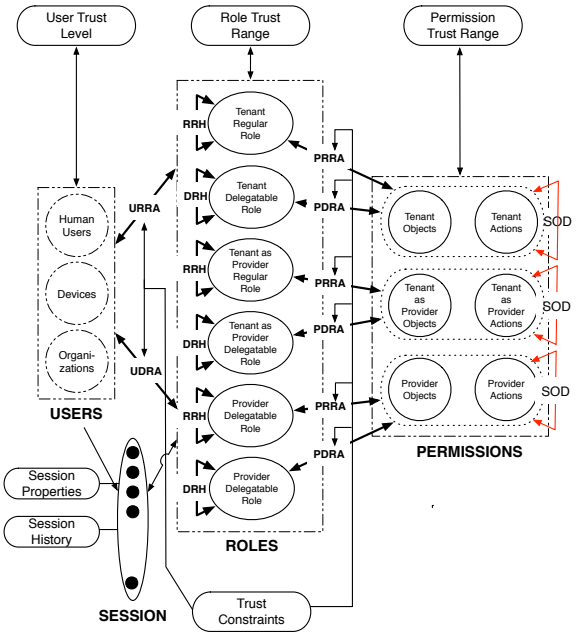


Fig. 2. Schematic overview of the new access control model for clouds

Recently, researchers have proposed the notion of risk-adaptable access control models [9,14,13] and trust-based access control models [7,4,23] to facilitate dynamic adaptation of access control policies based on operational needs and situational awareness. A recent work [1] combines these two philosophies into one comprehensive model. However, none of these works address the problem of delegation in mobile cloud systems. Nonetheless, these approaches look promising and form the basis of the current model.

3 Formal Access Control Model with Delegation

The proposed trust-based access control model is defined in terms of a set of elements and relations among those elements with trust-based constraints defined on these relations. We use a modified graph-theoretic approach similar to the one proposed by Chen and Crampton [8] to express the model semantics. This model is designed to integrate delegation and revocation, with revocation being the reverse process of delegation. The model offers the possibility to perform a single-step as well as a multi-step delegation and revocation. It enables both roles and permissions to be delegated. Fig. 2 provides a schematic overview of the proposed model. We use the access control scenario from Figure 1 to exemplify the model.

3.1 Model Elements

The model elements are of the following types: *user, user_properties, role, object, action, permissions, constraints, trust_level, session_instance, session_type, session,* and *session_history*. The corresponding element sets are represented by the symbols U, $Prop$, R, O, Act, P, $Const$, $\mathscr{D}$, S_I, S_T, S, S_H.

In this model, roles are separated into two broad classes: *regular roles* and *delegatable roles*. A user cannot delegate permissions assigned to a regular role, while permissions assigned to a delegatable role can be delegated. The cloud system is responsible for creating each regular role, while each delegatable role is created and owned by an individual user. Therefore, a regular role is a durable role, while a delegatable role is temporary, created and deleted at the user's discretion. In the cloud system, each user owns a set of delegatable roles that form a role hierarchy determined by the user. All user assignment to regular roles relations and all permission assignment to regular roles relations are managed by the cloud system, while each individual user is responsible for managing all user assignment to delegatable roles relations and all permission assignment to delegatable roles relations.

User. A user $u \in U$ is a human being, a device, an organization or any active agent running on behalf of these. Three categories of users can be found in the cloud, namely, tenant, a tenant-as-provider, and a provider. A tenant is a user that is receiving regular services offered in the cloud, while a tenant-as-provider is a cloud user that is receiving regular services as well as offering regular services. The last category, provider, refers to the cloud provider. To understand these notions, let consider the example of Netflix that uses Amazon's cloud. A Netflix subscriber is a tenant, while Netflix is the tenant-as-provider. The provider in this case is Amazon's cloud.

User_properties. Each user u has a certain set of properties $\mathscr{P}_u$, called *user_properties*. The set $Prop = \bigcup_{u \in U} \mathscr{P}_u$. A user can manifest any subset P_u of $\mathscr{P}_u$ (i.e., $P_u \in 2^{\mathscr{P}_u}$) in a particular session. User properties are used in our model to compute trust levels of users (see later in the list). Some examples of elements of user_properties are: user credentials, public key certificates, and membership in groups.

Role. The concept of role is the same as in the RBAC model. A role $r \in R$ is a job function with some associated semantics regarding the responsibilities conferred to the user. A user assigned to a role specifies the operational needs of the cloud. The set of roles R can be further subdivided into six disjoint subsets as follows:

1. TENANT-REGULAR-ROLE (TRR) – A set of job functions that are relevant for receiving regular services. For example, the human user in our scenario can be an elite member with the mobile wallet service provider that bestows him with certain privileges. In this case "elite member" will be an example of a tenant-regular role.
2. TENANT-DELEGATABLE-ROLE (TDR) – A set of job functions that are relevant for delegating services. Going back to our earlier scenario, the VAC instance working on behalf of the user to purchase tickets needs to have permission at the MobWC to use the user's wallet. Thus, a role such as "wallet user for ticket purchase" which the VAC needs to assume to execute the operation will be an example of tenant-delegatable-role.

3. TENANT-AS-PROVIDER-REGULAR-ROLE ($TPRR$) – A set of job functions that are relevant for receiving regular services as well as offering regular services. The NavC cloud is a provider of navigation services. It uses the infrastructure cloud to provide some of its services (such as the computation needed for the navigation). As a result, we will have a tenant-as-provider-regular-role "navigation computation" at the infrastructure cloud.
4. TENANT-AS-PROVIDER-DELEGATABLE-ROLE ($TPDR$) – A set of job functions that are relevant for receiving as well as offering delegation services. The VAC will have a tenant-as-provider-delegatable role "make ticket reservation" that will allow it to delegate different components of the ticket reservation.
5. PROVIDER-REGULAR-ROLE (PRR) – A set of job functions that are relevant for providing regular services. The instance of MobWC that will access the user's wallet needs to have permission to debit the wallet to provide the service. Thus, an example of this role will be "debit user wallet" that the instance of MobWC needs to acquire for this task.
6. PROVIDER-DELEGATABLE-ROLE (PDR) – A set of job functions that are relevant for providing delegation services. Assume that the TktC offers the ticketing service by using a database provider cloud to keep record of the transaction. Thus, an instance of the TktC that is performing the ticketing operation for the current user needs to delegate the record update operation to the database provider. A "decrement available seats" role will be an example of a provider-delegatable-role in this scenario.

Object. An object $o \in O$ is a data resource as well as a system resource. It can be thought of as a *container* that contains information. The set O (OBJECTS) is partitioned into three subsets:

1. TENANT-OBJECTS (TO) – Objects from this set are accessed when some services are needed. An example is the user's wallet at the MobWC.
2. TENANT-AS-PROVIDER-OBJECTS (TPO) – Objects from this set are accessed for receiving as well as providing services. An example is the database used by the TktC to keep record of transactions.
3. PROVIDER-OBJECTS (PO) – Objects in this set are accessed purely for providing services. Examples include objects used by the infrastructure cloud such as network objects.

Action. An action $a \in A$ is an executable image of a program that operates on some object. The set A (ACTIONS) may have three subtypes ACTIONS:

1. TENANT-ACTIONS (TA) – This set of actions act on objects from the set TENANT-OBJECTS. An example is "debit wallet".
2. TENANT-AS-PROVIDER-ACTIONS (TPA) – This set is comprised of actions that operate on elements of TENANT-AS-PROVIDER-OBJECTS. An example is "update database".
3. PROVIDER-ACTIONS (PA) – This set is comprised of actions that operate on elements of PROVIDER-OBJECTS. An example is "perform read on disk".

Permission. A permission $p \in P$ is an authorization to perform a certain task within the system. A permission is assigned to a role. The set P (PERMISSIONS) is partitioned into three subsets as follows:

1. TENANT-PERMISSIONS (TP) – The set of all ordered pairs $\langle TO, TA \rangle$ where $TO \in$ TENANT-OBJECTS and $TA \in$ TENANT-ACTIONS; that is, $TP = 2^{TO \times TA}$.
2. TENANT-AS-PROVIDER-PERMISSIONS (TPP) – The set of all ordered pairs $\langle TPO, TPA \rangle$ where $TPO \in$ TENANT-AS-PROVIDER-OBJECTS and $TPA \in$ TENANT-AS-PROVIDER-ACTIONS; that is, $TPP = 2^{TPO \times TPA}$.
3. PROVIDER-PERMISSIONS (PP) – The set of all ordered pairs $\langle PO, PA \rangle$ where $PO \in$ PROVIDER-OBJECTS and $PA \in$ PROVIDER-ACTIONS; that is, $PP = 2^{PO \times PA}$.

Trust_level. A trust_level $\in \mathscr{D}$ is a real number between 0 and 1, with 0 being the lowest level of trustworthiness and 1 the highest and intuitively encodes the security risk of the access decision. The larger the value of the trust_level the less risky it is to grant access to the user. A user u associated with a role r at some instant of time has a trust_level. The function $\mathscr{T}(u,r)$ gives the trust_level of the corresponding user associated with the role r.

Role_trust_range. A role_trust_range is an interval $[rt_{min}, 1]$ that is associated with roles. The role_trust_range indirectly encodes the situational factors under which the access decision is made. Each role $r \in R$ is associated with a role_trust_range that gives the minimum trust value rt_{min} required for a user to be assigned to the corresponding role. The function $\mathscr{L}(r)$ returns the value rt_{min} for the given role r.

Permission_trust_range. A permission_trust_range is an interval $[pt_{min}, 1]$ that is associated with permissions. Each permission $Perm \in P$ is associated with a permission_trust_range that gives the minimum trust value pt_{min} required by a role to be assigned the corresponding permission. The function $\mathscr{L}(Perm)$ returns the value pt_{min} for the given permission $Perm$.

Constraint. A constraint $\in$ Cons is defined as a predicate which when applied to a relation between two elements returns a value of "acceptable" (True or 1) or "not-acceptable" (False or 0). Constraints can be viewed as conditions imposed on the relationships and assignments and encode the situational factors under which access decisions need to be made.

Session_instance. A session_instance $\in S_I$ is a 'login' instance of an user. It is the set of roles activated by the user in that login instance. A user can instantiate multiple logins thereby initiating multiple session_instances at the same time. A session_instance is uniquely identified by a system generated id.

Session_type. A session_instance is identified with a type that is defined by the set of user_properties manifested in that session_instance by the user. For a session_instance u_s activated by a user u with set of properties $Prop_u$ ($Prop_u \subseteq \mathscr{P}_u$), the session_type is $prop_u$. Formally, the set $S_T = 2^{Prop}$.

Session. A session $\in S$ is identified by a session_instance with a session_type. A session with session_instance u_s and type $Prop_u$ is denoted by the symbol ${u_s}^{prop_u}$. Formally, $S = S_I \times S_T$.

Session_history. A session_history $\in S_H$ is a set of information regarding the user's roles, behavior and trust levels in a previous use of a session of that type.

3.2 User-Role and Permission-Role Assignment

We propose a graph theoretic semantics for our model. An *authorization graph* is defined in terms of a set V of vertices and a set E of edges. The set of vertices $V = U \cup TRR \cup TDR \cup TPRR \cup TPDR \cup PRR \cup PDR \cup TP \cup TPP \cup PP$ corresponds to the entities users, tenant-regular-roles, tenant-delegatable-roles, tenant-as-provider-regular-roles, tenant-as-provider-delegatable-roles, provider-regular-roles, provider-delegatable-roles, tenant-permissions, tenant-as-providers-permissions and provider-permissions respectively. The set of edges $E = URRA \cup UDRA \cup PRRA \cup PDRA \cup RRH_a \cup DRH_a \cup RRH_u \cup DRH_u$ corresponds to the different relationships between the model's entities.

We now define some relationships with roles that form the building block for the access control model, as follows:

Definition 1. *User-Regular-Role Assignment (URRA)* = $(U \times TRR) \cup (U \times TPRR) \cup (U \times PRR)$ *is a many-to-many relationship that provides the regular roles to which different users can be assigned in the cloud. It is represented as a pair of user-regular-roles.*

Definition 2. *User-Delegatable-Role Assignment (UDRA)* = $(U \times TDR) \cup (U \times TPDR) \cup (U \times PDR)$ *is a many-to-many relationship identifying delegatable roles to which different users can be assigned in the cloud. This relationship is represented as a pair of user-delegatable-roles.*

Definition 3. *Permission-Regular-Role Assignment (PRRA)* = $(TRR \times TP) \cup (TPRR \times TPP) \cup (PRR \times PP)$ *is a many-to-many relationship providing the set of permission-regular-roles that indicate to regular roles which permissions they are allowed to acquire.*

Definition 4. *Permission-Delegatable-Role Assignment (PDRA)* = $(TDR \times TP) \cup (TPDR \times TPP) \cup (PDR \times PP)$ *is a many-to-many relationship providing the set of permission-delegatable-roles identifying the permissions for which delegatable roles are authorized.*

The role-hierarchy defines parent-child relationships between different roles. This allows for easy administration of roles by reducing the number of explicit assignments in the *URRA*, *UDRA*, *PRRA*, and *PDRA* relations. Regular roles are organized in a Regular-Role-Hierarchy while delegatable roles are organized in a Delegatable-Role-Hierarchy with the two hierarchies being disjoint. We define the two hierarchies as follows:

Definition 5. *The Role-hierarchy (RH)* = $\Re \times \{a,u\}$*, is a partial order on the set of roles where* $\Re = ((TRR \times TRR) \cup (TPRR \times TPRR) \cup (PRR \times PRR))$ *in case role is a regular role. In the same case RH is written RRH to represent a regular role-hierarchy. When considering the case when role is a delegatable role, then* $(\Re = ((TDR \times TDR) \cup (TPDR \times TPDR) \cup (PDR \times PDR))$*, and RH becomes DRH to represent a delegatable role-hierarchy. The set* $\{a,u\}$ *can be considered as one of these:*

- *the* role-activation hierarchy
- *the* permission-usage hierarchy

Let $\leq_a$ denote the reflexive transitive closure of $\Re_a$ while $\leq_u$ denotes the reflexive transitive closure of $\Re_u$. Then role-activation hierarchy and permission-usage hierarchy are defined as follows.

Definition 6. *Role-activation hierarchy* ($\Re_a$) = $\{(r,r') : (r,r',a) \in \Re\}$ *is a subset of* $\Re$ *such that a user u may activate a role r in a session_instance if there exists a role* r' *such that* $\{u,r'\} \in URRA$ *and* $r \leq_a r'$ *if r and* r' *are regular roles, or* $\{u,r'\} \in UDRA$ *and* $r \leq_a r'$ *when r and* r' *are delegatable roles.*

Definition 7. *Permission usage hierarchy* ($\Re_u$) = $\{(r,r') : (r,r',u) \in \Re\}$ *such that a user is authorized for permission p if there exists roles* r,r' *such that u may activate* r'*, with* $(p,r) \in PRRA$ *and* $r \leq_u r'$ *in the case* r,r' *are regular roles, or with* $(p,r) \in PDRA$ *and* $r \leq_u r'$ *when* r,r' *are delegatable roles. In the first case* $\Re_u$ *becomes* RRH_u*, while it becomes* DRH_u *when roles are delegatable.*

3.3 Evaluating Access Requests

The cloud security administrator assigns trust constraints in the form of a corresponding *trust interval* to roles, permissions, and other associations between entities based on different characteristics of each model. Note that in the cloud structure, users, tenants or providers of the senior role can perform the same set of duties as its junior role; hence an entity who will be assigned to the senior role require more trustworthiness than the entity who will be assigned to junior role only. Based on this observation, when we introduce the notion of trust value, we assume that the trust value of the senior role always dominates the trust values of its junior roles.

We have previously proposed the notions of *activation path*, *usage path* and *access path* in [19] to evaluate if an access control request should be denied or granted according to the current policy. We adapt and extend these notions here to determine access request for mobile smartphone clouds.

Definition 8. *An activation path (or act-path) between two nodes* v_1 *and* v_n *in the authorization graph is a sequence of vertices* $v_1,\ldots,v_n \in E$ *such that either* $(v_1,v_2) \in URRA$ *and* $(v_{i-1},v_i) \in RRH_a$ *for* $i = 3,\ldots,n$ *on a regular-role hierarchy or* $(v_1,v_2) \in UDRA$ *and* $(v_{i-1},v_i) \in DRH_a$ *for* $i = 3,\ldots,n$ *on a delegatable-role hierarchy.*

Definition 9. *A usage path (or u-path) between two nodes* v_1 *and* v_n *in the authorization graph is a sequence of vertices* $v_1,\ldots,v_n \in E$ *such that either* $(v_i,v_{i+1}) \in RRH_u$ *for* $i = 1,\ldots,n-2$*, and* $(v_{n-1},v_n) \in PRRA$ *on a regular-role hierarchy or* $(v_i,v_{i+1}) \in DRH_u$ *for* $i = 1,\ldots,n-2$*, and* $(v_{n-1},v_n) \in PDRA$ *on a delegatable-role hierarchy.*

Definition 10. *An access path (or acs-path) between two nodes* v_1 *and* v_n *in the authorization graph is a sequence of vertices* $v_1,\ldots,v_n$*, such that* (v_1,v_k) *is an act-path, and* (v_k,v_n) *is an u-path.*

Based on these definitions of usage path, access path and activations we propose three different access control models, namely, the *standard model*, the *strong model* and the *weak model*. The models differ with respect to the trust constraints that must be satisfied by the entities for the authorization to be successful.

3.4 The Standard Model

In the standard model, individual entities are associated with trust values and describe how much the user is trusted to perform a specific role. The role_trust_range associated with a tenant role, a tenant-as-provider role or a provider role specifies the minimum trust level with respect to that role that the user has to acquire in order to activate the corresponding role. Similarly, the permission_trust_range associated with a tenant-permission, tenant-as-provider-permission or provider-permission specifies the minimum trust level with respect to the current role of the user, that needs to be acquired in order to invoke these permission.

Let the trust values for the user be specified by a function $\mathscr{T} : ((U_h \times R_h) \cup (U_d \times R_d)) \rightarrow t \in \mathscr{D}$ and the trust ranges for role and permission by functions $\mathscr{L} : (R \cup P) \rightarrow [l, 1] \subseteq \mathscr{D}$.

- For $u \in U, r \in R$, $\mathscr{T}(u, r)$ denotes the trust value of u with respect to r;
- For $r \in R$, $\mathscr{L}(r)$ denotes the trust interval in which r is active;
- For $p \in P$, $\mathscr{L}(p)$ denotes the trust interval in which p is active.

For any path $v_1, \ldots, v_n$ in the graph $G = (V, E, \mathscr{T}, \mathscr{L})$, where $E = URRA \cup UDRA \cup PRRA \cup PDRA \cup RRH_a \cup DRH_a \cup RRH_u \cup DRH_u$, we write $\hat{\mathscr{L}}(v_2, \ldots, v_n) = \hat{\mathscr{L}}(v_2, v_n) \subseteq \mathscr{D}$ to denote $\bigcap_{i=2}^{n} \mathscr{L}(v_i)$. In other words, $\hat{\mathscr{L}}(v_2, v_n)$ is the trust interval in which every vertex $v_i \in R \cup P$ is enabled.

Authorization in the Standard Model is based on the policy decision point making the following three determinations corresponding to the requested access:

1. Is the role that the user needs to activate in the current session in order to acquire the desired permission, authorized for the permission?
2. Can the user activate the corresponding role?
3. Is the user authorized for the desired permission?

If the policy decision point makes a positive determination for all these conditions a decision to allow the access is made. These three determinations are made based on the following propositions.

Proposition 1. *A role $v_1 \in R$ is authorized for permission $v_n \in P$ if and only if there exists an u-path $v_1, v_2, \ldots, v_n$ and $\mathscr{L}(v_1) \subseteq \hat{\mathscr{L}}(v_1, v_n)$.*

Proposition 2. *A user $v_1 \in U$ may activate role $v_n \in R$ if and only if there exists an act-path $v_1, v_2, \ldots, v_n$ and $\mathscr{T}(v_1, v_2) \in \mathscr{L}(v_2)$.*

Proposition 3. *A user $v_1 \in U$ is authorized for permission $v_n \in P$ if and only if there exists an acs-path $v_1, v_2, \ldots, v_k, \ldots, v_n$ such that $v_k \in R$ for some k, $v_1, \ldots, v_k$ is an act-path, $v_k, \ldots, v_n$ is a u-path, v_1 can activate v_k, and v_k is authorized for v_n.*

3.5 Controlling Delegation

A *delegation operation* temporarily changes the access control state so as to allow the delegatee to use the initial user's access privileges. Delegation operations are classified into two kinds: *grant* operations and *transfer* operations [3]. Grant operations are defined by *grant delegation models* and allow the delegated access rights to be available to both the delegator and the delegatee, after a successful completion of a delegation operation. Transfer operations, on the other hand, are defined by *transfer delegation models* to allow the delegated access right to be acquired by the delegatee, while preventing the delegator from continuing to use the delegated access right. Most works done in this field focus on grant operation [3,2,18,22,24,25]. Crampton and Khambhammettu first introduced the notion of transfer operation in [10].

Designing a mechanism for delegation control involves specifying under which condition a delegation operation is permitted. This requires resolving, in order, the following two issues: (1) Determine whether or not a given user is authorized to delegate a role or permission available to the user. (2) Determine whether a given role or permission can be delegated to a user.

Relations have been the main way to control delegation [2,24,25] in role-based systems. Two principal relations, *can_delegate* and *can_receive*, are used for controlling delegation. Relations are suited to be used in a *RBAC* system but only where *RBAC* relations remain static. However, in a cloud system, *RBAC* relations, such the role hierarchy, are frequently updated. This situation can generate serious inconsistencies in relations *can_delegate* and *can_receive* that specify respectively, who is authorized to delegate access rights, and who is allowed to receive delegated access rights. For this reason, we have opted to use the concept of *administrative scope* [11] to deal with delegation control. Administrative scope is a concept borrowed from *RHA* family of administrative models, and is defined as follows [11].

Definition 11. *Let $r \in R$. We define $\sigma(r)$ the* administrative scope of a role *r as the set $\sigma(r) = \{r'\}$ such that if a role $r' \in \sigma(r)$ then in the graph formed by the role hierarchy, any path starting at r' passes through r.*

Definition 12. *Let s be a session activated by a user u. We define $\sigma(s)$ the* administrative scope of a session *s as $\sigma(s) = \cup_{r \in s} \sigma(r)$. The idea expressed by this definition is that the administrative scope of a session is simply the union of administrative scopes of all roles activated in the session s by the user u.*

The concept of administrative scope provides us with a way to divide a role hierarchy graph into sub-hierarchies that form a natural unit of administration for the role *r*. Two approaches are the most favored for performing role-based administration, *ARBAC*97 and administrative scope. Among these two approaches, administrative scope is found to be a more flexible approach [11]. This quality makes it suitable for role-based delegation in a highly dynamic environment that is the cloud system.

Using administrative scope, a user *u* is allowed to delegate a role *r* only if $r \in \sigma(s)$, the administrative scope of the users session. This limits the user to delegate only roles that are within his administrative scope. Administrative scope can be used to regulate who can receive a delegation. A user *v* is authorized to receive a delegation of a role

r from user u if for all $r' < r$ such that $r' \notin \sigma(s)$, there exists r'' such that $r' \leq r''$ and $(v, r') \in UDA$. In order words, the delegatee v is authorized to receive the delegation of a role r if v is already authorized for all role $r' < r$. In order for the delegatee to receive a delegation, the idea is to require him to be already assigned to any roles outside the delegators administrative scope that the delegatee will inherit by successfully receiving the delegation.

4 Chaining Delegation and Transfer

A user that has previously received some access rights through a delegation process may decide to further delegate those rights. This can result in what can be called a chained delegation. In this delegation chain the trustworthiness of users is used by each node as a factor to decide about the level of delegation. These trust values form a trust chain. We use the notion of trust graph expressed in [20] to formalize the notion of trust chain in a given context. A trust graph $T = (N, E)$ is a directed acyclic graph that is defined in terms of a set N of nodes and a set E of edges. Nodes represent entities in the delegation chain, while edges represent trust relationship between these entities. These entities could be either users or roles respectively for user delegation and role delegation.

Using trust graphs node n_i trusting node n_j can be represented by (n_i, n_j). The degree to which an entity n_i trusts n_j is represented by a weight denoted by $w(n_i, n_j)$, $0 < w(n_i, n_j) \leq 1$, on the edge (n_i, n_j). In order to control the propagation of the access right during a delegation process, every entity puts a constraint on its trust relationship. This trust constraint is denoted by $c(n_i, n_j)$ where $0 < c(n_i, n_j) \leq 1$. Access rights can be propagated only if $c(n_i, n_j) \leq w(n_i, n_j)$. This implies that delegation cannot be carried along every edge of the trust graph. Therefore identifying valid paths to carry on a delegation becomes a challenging problem.

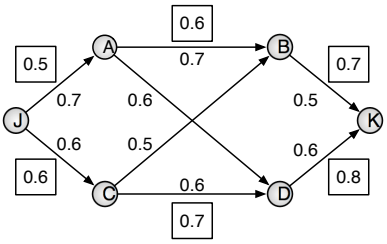


Fig. 3. Example of a trust graph

We consider the trust graph represented in Fig. 3. The numbers that are within boxes associated with the edges denote the trust values on the corresponding edges of the graph while the numbers that are not inside boxes denote the trust constraints. Assume that the entity J desires to identify all the valid paths it can use to delegate some access rights to entity K. Using the trust values and the trust constraints, the first challenge is to

determine all valid paths and thereafter to compute the transitive trust value for each of the found valid paths from J to K. We use three operators on trust values – a *comparison* operator to compare the trust value and the trust constraint, a *parallel* operator to find the minimum of two trust values, and a *sequential* operator that gives the product of two trust values.

Definition 13. *Comparison Operator* $\ominus : [0,1] \times [0,1] \rightarrow 0,1$ *is a binary operation whose input is a trust value and a trust constraint, and that returns* 1 *when the trust value is greater or equal to the trust constraint. Otherwise, the operator returns* 0.

Definition 14. *Parallel Operator, denoted* $\oplus : [0,1] \times [0,1] \rightarrow [0,1]$*, is a binary operator whose inputs are two trust values, and that returns the minimum of the two trust values.*

Definition 15. *Sequential Operator, denoted* $\otimes : [0,1] \times [0,1] \rightarrow [0,1]$*, is a binary operator whose inputs are two trust values, and that returns the product of these two trust values.*

The comparison operator is used to identify all the valid paths in a trust graph. Those valid paths are the ones able to carry on the delegation from J to K. The checking of valid paths may return either multiple valid paths, or a single valid path. In either case, we need to compute the transitive trust value. The parallel operator is used to compute the transitive trust value in case of multiple valid paths while the sequential operator is used for the same purpose but in case of a single valid path.

These operators are used within algorithm 1 to find the transitive trust value. The algorithm works as follows. Using a set of edges of the trust graph given as input, the algorithm uses the comparison operator on each pair of edges that form an arc to compare its trust values to its trust constraint. All arcs whose trust value is at least equal to the trust constraint are retained. If after being linked those arcs form one or more paths from the source node to the destination node, then those paths are considered to be the valid paths. If the algorithm find multiple valid paths, it uses the parallel operator to find the path with the minimum transitive trust value. Otherwise, the algorithm uses the sequential operator to return the transitive trust value of the unique path. In case of multiple valid paths, we made the design decision to choose the path with the minimum transitive trust value in order to be more conservative and more protective. Choosing the path with a maximum transitive trust value is also a valid choice, but is not considered in this work.

To see how the propagation of delegation works, we use the trust graph in Fig. 3 and execute our algorithm in order to find the transitive trust value from node J to node K. Before running the algorithm on the given graph, a processing step is necessary. This preprocessing involves doing a Depth-First Search on the graph in order to collect all the arcs from nodes J to K. The result of the DFS is as follows. $DFS(G) = (J,C),(C,D),(D,K),(C,B),(B,K),(J,A),(A,D),(A,B)$. This list of edges is provided to the algorithm as input.

Algorithm 1. Algorithm to identify all valid paths for delegation propagation

Input: edges $e_1, ..., e_n$
Output: set of valid paths and transitive trust on the valid paths
$valid\_paths \leftarrow \{\}$
for all i : 1 ≤ i ≤ n **do**
  $comp\_trust_i \leftarrow 0$
end for
for all $i : 1 \leq i \leq n$ **do**
  $compt\_trust_l \leftarrow w(e_i) \ominus c(e_i)$
  if $comp\_trust_l = 1$ **then**
    $valid\_paths \leftarrow (e_i)$
  end if
end for
if |Valid_paths| > 1 **then**
  Let $valid\_paths = \{(n_1, n_{2_1}, \ldots, n_{k-1_1}, n_{k_1}), (n_1, n_{2_2}, \ldots$
  $n_{k-1_2}, n_{k_2}), ..., (n_1, n_{2_j}, ..., n_{k-1_j}, n_{k_j})\}$
  min ← 1;
  for all $l : 1 \leq l \leq j$ **do**
    $trans\_trust_l \leftarrow 0$
  end for
  for all $l : 1 \leq l \leq j$ **do**
    for all $i : 1 \leq i \leq (k-2)$ **do**
      $trans\_trust_l \leftarrow trans\_trust_l + w(n_i, n_{i+1}) \otimes w(n_{i+1}), n_{i+2})$
    end for
  end for
  for all $l : 1 \leq l \leq j$ **do**
    $min \leftarrow min \oplus (trans\_trust)_l$
  end for
end if
if | $valid\_paths$ |= 1 **then**
  Let $valid\_paths = \{n_1, n_2, \ldots, n_k\}$
  $trans\_trust_l \leftarrow 0$
  for all $i : 1 \leq i \leq (k-2)$ **do**
    $trans\_trust_l \leftarrow trans\_trust_l \times w(n_i, n_{i+1}) \otimes w(n_{i+1}, n_{i+2})$
  end for
end if
return $valid\_paths, trans\_trust_l$

Given the list of arcs, we run the algorithm as follows.

- (J,C) : w(J,C) ⊖ c(J,C) = 0.6 ⊖ 0.6 = 1 valid_paths = {(J,C)}
- (C,D) : w(C,D) ⊖ c(C,D) = 0.7 ⊖ 0.6 = 1 valid_paths = {(J,C),(C,D)}
- (D,K) : w(D,K) ⊖ c(D,K) = 0.8 ⊖ 0.6 = 1 valid_paths = {(J,C),(C,D),(D,K)}
- (C,B) : w(C,B) ⊖ c(C,B) = 0.6 ⊖ 0.5 = 1 valid_paths = {(J,C),(C,D),(D,K),(C,B)}
- (B,K) : w(B,K) ⊖ c(B,K) = 0.7 ⊖ 0.5 = 1 valid_paths = {(J,C),(C,D),(D,K),(C,B),(B,K)}
- (J,A) : w(J,A) ⊖ c(J,A) = 0.5 ⊖ 0.7 = 0 valid_paths = {(J,C),(C,D),(D,K),(C,B),(B,K)}
- (A,D) : w(A,D) ⊖ c(A,D) = 0.4 ⊖ 0.6 = 0 valid_paths = {(J,C),(C,D),(D,K),(C,B),(B,K)}
- (A,B) : w(A,B) ⊖ c(A,B) = 0.6 ⊖ 0.7 = 0 valid_paths = {(J,C),(C,D),(D,K),(C,B),(B,K)}
- Finally valid paths = {J,C,D,K}, {J,C,B,K}

Since we have found two valid paths, we will run the first if statement of the algorithm as follows.

- $trans\_trust_1 = 0 + 0.6 \otimes 0.7 \otimes 0.8 = 0.336$
- $trans\_trust_2 = 0 + 0.6 \otimes 0.6 \otimes 0.7 = 0.252$
- $min = trans\_trust_1 \oplus trans\_trust_2 = 0.336 \oplus 0.252$

$trans\_trust_2 = 0.252$ is the minimum transitive trust value. Therefore the valid path J,C,B,K is the one that will be used to propagate the delegation of access rights from J to K.

5 Conclusion

In this work, we formalized a trust-based access control model for providing fine-grained access in smartphone clouds. The model extends traditional RBAC with the notion of trust and also addresses the problem of trustworthy delegations. A lot of work remains to be done. We plan to analyze this model and detect inconsistencies and errors in the policy specification. The access control configuration of the cloud is dynamic in nature. Towards this end, we plan to investigate how to perform access control analysis in real-time to ensure that security breaches do not occur. We also plan to deploy this model in real-world environments and study its impact on the performance and usability of cloud applications.

Acknowledgment. This work was supported in part by a fellowship from AFRL Information Institute under the 2012 Summer VFRP. The authors thank Mr. John Graniero AFRL/RIB for his support of this research. The views and conclusions are those of the authors, and should not be automatically interpreted as representing official policies, either expressed or implied, of the Air Force Research Laboratory or other federal government agencies.

References

1. Baracaldo, N., Joshi, J.B.D.: A Trust-and-Risk Aware RBAC Framework: Tackling Insider Threat. In: Proceeding of the Symposium on Access Control Models and Technologies, Newark, NJ (June 2012)
2. Barka, E., Sandhu, R.: A Role-Based Delegation Model and Some Extensions. In: Proceedings of the 16th Annual Computer Security Applications Conference, New Orleans, Louisiana, USA (December 2000)
3. Barka, E., Sandhu, R.: Framework for Role-Based Delegation Models. In: Proceedings of the 23rd National Information Systems Security Conference, Baltimore, Maryland, USA (October 2000)
4. Bhatti, R., Bertino, E., Ghafoor, A.: A Trust-based Context-Aware Access Control Model for Web-Services. In: Proceedings of the IEEE International Conference on Web Services (ICWS 2004), pp. 184–191. IEEE Computer Society, San Diego (2004)
5. Bobba, R., Fatemieh, O., Gunter, C.A., Khurana, H.: Using Attribute-Based Access Control to Enable Attribute-Based Messaging. In: Proceedings of the Annual Computer Security Applications Conference, Miami Beach, FL (December 2006)
6. Bonati, P., Samarati, P.: A Unified Framework for Regulating Access and Information Release on the Web. Journal of Computer Security 10(3), 241–272 (2002)
7. Chakraborty, S., Ray, I.: TrustBAC: Integrating Trust Relationships into the RBAC Model for Access Control in Open Systems. In: Proceedings of the 11th ACM Symposium on Access Control Models and Technologies, Lake Tahoe, CA (June 2006)
8. Chen, L., Crampton, J.: On Spatio-Temporal Constraints and Inheritance in Role-Based Access Control. In: Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, Tokyo, Japan (March 2008)
9. Cheng, P.C., Rohatgi, P., Keser, C., Karger, P.A., Wagner, G.M., Reninger, A.S.: Fuzzy Multi-Level Security: An Experiment on Quantified Risk-Adaptive Access Control. In: Proceedings of 27th IEEE Symposium on Security and Privacy, Oakland, CA (May 2007)

10. Crampton, J., Khambhammettu, H.: Delegation in Role-Based Access Control. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 174–191. Springer, Heidelberg (2006)
11. Crampton, J., Loizou, G.: Administrative Scope: A Foundation for Role-Based Administrative Model. ACM Transaction on Information and System Security 6(2), 201–231 (2003)
12. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST Standard for Role-Based Access Control. ACM Transactions on Information and Systems Security 4(3), 224–274 (2001)
13. Kandala, S., Sandhu, R., Bhamidipati, V.: An Attribute Based Framework for Risk-Adaptive Access Control Models. In: Proceedings of the 5th International Conference on Availability, Reliability and Security, Vienna, Austria (August 2011)
14. McGraw, R.W.: Risk-Adaptable Access Control. In: Proceedings of the 1st NIST Privilege Management Workshop, Gaithersburg, MD (September 2009)
15. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. NIST Special Publication 800-145 (September 2011), `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`
16. Ray, I., Ray, I., Chakraborty, S.: An Interoperable Context Sensitive Model of Trust. Journal of Intelligent Information Systems 32(1), 75–104 (2009)
17. Sandhu, R., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role Based Access Control Models. IEEE Computer 29(2), 38–47 (1996)
18. Tamassia, P., Yao, D., Winsborough, W.: Role-Based Cascaded Delegation. In: Proceedings of the 9th ACM Symposium on Access Control Models and Technologies, Yorktown Heights, New York, USA (June 2004)
19. Toahchoodee, M., Abdunabi, R., Ray, I., Ray, I.: A Trust-Based Access Control Model for Pervasive Computing Applications. In: Gudes, E., Vaidya, J. (eds.) Data and Applications Security XXIII. LNCS, vol. 5645, pp. 307–314. Springer, Heidelberg (2009)
20. Toahchoodee, M., Xie, X., Ray, I.: Towards Trustworthy Delegation in Role-Based Access Control Model. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 379–394. Springer, Heidelberg (2009)
21. U.S. Department of Defense: Trusted Computer System Evaluation Criteria. Department of Defense Standard DOD 5200-28-STD (December 1985)
22. Wainer, J., Kumar, A.: A Fine-Grained, Controllable, User-to-User Delegation Method in RBAC. In: Proceedings of the 10th ACM Symposium on Access Control Models and Technologies, Stockholm, Sweden (June 2005)
23. Ya-Jun, G., Fan, H., Qing-Guo, Z., Rong, L.: An Access Control Model for Ubiquitous Computing Application. In: Proceedings of the 2nd International Conference on Mobile Technology, Applications and Systems, Guangzhou, China (November 2005)
24. Zhang, L., Ahn, G.J., Chu, B.T.: A Rule-Based Framework for Role-Based Delegation and Revocation. ACM Transaction on Information and System Security 6(3), 404–441 (2003)
25. Zhang, X., Oh, S., Sandhu, R.: A Flexible Delegation Model in RBAC. In: Proceedings of the 8th ACM Symposium on Access Control Models and Technologies, Como, Italy (June 2003)

Result Integrity Verification of Outsourced Frequent Itemset Mining

Boxiang Dong, Ruilin Liu, and Hui (Wendy) Wang

Stevens Institute of Technology, Hoboken, NJ, USA
{bdong,rliu3,Hui.Wang}@stevens.edu

Abstract. The data-mining-as-a-service ($DMaS$) paradigm enables the data owner (client) that lacks expertise or computational resources to outsource its mining tasks to a third-party service provider (server). Outsourcing, however, raises a serious security issue: how can the client of weak computational power verify that the server returned correct mining result? In this paper, we focus on the problem of frequent itemset mining, and propose efficient and practical probabilistic verification approaches to check whether the server has returned correct and complete frequent itemsets.

Keywords: Cloud computing, data mining as a service, integrity verification.

1 Introduction

Cloud computing, an emerging trend of provisioning scalable computing services, provides the opportunity that data mining is offered as an outsourced service. Though the data-mining-as-a-service ($DMaS$) paradigm is advantageous to achieve sophisticated data analysis in a cost effective way, end users hesitate to place full trust in Cloud computing. This raises serious security concerns. One of the main security issues is the *integrity* of the mining result. There are many possible reasons for the service provider to return incorrect answers. For instance, the service provider would like to improve its revenue by computing with less resources while charging for more. Therefore, it is important to provide efficient mechanisms to verify the result integrity of outsourced data mining computations.

In this paper, we focus on *frequent itemset mining*, an important data mining problem, as the main outsourced data mining service. We aim to address the particular problem of verifying whether the server has returned correct and complete frequent itemsets. By *correctness*, we mean that all itemsets returned by the server are frequent. By *completeness*, we mean that no frequent itemset is missing in the server's result.

The key idea of our verification methods is to construct a set of (in)frequent itemsets from *real* items, and use these (in)frequent itemsets as *evidence* to check the integrity of the server's mining result. We remove real items from the original dataset to construct artificial infrequent itemsets, and insert copies of items

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 258–265, 2013.

that exist in the dataset to construct artificial frequent items. A nice property of our verification approach is that the number of required evidence (in)frequent itemsets is independent from the size of the dataset as well as the number of real frequent itemsets. This is advantageous as our verification approach will be especially suitable for verification of frequent mining on large datasets. Compared with the verification techniques based on fake items (e.g, [13]), our verification techniques are more robust to catch the untrusted server that may try to escape verification by utilizing additional background knowledge such as the item frequency distribution information in the outsourced data. Our experimental results show that our verification approach can achieve strong correctness/completeness guarantee with small overhead.

The paper is organized as follows. We discuss related work in Section 2 and preliminaries in Section 3. We present our EF and EI construction mechanisms for completeness and correctness verification in Section 4 and 5 respectively. In Section 6 we describe the post-processing procedures at the client side. In Section 7, we evaluate the performance of our approach. We conclude in Section 8.

2 Related Work

The problem of verifiable computation was tackled previously by using interactive proofs [4], probabilistically checkable proofs [2], zero-knowledge proofs [14], and non-interactive verifiable computing [3]. Unfortunately, this body of theory is impractical, due to the complexity of the algorithms and difficulty to use general-purpose cryptographic techniques in practical data mining problems.

In the last decade, intensive efforts have been put on the security issues of the database-as-a-service (DaS) paradigm (e.g., [6,9]). The main focus is the integrity (i.e., correctness and completeness) of result of range query evaluation. Only until recently some attention was paid to the security issues of the data-mining-as-a-service ($DMaS$) paradigm [10]. However, most of these work only focus on how to encrypt the data to protect data confidentiality and pattern privacy, while we focus on integrity verification of mining result.

There is surprisingly very little research [13,8] on result verification of outsourced data mining computations in the $DMaS$ paradigm. Among these work, [13] is the one the most related to ours. It proposed a result verification scheme for outsourced frequent itemset mining. Its basic idea is to insert some *fake items* that do not exist in the original dataset into the outsourced data; these fake items construct a set of fake (in)frequent itemsets. Then by checking the fake (in)frequent itemsets, the client can verify the correctness and completeness of the mining answer by the server. Though effective, this method assumes that the server has no background knowledge of the items in the outsourced data, and thus it has equal probability to cheat on the fake and real itemsets. We argue that using fake items cannot catch the malicious server that may have some background knowledge of the outsourced dataset. For example, if the server knows that there are k unique items in the original dataset, let $k'(k' > k)$ be the number of items in the outsourced dataset. The probability that an item is real is k/k'.

If the number of artificial items is relatively small compared with the number of real items, the server has a high probability to identify a real item. Furthermore, the verification approach in [13] still preserves the frequency of items, which may enable the server to identify the real/artificial items by the frequency-based attack (e.g, [12,11]). Our approach is much more challenging than using fake items (as in [13]), since insertion/deletion of real items may modify the true frequent itemsets. Our goal is to minimize the undesired change on the real frequent itemsets, while provide quantifiable correctness/completeness guarantee of the returned result.

3 Preliminaries

Frequent Itemset Mining. Given a transaction dataset D that consists of n transactions, let $\mathcal{I}$ be the set of unique items in D. The *support* of the itemset $I \subseteq \mathcal{I}$ (denoted as $sup_D(I)$) is the number of transactions in D that contain I. An itemset I is *frequent* if its support is no less than a support threshold min_{sup} [1]. The (in)frequent itemsets behave the following two monotone properties: (1) any superset of an infrequent itemset must be infrequent, and (2) any subset of a frequent itemset must be frequent.

Untrusted Server and Verification Goal. Due to many reasons (e.g., code bugs, software misconfiguration, and inside attack), a service provider may return incorrect data mining results. In this paper, we consider the server that possesses the background knowledge of the outsourced dataset, including the domain of items and their frequency information, and tries to escape from verification by utilizing such information. We formally define the correctness and completeness of the frequent itemset mining result. Let F be the real frequent itemsets in the outsourced database D, and F^S be the result returned by the server. We define the *precision* P of F^S as $P = \frac{|F \cap F^S|}{|F^S|}$ (i.e., the percentage of returned frequent itemsets that are correct), and the *recall* R of F^S as $R = \frac{|F \cap F^S|}{|F|}$ (i.e., the percentage of correct frequent itemsets that are returned). Our aim is to catch any answer that does not meet the predefined precision/recall requirement with high probability. Formally, given a dataset D, let F^s be the set of frequent itemsets returned by the server. Let pr_R and pr_P be the probability to catch F^s of recall $R \leq \alpha_1$ and precision $P \leq \alpha_2$, where $\alpha_1, \alpha_2 \in [0, 1]$ are given thresholds. We say a verification method M can verify (α_1, β_1)-*completeness* ((α_2, β_2)-*correctness*, resp.) if $pr_R \geq \beta_1$ ($pr_P \geq \beta_2$, resp.), where $\beta_1 \in [0, 1]$ ($\beta_2 \in [0, 1]$, resp.) is a given threshold. Our goal is to find a verification mechanism that can verify (α_1, β_1)-completeness and (α_2, β_2)-correctness.

4 Construction of Evidence Frequent Itemsets (*EFs*)

Our key idea of completeness verification is that the client uses a set of frequent itemsets as the *evidence*, and checks whether the server misses any evidence frequent itemset in its returned result. If it does, the incomplete answer by the

server is caught with 100% certainty. Otherwise, the client believes that the server returns incomplete answer with a probability. In particular, the probability pr_R of catching the incomplete frequent itemsets F^S of recall R by ℓ evidence frequent itemsets (EFs) is $pr_R = 1 - R^\ell$. Clearly, to satisfy (α_1, β_1)-completeness (i.e., $pr_R \geq \beta_1$), it must be true that $\ell \geq \lceil log_{\alpha_1}(1 - \beta_1) \rceil$. Further analysis can show that to catch a server that fails to return a small fraction of frequent itemsets with high completeness probability does not need large number of EFs. For instance, when $\alpha_1 = 0.95$ and $\beta_1 = 0.95$, only 58 EFs are sufficient. Apparently the number of required EFs is independent from the size of the dataset as well as the number of real frequent itemsets. Therefore our verification approach is especially suitable for large datasets.

We propose the *MiniGraph* approach to construct EFs. The basic idea of the *MiniGraph* approach is to construct itemsets that are guaranteed to be infrequent in the original dataset D. To construct these itemsets quickly without doing any mining, we construct the itemsets that contain at least one infrequent 1-itemset. The *MiniGraph* approach consists of the following steps:

Step 1: Pick the shortest infrequent itemset (can be 1-itemset) of the largest support as I_s.

Step 2: Find transactions $D_s \subseteq D$ that contain I_s. Construct the *MiniGraph* G from D_s, with the root of G representing I_s, and each non-root node in G representing a transaction in D_s. There is an edge from node N_i to node N_j if the transaction of node N_j is the maximum subset of the transaction of node N_i in D (i.e., no other transactions in D that contain the transaction of node N_i).

Step 3: Mark all nodes at the second level of G as candidates. For each candidate, all of its subset itemsets that contain I_s will be picked as EFs. If the total number of candidates is less than $\ell = \lceil log_{\alpha_1}(1 - \beta_1) \rceil$, we add the next infrequent 1-itemset of the largest frequency as another I_s, and repeat Step 1 - 3, until we either find ℓ EFs or there is no infrequent 1-itemset left.

Step 4: For each EF, construct $(min_{sup} - s)$ copies as artificial transactions, where s is the support of EF in D.

The time complexity of the *MiniGraph* approach is $O(|D|)$.

5 Construction of Evidence Infrequent Itemsets (EIs)

Our basic idea of correctness verification is that the client uses a set of infrequent itemsets as the *evidence*, and checks whether the server returns any evidence infrequent itemset. If it does, the incorrect answer by the server is caught with 100% certainty. Otherwise, the client believes that the server returns the incorrect answer with a probability. In particular, the probability pr_P of catching the incorrect frequent itemsets with precision P by using ℓ EIs is $pr_P = 1 - P^\ell$. To satisfy (α_2, β_2)-correctness (i.e., $pr_P \geq \beta_2$), it must satisfy that $\ell \geq \lceil log_{\alpha_2}(1 - \beta_2) \rceil$. As pr_P and pr_R (Section 4) are measured in the similar way, we have the same observation of the number of EIs as the number of EFs.

Our EI construction method will identify a set of real frequent itemsets and change them to be infrequent by removing items from the transactions that

contain them. Our goal is to minimize the number of items that are removed. Next, we explain the details.

Step 1: Pick Evidence Infrequent Itemsets (EIs). First, we exclude items that are used as I_s for EF construction from the set of 1-itemset candidates. This ensures that no itemset will be required to be EI and EF at the same time. Second, we insert all infrequent 1-itemsets into the evidence repository $\mathcal{R}$. If $|\mathcal{R}| \geq \ell = \lceil log_{\alpha_2}(1-\beta_2) \rceil$, we terminate EI construction. Otherwise, we compute h, the minimal value to make $\binom{m-|\mathcal{R}|}{h} \geq \ell - |\mathcal{R}|$, where m is the number of unique items in D. Third, we compute k, the minimal value to make $\binom{k}{h} \geq \ell - |\mathcal{R}|$. We pick the first k *frequent* 1-itemsets S following their frequency in ascending order, and construct all h-itemset candidates S_h that contain h items from S. The h-itemset candidates of non-zero support in D will be inserted into $\mathcal{R}$. To efficiently find the itemset I that has non-zero support in D, we make use of a simpler version of the FP-tree [7] to store D in a compressed way. More details of this data structure is omitted due to space limit.

Step 2: Pick Transactions for Item Removal. We aim at transforming those frequent EIs (i.e., artificial infrequent EIs) picked by Step 1, notated as AI, to be infrequent. To achieve this, we pick a set of transactions $D' \subseteq D$, so that for each frequent itemset $I \in AI$, $sup_{D'}(I) \geq sup_D(I) - min_{sup} + 1$.

Step 3: Pick Item Instances for Removal. We decide which items in the transactions picked by Step 2 will be removed. To minimize the total number of removed items, we prefer to remove the items that are shared among patterns in AI. Therefore, we sort items in AI by their frequency in AI in descending order. We follow this order to pick items to be removed.

The time complexity of the EI construction approach is $O(|EI||D| + k!|T|)$, where k is the number of frequent 1-itemsets for construction of EIs, and T is the FP-tree constructed for checking the existence of itemsets in D. Normally $k << m$, where m is the number of items in D, and $|T| << |D|$.

6 Post-processing

There are two types of side effects by introducing EFs and EIs that need to be compensated: (1) EFs may introduce artificial frequent itemsets that do not exist in D; and (2) EIs may make some real frequent itemsets disappear. Removal of artificial frequent itemsets is straightforward. As the client is aware of the seed item I_s that is contained in all EFs, it only needs to remove all the returned frequent itemsets that contain I_s. To recover missing real frequent itemsets, the client maintains locally all AIs when it constructs EIs. During post-processing, the client adds these AIs back to F^S as frequent itemsets.

7 Experiments

In this section, we experimentally evaluate our verification methods. All experiments were executed on a Macbook Pro machine with 2.4GHz CPU, 4GB

memory, running Mac OS X 10.7.3. We implemented a prototype of our algorithm in Java.

We evaluated our algorithm on two type of datasets: (1) the *dense* dataset in which most of transactions are of similar length, and contain $> 75\%$ of items; and (2) the *sparse* dataset in which the transactions are of skewed length distribution. We use the *NCDC* dataset[1] (500 items, 365 transactions) as the dense dataset, and the *Retail* dataset[2] (16470 items, 88162 transactions) as the sparse dataset. Due to its density/sparsity, *NCDC* dataset has a large number of frequent 1-itemsets, while *Retail* dataset has a large number of infrequent 1-itemsets. We use the Apriori algorithm [1], a classic frequent itemset mining algorithm, as the main mining algorithm. We use the implementation of Apriori algorithm available at http://www.borgelt.net/apriori.html.

Robustness. We measure the robustness of our probabilistic approach by studying the probability that the incorrect/incomplete frequent itemsets can be caught by using artificial EIs/EFs. We use the *Retail* dataset and vary α_1 and α_2 values to control the amount of mistakes that the server can make on the mining result. For each α_1 (α_2, resp.) value, we randomly modify $(1 - \alpha_1)$ ($(1 - \alpha_2)$, resp.) percent of frequent (infrequent, resp.) itemsets (including both true and artificial ones) to be infrequent (frequent, resp.). Then with various β_1 and β_2 values, we construct artificial tuples to satisfy (α_1, β_1)-completeness and (α_2, β_2)-correctness. Detection of any missing EF or the presence of any EIs will be recorded as a successful trial of catching the server. We repeat 1,000 times and record the percentage of trials (as *detection probability*) that the server is caught, with $\alpha_1, \alpha_2 \in [0.7, 0.9]$ and $\beta_1, \beta_2 \in [0.7, 0.9]$. It shows that the detection probability for the completeness and correctness verification is always higher than β_1 and β_2 respectively. This proves the robustness of our probabilistic approach. The results are omitted due to limited space.

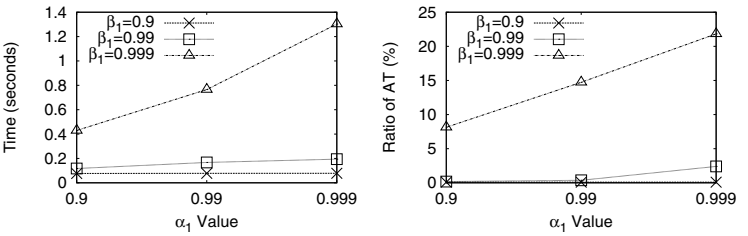
Completeness Verification. First, we measure the EF construction time for various α_1 and β_1 values. The result in Figure 1 (a) shows that EF construction time grows when α_1 and β_1 grow, since the *MiniGraph* approach has to search for more I_s to construct more EFs for higher completeness guarantee.

Second, we measure the amount of inserted artificial transactions and compare it with the size of the database. In particular, let t be the number of artificial transactions to be inserted, we measure the ratio $r = \frac{t}{m}$, where m is the number of real transactions in D. As shown in Figure 1 (b), for *Retail* dataset, the inserted artificial transactions only take a small portion of the original database. For example, when $\beta_1 \leq 0.99$, the ratio is less than 3%. Even for large values such as $\alpha_1 = \beta_1 = 0.999$, the ratio is no more than 25% .

Correctness Verification. First, we measure the EI construction time on *NCDC* dataset. The performance result is shown in Figure 2 (a). It is not

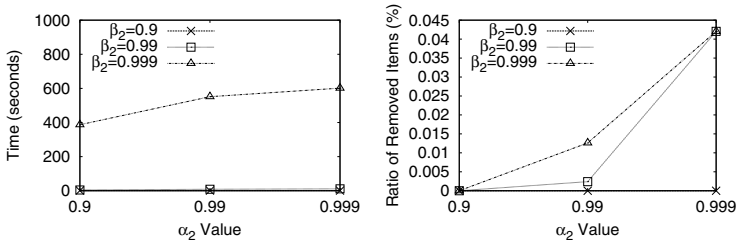
[1] National Climatic Data Center of U.S. Department of Commerce: `http://lwf.ncdc.noaa.gov/oa/climate/rcsg/datasets.html`

[2] Frequent Itemset Mining Dataset Repository: `http://fimi.ua.ac.be/data/`.



(a) EF construction time (b) Ratio of Artificial Transactions

Fig. 1. Ratio of Artificial Transactions and Mining Overhead (*Retail* dataset)



(a) EI Construction Time (b) Ratio of Removed Items

Fig. 2. EI Construction ($NCDC$ dataset)

surprising that it needs more time to construct EIs for higher α_2 and β_2 values. With a closer look of the result, when $\beta_2 = 0.9$ and 0.99, EI construction is very fast (no more than 1 second), since all EIs are real infrequent itemsets and there is no need to remove any item. However, when β_2 grows to 0.999, the construction time jumps to 400 - 600 seconds, since now the algorithm needs to find frequent itemset candidates to be EIs as well as the items to be removed. We also measure the EI construction time of *Retail* dataset. It does not increase much when β_2 increases from 0.9 to 0.999, since all EIs are real infrequent 1-itemsets.

Second, we measure the amount of item instances that are removed by EI construction. In particular, let t be the number of item instances to be removed, we measure the ratio $r = \frac{t}{|D|}$. The result of $NCDC$ dataset is shown in Figure 2 (b). It can be seen that the number of item instances to be removed is a negligible portion (no more than 0.045%) of $NCDC$ dataset. There is no item that is removed from *Retail* dataset, as it has a large number of infrequent 1-itemsets, which provides sufficient number of EI candidates. This shows that we can achieve high correctness guarantee to catch small errors by slight change of the dataset.

8 Conclusion

In this paper, we present our methods that verify the correctness and completeness of outsourced frequent itemset mining. We propose a lightweight verification approach that constructs evidence (in)frequent itemsets. In particular, we remove a small set of items from the original dataset and insert a small set of artificial transactions into the dataset to construct evidence (in)frequent itemsets. Our experiments show the efficiency and effectiveness of our approach. An interesting direction to explore is to design verification approaches that can provide *deterministic* correctness/completeness guarantee without extensive computational overhead.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB (1994)
2. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. Journal of ACM 45, 501–555 (1998)
3. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
4. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal of Computing 18, 186–208 (1989)
5. Goodrich, M.T., Nguyen, D., Ohrimenko, O., Papamanthou, C., Tamassia, R., Triandopoulos, N., Lopes, C.V.: Efficient verification of web-content searching through authenticated web crawlers. PVLDB 5(10), 920–931 (2012)
6. Hacigümüş, H., Iyer, B., Li, C., Mehrotra, S.: Executing sql over encrypted data in the database-service-provider model. In: SIGMOD (2002)
7. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Mining Knowledge and Discovery 8(1), 53–87 (2004)
8. Liu, R., Wang, H., Monreale, A., Pedreschi, D., Giannotti, F., Guo, W.: Audio: An integrity auditing framework of outlier-mining-as-a-service systems. In: ECML/PKDD (2012)
9. Pang, H., Jain, A., Ramamritham, K., Tan, K.-L.: Verifying completeness of relational query results in data publishing. In: SIGMOD (2005)
10. Tai, C.-H., Yu, P.S., Chen, M.-S.: k-support anonymity based on pseudo taxonomy for outsourcing of frequent itemset mining. In: SIGKDD (2010)
11. Wang, W.H., Lakshmanan, L.V.S.: Efficient secure query evaluation over encrypted xml databases. In: VLDB (2006)
12. Wong, W.K., Cheung, D.W., Hung, E., Kao, B., Mamoulis, N.: Security in outsourcing of association rule mining. In: VLDB (2007)
13. Wong, W.K., Cheung, D.W., Kao, B., Hung, E., Mamoulis, N.: An audit environment for outsourcing of frequent itemset mining. In: PVLDB, vol. 2, pp. 1162–1172 (2009)
14. Zhu, Y., Wang, H., Hu, Z., Ahn, G., Hu, H.: Zero-knowledge proofs of retrievability. Science China Information Sciences 54(8), 1608–1617 (2011)

An Approach to Select Cost-Effective Risk Countermeasures

Le Minh Sang Tran[1], Bjørnar Solhaug[2], and Ketil Stølen[2,3]

[1] University of Trento, Italy
[2] SINTEF ICT, Norway
[3] Department of Informatics, University of Oslo, Norway
tran@disi.unitn.it, {bjornar.solhaug,ketil.stolen}@sintef.no

Abstract. Security risk analysis should be conducted regularly to maintain an acceptable level of security. In principle, all risks that are unacceptable according to the predefined criteria should be mitigated. However, risk mitigation comes at a cost, and only the countermeasures that cost-efficiently mitigate risks should be implemented. This paper presents an approach to integrate the countermeasure cost-benefit assessment into the risk analysis and to provide decision makers with the necessary decision support. The approach comes with the necessary modeling support, a calculus for reasoning about the countermeasure cost and effect, as well as means for visualization of the results to aid decision makers.

1 Introduction

Security risk analysis concludes with a set of recommended options for mitigating unacceptable risks [8]. The required level of security and the acceptable level of risk should be defined by the risk criteria. However, deciding which countermeasures to eventually implement depends also on the trade-off between benefit and spending. No matter the criteria and the mitigating effect of the countermeasures, risk mitigation should ensure return on investment in security [2]. Currently there exists little methodic support for systematically capturing and analyzing the necessary information for such decision making as an integrated part of the security risk analysis process.

The contribution of this paper is an approach to integrate the assessment of countermeasures and their cost and effect into the risk analysis process. The approach comes with the necessary modeling support, a calculus for reasoning about risks, countermeasures, costs and effects within the risk models, as well as support for decision making. A formal foundation is provided to ensure rigorous analysis and to prove the soundness of the calculus. The approach is generic in the sense that it can be instantiated by several established risk modeling techniques. The reader is referred to the full technical report [12] for the formal foundation, the soundness proofs and other details. The report demonstrates the instantiation in CORAS [9] with an example from the eHealth domain.

In Section 2 we present our approach, including the method, the modeling support and the analysis techniques. Section 3 gives a small example. Related work is presented in Section 4, before we conclude in Section 5.

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 266–273, 2013.

2 Our Approach

Our approach (see Fig. 1) takes a risk model resulting from a risk assessment and the associated risk acceptance criteria as input, and delivers a recommended countermeasure alternative as output. Hence, the approach assumes that the risk assessment has already been conducted, *i.e.* that risks have been identified, estimated and evaluated and that the overall risk analysis process is ready to proceed with the risk treatment phase. We moreover assume that the risk analysis process complies with the ISO 31000 risk management standard [8], in which risk countermeasure is the final phase. Our process consists of three main steps.

In Step 1, the risk model is annotated with relevant information including the countermeasures, their cost, their reduction effect (*i.e.* effect on risk value), as well as possible effect dependencies (*i.e.* countervailing effects among countermeasures). In Step 2, we perform countermeasure analysis by enumerating all countermeasure alternatives (*i.e.* combinations of countermeasures to address risks) and reevaluating the risk picture for each alternative. This analysis makes use of the annotated risk model and a calculus for propagating and aggregating the reduction effect and effect dependency along the risk paths of the model. Step 3 performs synergy analysis for selected risks based on decision diagrams. The output is a recommended countermeasure alternative.

Fig. 2 presents the underlying concepts of our approach. A *Risk Model* is a structured way of representing unwanted incidents, their causes and consequences using graphs, trees or block diagrams. An unwanted incident is an event that harms or reduces the value of an asset, and a risk is the likelihood of an unwanted incident and its consequence for a specific asset [8]. A *Countermeasure* mitigates risk by reducing its likelihood and/or consequence. The *Expenditure* includes the expenditure of countermeasure implementation, maintenance and so on for a defined period of time. The *Effects Relation* captures the extent to which a countermeasure mitigates risks. The *Effects Relation* could be the reduction of likelihood, and/or the reduction of consequence of a risk. The *Dependency relation* captures the countervailing effect among countermeasures that must be taken into account in order to understand the combined effect of identified countermeasures. The *Calculus* provides a mechanism to reason about the annotated risk model. Using the *Calculus*, we can perform countermeasure analysis on annotated risk models to calculate the residual risk value for each individual risk. A *Decision Diagram* facilitates the decision making process based on the countermeasure analysis.

As already explained, the input required by our approach is the result of a risk assessment in the form of a risk model, and the corresponding risk

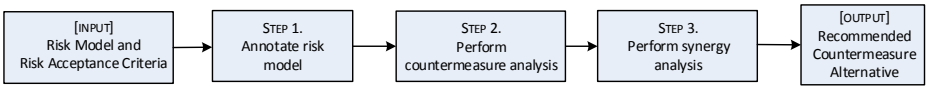


Fig. 1. Three-steps approach

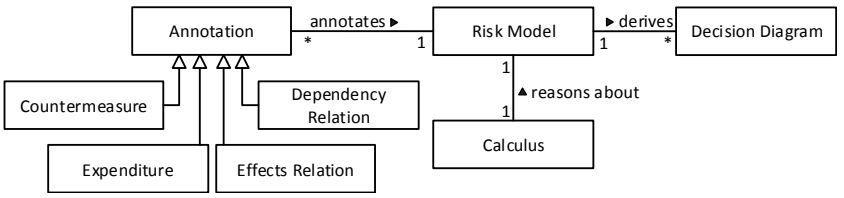


Fig. 2. Conceptual model

acceptance criteria. To ensure that our approach is compatible with established risk modeling techniques, we only require that the risk model can be understood as a risk graph. A risk graph [3] is a common abstraction of several established risk modeling techniques such as Fault Tree Analysis (FTA), Event Tree Analysis (ETA), Attack Trees, Cause-Consequence Diagrams, Bayesian networks, and CORAS risk diagrams. Hence, our approach complies with these risk modeling techniques, and can be instantiated by them.

A risk graph is a finite set of vertices and relations (see Fig. 3(a)). Each vertex v represents a threat scenario, *i.e.* a sequence of events that may lead to an unwanted incident, and can be assigned a likelihood f, and a consequence co. The likelihood can be either probabilities or frequencies, but here we use only the latter. A *leads-to* relation from v_1 to v_2 means that the former threat scenario may lead to the latter. The positive real numbers decorating the relations capture statistical dependencies between scenarios, such as conditional probabilities.

2.1 Detailing of Step 1 – Annotate Risk Model

This step is to annotate the input risk model with required information for further analysis. There are four types of annotation as follows.

Countermeasures are represented as rectangles. In Fig. 3(b) there is one countermeasure, namely *cm*. An *expenditure* is expressed within square brackets following the countermeasure name (e in Fig. 3(b)). This is an estimation of the expense to ensure the mitigation of countermeasure including the expense of implementation, maintenance, and so on. An *effects relation* is represented by

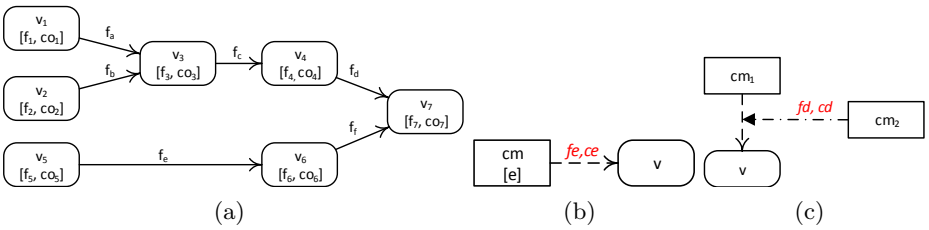


Fig. 3. A risk graph (a) and its extended annotations: Effect relation (b), and Dependency relation (c)

a dashed arrow decorated by two numbers (*fe* and *ce* in Fig. 3(b)). It captures the mitigating effect of a countermeasure in terms of reduced frequency (*i.e. frequency effect - fe*), reduced consequence (*i.e. consequence effect - ce*), or both. Both fe and *ce* are relative percentage values, *i.e.* $fe, ce \in [0, 1]$. A *dependency relation* is represented by a dash-dot arrow with solid arrowhead decorated by two numbers, namely a *frequency dependency - fd* and a *consequence dependency - cd* as illustrated in Fig. 3(c). A dependency relation captures the impact of a countermeasure on the effect of another when both are implemented. In Fig. 3(c) the *fd* impacts *fe* while the *cd* impacts *ce*. Both *fd* and *cd* are relative percentage values, *i.e.* $fd, cd \in [0, 1]$.

2.2 Detailing of Step 2 – Countermeasure Analysis

The countermeasure analysis is conducted for every risk of the annotated risk model. The analysis enumerates all possible countermeasure combinations, called *countermeasure alternatives* (or *alternatives* shortly), and evaluates the residual risk value (*i.e.* residual frequency and consequence value) with respect to each alternative to determine the most effective one. The residual risk value is obtained by propagating the reduction effect along the risk model.

From the leftmost threat scenarios (*i.e.* scenarios that have only outgoing *leads-to* relations), frequencies assigned to threat scenarios are propagated to the right using the formal calculus. The reader is referred to [12] for the full formal calculus and for the soundness proofs. During the propagation, frequencies assigned to *leads-to* relations, reduction effects, and effect dependencies are taken into account. Finally, the propagation stops at the rightmost threat scenarios (*i.e.* scenarios that have only incoming *leads-to* relations). Based on the results from the propagation, the residual risk value is computed.

A *Decision Diagram*, as depicted in Fig. 4 for two different risks, is a directed graph used to visualize the outcome of a countermeasure analysis. A node in the diagram represents a *risk state* which is a triplet of a likelihood, a consequence, and a countermeasure alternative for the risk being analyzed. The frequency and consequence are the X and Y coordinates, respectively, of the node. The countermeasure alternative is annotated on the path from the *initial state* S_0 (representing the situation where no countermeasure has yet been applied). Notice that we ignore all states whose residual risks are greater than those of S_0 since it is useless to implement such countermeasures.

2.3 Detailing of Step 3 – Synergy Analysis

The synergy analysis aims to recommend a cost-effective countermeasure alternative for mitigating all risks. It is based on the decision diagrams of individual risks (generated in Step 2), the risk acceptance criteria, and the overall cost (OC) of each countermeasure alternative. The OC is calculated as follows:

$$\mathrm{OC}(ca) = \sum_{r \in R} \mathrm{rc}(r) + \sum_{cm \in ca} \mathrm{cost}(cm) \tag{1}$$

Here, *ca* is a countermeasure alternative; R is the set of risks; rc() is a function that yields the loss (in monetary value) due to the risk taken as argument (based on its likelihood and consequence); cost() is a function that yields the expenditure of the countermeasure taken as argument.

The synergy analysis is decomposed into the following three substeps:

STEP 3A *Identify countermeasure alternatives:* Identify the set of countermeasure alternatives CA for which all risks are acceptable with respect to the risk acceptance criteria. CA could be identified by exploiting decision diagrams.

STEP 3B *Evaluate countermeasure alternatives:* If there is no countermeasure alternative for which all risks fulfill the risk acceptance criteria ($CA = \emptyset$), do either of the following:

- identify new countermeasures and go to STEP 1, or
- adjust the risk acceptance criteria and go to STEP 3A.

Otherwise, if there is at least one such countermeasure alternative ($CA \neq \emptyset$), calculate the overall cost of each $ca \in CA$.

STEP 3C *Select cost-effective countermeasure alternative:* If there is at least one countermeasure $ca \in CA$ for which OC(ca) is acceptable (for the customer company in question) select the cheapest and terminate the analysis. Otherwise, identify more (cheaper and/or more effective) countermeasures and go to STEP 1.

The above procedure may of course be detailed further based on various heuristics. For example, in many situations, with respect to STEP 3A, if we already know that countermeasure alternative *ca* is contained in CA then we do not have to consider other countermeasure alternatives ca' such that $ca \subseteq ca'$. However, we do not go into these issues here.

3 Example

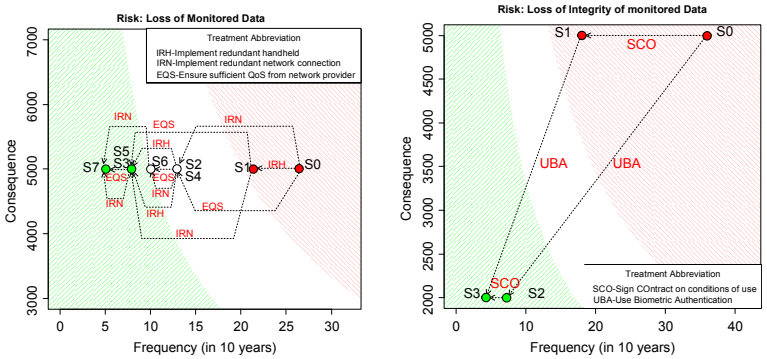
In the following we give a small example of the synergy analysis based on our eHealth assessment [12]. The scenario is on remote patient monitoring, where one of the identified risks is *loss of monitored data* (LMD). Table 1 is input from STEP 2, namely the result of the analysis of seven treatment alternatives given three identified treatments. The corresponding decision diagram is depicted in Fig. 4(a). The shaded area to the lower left are the acceptable risks levels, whereas the upper right are the unacceptable levels. Notice that while the treatment alternatives for LMD reduce only the consequence, some of the alternatives for *loss of integrity of monitored data* (LID) also reduce the frequency.

The results of the synergy analysis of three risks are depicted in Table 2. Their respective treatment alternatives that yield acceptable risk levels are shown in the middle, whereas their combinations are shown in the first column. The third column shows the overall costs as calculated in STEP 3. If also the costs are acceptable, the cheapest alternative should be selected.

Table 1. Analysis for the risk *Loss of monitored data*

Each treatment alternative S is shown in the first column (*Risk State*) followed by its combination of treatments. The *Frequency* column is the number of occurrences in ten years. Both *Frequency* and *Consequence* columns are valued after considering the treatments.

Risk State	Ensure sufficient QoS from network provider	Implement Redundant Network connection	Implement Redundant Handheld	Frequency	Consequence
S0				26.4	5000
S1	•			21.36	5000
S2		•		12.96	5000
S3	•	•		7.92	5000
S4			•	12.96	5000
S5	•		•	7.92	5000
S6		•	•	10.08	5000
S7	•	•	•	5.04	5000



(a) (b)

Fig. 4. Decision diagrams of two risks in the eHealth scenario

4 Related Work

In risk management, decision on different risk mitigation alternatives has been emphasized in many studies [6,10,11]. The guideline in [11] proposes cost-benefit analysis to optimally allocate resources and implement cost-effective controls after identifying all possible countermeasures. This encompasses the determination of the impact of implementing (and not implementing) the mitigations, and the estimated costs of them. Another guideline [6] provides a semi-quantitative risk assessment. The probability and impact of risks are put into categories which are assigned scores. The differences between the total score for all risks before and after any proposed risk reduction strategy relatively show the efficiency among strategies, and effectiveness of their costs. It also suggests that the economic costs for baseline risks should be evaluated. However, the proposed methods for conducting the evaluation have not been designed to assess cost of treatments, but rather cost of risks.

Table 2. Results from synergy analysis of three risks

	Individual Risk			
Treatment Alternative	LID	LMD	DAS	Overall Cost
{UBA,SCO,IRH,IRN,USW}	S3	S3	S3	101740
{UBA,SCO,IRH,IRN,EQS,USW}	S3	S7	S3	102340
{UBA,IRH,IRN,USW}	S2	S3	S3	104500
{UBA,IRH,IRN,EQS,USW}	S2	S7	S3	105100
{UBA,SCO,IRH,IRN}	S3	S3	S2	108740
{UBA,SCO,IRH,IRN,EQS}	S3	S7	S2	109340
{UBA,IRH,IRN}	S2	S3	S2	111500
{UBA,IRH,IRN,EQS}	S2	S7	S2	112100

Norman [10] advocates the use of Decision Matrix to agree on countermeasure alternative. A Decision Matrix is a simple spreadsheet consisting of countermeasures and their mitigated risks. The approach, however, is not clearly defined, and the spreadsheets are complicated to implement and follow. Meanwhile, our proposal is graphical and backed up with a formal definition and reasoning. Butler [4] proposes the Security Attribute Evaluation Method (SAEM) to evaluate alternative security designs in four steps: benefit assessment, threat index evaluation, coverage assessment, and cost analysis. This approach, however, focuses mostly on the consequence of risks rather than cost of countermeasures, whereas our approach captures both.

Chapman and Leng [5] describe a decision methodology to measure the economic performance of risk mitigation alternatives. It focuses on the cost-difference aspect, but does not consider the benefit-difference (*i.e.* level of risks reduced) among alternatives.

Houmb et al. [7] introduce SecInvest, a security investment support framework which derives a security solution fitness score to compare alternatives and decide whether to invest or to take the associated risk. SecInvest relies on a trade-off analysis which employs existing risk assessment techniques. SecInvest ranks alternatives with respect to their cost and effect, trade-off parameters, and investment opportunities. However, this approach does not provide a systematic way to assess the effects of alternatives on risks, and does not take into account the dependency among countermeasures in an alternative.

Beresnevichiene et al. [1] propose a methodology incorporating a multi-attribute utility evaluation and mathematical system modeling to assist decision makers in the investment on security measures. It can be employed in existing risk assessment methods, including ours, to evaluate the residual risk.

5 Conclusion

We have presented an approach to select a cost-effective countermeasure alternative to mitigate risks. The approach requires input in the form of risk models represented as risk graphs. The approach analyses risk countermeasures with respect to different aspects such as the mitigating effect, how countermeasures affect others, and how much countermeasures cost. We have developed a formal

calculus extending the existing calculus for risk graphs. The extended calculus can be used to propagate likelihoods and consequences along risk graphs, thereby facilitating a quantitative countermeasure analysis on individual risks, and a synergy analysis on all the risks. The outcome is a list of countermeasure alternatives quantitatively ranked according to the their overall cost. These alternatives are represented not only in tabular format, but also graphically in the form of decision diagrams. The approach is generic in the sense that it can be instantiated by several existing risk assessment techniques.

Acknowledgments. This work has received funding from the European Commission via the NESSoS NoE (256980) and the RASEN project (316853), and from the Research Council of Norway via the DIAMONDS project (201579/S10).

References

1. Beresnevichiene, Y., Pym, D., Shiu, S.: Decision support for systems security investment. In: Network Operations and Management Symposium Workshops (NOMS 2010), pp. 118–125. IEEE/IFIP (2010)
2. Birch, D.G., McEvoy, N.A.: Risk analysis for information systems. Journal of Information Technology 7, 44–53 (1992)
3. Brændeland, G., Refsdal, A., Stølen, K.: Modular analysis and modelling of risk scenarios with dependencies. J. Syst. Softw. 83(10), 1995–2013 (2010)
4. Butler, S.A.: Security attribute evaluation method: a cost-benefit approach. In: Proceedings of the 24th International Conference on Software Engineering (ICSE 2002), pp. 232–240. ACM (2002)
5. Chapman, R.E., Leng, C.J.: Cost-effective responses to terrorist risks in constructed facilities. Technical report, U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology (2004)
6. Risk Characterization of Microbiological Hazards in Food: Guidelines. Microbiological Risk Assessment Series No. 17. Food and Agriculture Organization of the United Nations (FAO)/World Health Organization (WHO) (2009)
7. Houmb, S.H., Ray, I., Ray, I.: SecInvest: Balancing security needs with financial and business constraints. In: Dependability and Computer Engineering, pp. 306–328. IGI Global (2012)
8. International Organization for Standardization. ISO 31000 Risk management – Principles and guidelines (2009)
9. Lund, M.S., Solhaug, B., Stølen, K.: Model-Driven Risk Analysis: The CORAS Approach. Springer (2011)
10. Norman, T.L.: Risk Analysis and Security Countermeasure Selection. CRC Press (2010)
11. Stoneburner, G., Goguen, A., Feringa, A.: Risk Management Guide for Information Technology Systems. National Institute of Standards and Technology, pp. 800–830. NIST Special Publication 800-30 (2002)
12. Tran, L.M.S., Solhaug, B., Stølen, K.: An approach to select cost-effective risk countermeasures exemplified in CORAS. Technical Report A24343, SINTEF ICT (2013)

Enhance Biometric Database Privacy: Defining Privacy-Preserving Drawer Size Standard for the Setbase

Benjamin Justus[1], Frédéric Cuppens[1], Nora Cuppens-Boulahia[1], Julien Bringer[2], Hervé Chabanne[2], and Olivier Cipiere[2]

[1] Lab-STICC, Télécom Bretagne, Cesson Sévigné, France
[2] Morpho, Paris, France

Abstract. Shamir proposed the setbase approach as a means of improving security and privacy of the traditional biometric system. In this paper, we propose privacy-preserving drawer size standards for the biometric setbase. The proposal incorporates database privacy metrics such as k-anonymity and l-diversity into the definition of privacy-preserving drawer size standard for the biometric setbase. We also empirically evaluate the system reliability of the prototype setbase for the purpose of studying the trade-off values between the level of privacy protection and the level of system security.

Keywords: database fragmentation, setbase, k-anonymity, l-diversity, biometric privacy.

1 Introduction

The setbase approach was proposed by Adi Shamir [6] (see also [2]) as a means of improving security and privacy of the traditional biometric system. The traditional biometric database uses a one-to-one data linking between biometric data stored in the biometric database and personal information stored in the identity database. The setbase approach depends on the creation of drawers in which the biometric and personal data are stored. The linking between the identity and the corresponding biometric data takes places at the level of linking identical drawer IDs in the databases. Since each drawer contains multiple records, the identification of an identity is blurred which leads to privacy protection of an individual. With the number of drawers sufficiently large, the probability of an identity theft can be made negligible. Furthermore, the lack of cryptographic mechanisms in a setbase can be seen as another privacy virtue because we do not want to place too much trust in a single entity as for instance the owner of the cryptographic keys.

Given the theoretic interest of Shamir's proposal, there have been hitherto no studies in the literature on how practical shamir's scheme is. Specifically, one would like to know at least in theory:

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 274–281, 2013.

1. Set of metrics capable of measuring the privacy level of the setbase approach.
2. Value ranges of the important setbase system parameters, such as the number of drawers and possible sizes of a drawer.
3. Impact of the setbase parameters on the level of database privacy.

The importance of choosing the appropriate drawer sizes in a setbase can be seen in the context of preserving privacy of individuals in a biometric database. To make things concrete, we explain the concept using the crime investigation example. In such a scenario, the investigator has gathered some generic information such as the sex, the age group, and the domicile region of a suspect. Furthermore, the investigator based on the biometric data (e.g. fingerprints collected at the crime scene) is able to locate the drawer in which a suspect is located. To identify the person, he must retrieve the identity of a suspect. If the drawer size is small, the investigator would have a little trouble in sieving out entries inside the drawer that satisfy his searching criteria. In the extreme case of a traditional biometric database (one-to-one biometric and identity association), the identity of a suspect would be revealed once a successful bio and identity matching is achieved. The increase of the drawer size would make an investigator's search more difficult. Thus strong privacy in a setbase requires a large drawer size. The downside of a large drawer size is that it would expose the system to a higher chance of drawer ID collision attacks (see section 3). The appropriate choices of the drawer size for a setbase require a judicious way of balancing between privacy and security.

1.1 Our Contribution

The starting point of our project is a prototype implementation of the biometric setbase in the context of studying its suitability for the integration with the French national identity cards. There had been many propositions in the French National Assembly advocating the protection of sensitive information such as those contained in the national identity cards [1,5]. The main goal of our project is on the one hand to measure the feasibility of the setbase approach for a real world implementation, and on the other hand to quantify as accurately as possible how much privacy is enhanced by using the setbase approach.

In order to measure the privacy level of the setbase, we had to develop a drawer size standard that depends on a set of well known database privacy metrics. The current paper relies on the concept of k-anonymity and l-diversity as a means of measuring the privacy level of data tables contained in the database. Our study is based on the French population data released from National Institute of Statistics and Economic Studies. The present study shows that the drawer size standard proposed in the paper permits feasible setbase solution for the integration with the traditional biometric database currently in use.

1.2 Paper Outline

The paper is organized as follows. The proposed privacy-preserving drawer size formula is presented in section 2. We evaluate the prototype system performance

empirically in section 3. Section 4 provides a detailed analysis of the trade-off between security and privacy when deciding the drawer size.

1.3 Notation

We use the following notations throughout the paper:

NT **:** the number of drawers that is maintained by the system
TT **:** the size of a drawer that is maintained by the system

2 Database Tracing Modeling

The aim of this section is to provide some reasonings behind the privacy-preserving drawer size formula:

$$TT = Tol(k) \cdot E(X)_{\mathbf{P}} + N_{expire} \tag{1}$$

where $Tol(k)$ is a linear function depending on the k-anonymity parameter (see [4,3] for the definition). N_{expire} is a fixed number that represents the number of expired entries in the drawer. It is a numerical parameter belonging to the drawers due to the definite expiration date of the biometric data. The value N_{expire} is usually determined by the relevant government policies in the public domains. The random variable X represents the number of sequential searches required before one hits an identity record that satisfies the specified sex, age group, and domicile region criteria. The random variable X assumes a geometric distribution. And the expected value $E(X)_{\mathbf{P}}$ is calculated from the default population distributions $\mathbf{P_s}$, $\mathbf{P_a}$,$\mathbf{P_d}$ on the sex, age group, and domicile region attributes inside the biometric identity database.

We first introduce the required variables in modeling the identity tracing procedure during the biometric-identity matching procedure. The deduction of (1) takes place in section 2.3.

2.1 Database Related and Search Variables

The database related variables and the search variable are discrete random variables. They are summarized in Table 1.

The random variable S represents the sex of an identity. It takes on two values 1 (male), and 2 (female) with the corresponding probabilities $\{p_1^s, p_2^s\}$. The random variable A represents the age group to which an identity belongs. In this paper, A has 8 possible values with the corresponding probabilities $\{p_1^a, p_2^a, ..., p_8^a\}$. The values represent 8 different age groups that covers the age span of an entire population. The values $\{p_1^a, p_2^a, ..., p_8^a\}$ are taken from the current census data as shown in table 2.[1]

[1] National Institute of Statistics and Economics Studies, `http://www.insee.fr/en/themes/tableau.asp`

Table 1. Random Variables

Random Variable	Description
S	Sex: 1 (M), 2 (F)
A	Age Group: $1, 2, ..., 8$
D	Domicile Region: $1, 2, ..., 101$
X	The number of searches needed before the condition $S = i$ **and** $A = j$ **and** $D = k$ is satisfied, for fixed i, j, k

Table 2. Age Distribution

Age Group	Probability
< 15 ans	0.185
$15 - 24$ ans	0.123
$25 - 34$ ans	0.123
$35 - 44$ ans	0.134
$45 - 54$ ans	0.136
$55 - 64$ ans	0.127
$65 - 74$ ans	0.081
> 75 ans	0.09

The random variable D represents the domicile of an identity. This variable has a range of 1 to 101 with the corresponding probabilities $\{p_1^d, p_2^d, ..., p_{101}^d\}$. This is a French scenario because currently there exists 101 French departments.[2] The random variable X represents the number of sequential searches required before one hits a personal record in the identity database that satisfies the specified sex, age group, and domicile region criteria. The search random variable X has a value range from 1 to the size of a drawer TT, and it assumes a discrete geometric distribution.

2.2 Anonymity Related Variables

The variables described in this section are related to the quantization of anonymity levels for a specific criterion. To guarantee the k-anonymity requirement for a specific criterion, we require that each specified criterion in the released table has at least k occurrences [3]. Table 3 summarizes the variables.

$Tol_{s,a,d}$**:** the minimum standard needed for achieving indistinguishability among people who satisfy the respective criterion. In order to achieve k-anonymity, one should have $Tol_s \geq 2k, Tol_a \geq 8k, Tol_d \geq 101k$.

ω_i**:** the weights associated with the variables Tol_s, Tol_a and Tol_d respectively. They represent the reliability of information held by an investigator. The weight ω_i is normally set at 1 unless there are reliability issues on the information, which would lead ω_i to a value strictly less than 1. The value $\omega_i = 0$ indicates that the specific criterion is not used in the investigation.

Tol**:** the global value that represents the tolerance for all the criteria used in the investigation. It is defined as

$$Tol = \omega_1 Tol_s + \omega_2 Tol_a + \omega_3 Tol_d. \tag{2}$$

We explain below the lower bound associated with each anonymity related Tols (see the variable description above). In the course of sequential searching for a

[2] `http://www.insee.fr/en/methodes/default.asp?page=definitions/departement.htm`

Table 3. Anonymity Related Variables

Variable	Description
k	the anonymity-preserving level k
Tol_s	the minimum standard needed for achieving indistinguishability among people who have the same sex criterion
Tol_a	the minimum standard needed for achieving indistinguishability among people who have the same age group criterion
Tol_d	the minimum standard needed for achieving indistinguishability among people who have the same domicile criterion
ω_1	weight for Tol_s, $0 \leq \omega_1 \leq 1$
ω_2	weight for Tol_a, $0 \leq \omega_2 \leq 1$
ω_3	weight for Tol_d, $0 \leq \omega_3 \leq 1$
Tol	global tolerance $:= \omega_1 Tol_s + \omega_2 Tol_a + \omega_3 Tol_d$

specific criterion, we use a random variable Y to represent the outcome of each search, whether be success or failure. To preserve k-anonymity, we require that the expected number of successful identifications exceeds k among n sequential searches. That is

$$E(Y) = np \geq k \Longleftrightarrow n \geq \frac{k}{p} \tag{3}$$

where n is the number of searches, p the probability of a successful identification. The value p is determined by the probability associated with the particular search criterion. For practical calculations as used in deriving the lower bounds associated with Tol_s, Tol_a, Tol_d, we have used a rough estimate $p = 1/\#Group$, where $\#Group$ is the number of groups under a specific criterion.

2.3 Derivation of Formula (1)

We prove the privacy-preserving drawer size formula (1) in this section. To determine the drawer size TT, one must fix a priori a search scenario specifying the sex, age group, and domicile region criteria. For instance, the search profile contains the following information: sex $S = i$, age group $A = j$, domicile $D = k$, where i, j, k are fixed in their respective domains. Henceforth, the expected value $E(X)_{\mathbf{P}}$ based on the default distributions $\mathbf{P_s}$, $\mathbf{P_a}$,$\mathbf{P_d}$ is

$$\begin{aligned} E(X)_{\mathbf{P}} &= \frac{1}{Pr(S = i; A = j; D = k)} \\ &= \frac{1}{p_i^s \cdot p_j^a \cdot p_k^d} \end{aligned}$$

assuming a geometric distribution on the search random variable X. Thus, the number of persons in the drawer satisfying the condition $(S = i; A = j; D = k)$ is expected to be about $TT/E(X)$. To preserve indistinguishability among the matched individuals, $TT/E(X)$ must exceed the anonymity tolerance $Tol(k)$:

$$\frac{TT}{E(X)} \geq Tol(k) \Longleftrightarrow TT \geq E(X) \cdot Tol(k).$$

The formula (1) now follows if we let $TT = E(X) \cdot Tol(k)$. And one may obviously without loss of generality assume $N_{expire} = 0$ in the proof.

3 Empirical Study of System Reliability

3.1 Attack Model

We present an attack model here, based on which the reliability of the system can be defined. In an identity theft scenario, the attacker assumes the identity of another person (or he hijacks the biometric data of another person). Presenting himself before the registration authority, the theoretical probability of his success in finding a match between the drawer ID of the identity and the drawer ID of the biometric data is:

$$p = \frac{1}{NT} \tag{4}$$

and we define the system reliability as $1 - p$. The lower attack probability is equivalent to a higher system reliability.

3.2 Test Methodology

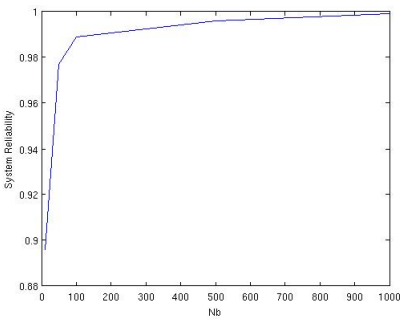
Our confidence tests are based upon the attack model described above. The test is performed on the prototype setbase system. Specifically for a fixed population of 10,000 and a fixed number of drawers, we perform identity thefts at the registration authority, and record the number of attack successes. The test results are recorded in Table 4. The theoretical attack rates are recorded in the second column. The observed attack rates are recorded in the third column. The observed attack rate is expressed as a ratio of the number of observed attack successes over the number of attack attempts. For the statistics generated in this section, we have performed attack attempts in the range of 1000 to 5000. The system reliability rates are obtained by subtracting the observed attack rates from 1. Figure 5 is a plot of the system reliability versus the number of drawers NT.

The observed attack rates adhere to the theoretical attack rates. This indicates that the basic architecture of the prototype system is sound and corresponds to how it should be. The source of absolute deviation between the theoretical rates and observed rates stems from the system performance fluctuations, and the inherent system errors.

Table 4. Probability of Attack Success for various NT

Test Population = 10,000		
Nb. Drawers NT	Theoretical attack Prob.	Observed Prob.
10	10%	10.4%
50	2%	2.3%
100	1%	1.1%
500	0.2%	0.4%
1,000	0.1%	0.1%
5,000	0.02%	0.01%

Table 5. Confidence Test: System Reliability versus NT



4 Privacy versus Security

For sufficiently large population, the variables number of drawers (NT) and size of a drawer (TT) are related by

$$Population \approx NT \cdot TT. \tag{5}$$

If we need privacy protection in the system, one possible solution would be to determine the minimum drawer size by means of formula (1). We then may adjust the number of drawers to reach a suitable system reliability level. We make this analysis rigorous in this section. Without consideration of privacy, we could chose a suitable NT based on the test results described in the previous section (Table 4).

We need the following data for carrying out the relational analysis: TT, NT, anonymity level k, attack probability, system reliability. Precisely, we first calculate the anonymity-preserving drawer size TT based on the general formula (1). The next step is to calculate NT using the formula (5), and the corresponding attack probability is calculated using the single attempt attack model (4). The reliability of the system is subsequently derived by subtracting the attack probability from 1.

We demonstrate the relation between TT and NT by studying the relation between the anonymity level variable k and the system reliability variable. Figure 1 and Figure 2 are two typical relational plots for the variables k and system reliability. Figure 1 reflects the entire range of $k, 2 \leq k \leq 30$. Figure 2 reflects the range $k, 2 \leq k \leq 10$ within which the highest system reliability rates occur. As can be been from the plots, the relation between k and system reliability is linear, inversely proportional. This fact again illustrates the underlying principle of the setbase: a stricter anonymity requirement (large k) requires a larger drawer size to be in place, that corresponds to a decrease on the number of drawers which in turn leads to a lower system reliability rate.

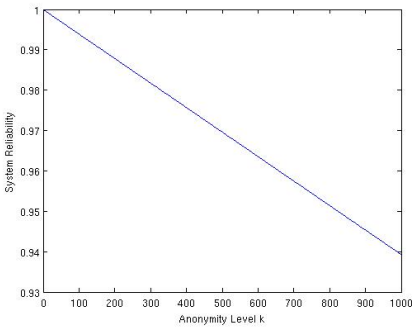


Fig. 1. System Reliability vs. Anonymity Level k, $2 \leq k \leq 1000$

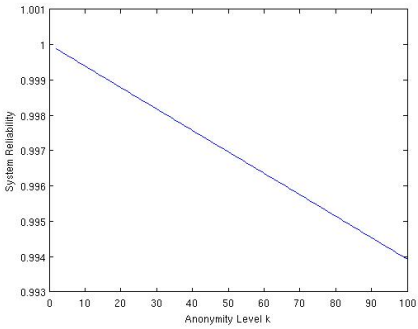


Fig. 2. System Reliability vs. Anonymity Level k, $2 \leq k \leq 100$

5 Conclusion

We have in this paper proposed a privacy-preserving drawer size standard for the setbase. The standard incorporates well known data privacy metrics such as k-anonymity and l-diversity as part of the drawer size formulation. The mathematical formulation gives one the ability to adjust the drawer size according the desired level of privacy. The future research in this direction includes issues such as how one can incorporate other database privacy notions into the privacy-preserving drawer size standard.

Acknowledgments. We would like to thank Vincent Pineau for implementing the prototype setbase in JAVA.

References

1. Assemblée Nationale No. 3599. Rapport relative à la protection de l'identité. Technical report (2011)
2. Didier, B., Rieul, F.: Person identification control method and system for implementing same, Patent US7724924 (2010)
3. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Trans. Knowl. Data Eng. 13(6), 1010–1027 (2001)
4. Samarati, P., Sweeney, L.: Generalizing data to provide anonymity when disclosing information (abstract). In: PODS, p. 188 (1998)
5. No.126 SÉNAT. Proposition De Loi: relative à la protection de l'identité. Technical report (2011)
6. Shamir, A.: Adding privacy to biometric databases: The setbase approach. Presentation Slide (2009)

Rule Enforcement with Third Parties in Secure Cooperative Data Access

Meixing Le, Krishna Kant, and Sushil Jajodia

George Mason University, Fairfax, VA 22030
{mlep,kkant,jajodia}@gmu.edu

Abstract. In this paper, we consider the scenario where a set of parties need to cooperate with one another. To safely exchange the information, a set of authorization rules is given to the parties. In some cases, a trusted third party is required to perform the expected operations. Since interactions with the third party can be expensive and there maybe risk of data exposure/misuse, it is important to minimize their use. We formulate the minimization problem and show the problem is in NP-hard. We then propose a greedy algorithm to find close to optimal solutions.

Keywords: Authorization rule, Trusted third party, Join Path.

1 Introduction

In many cases, enterprises need to interact with one another cooperatively to provide rich services. For instance, an e-commerce company needs to obtain data from a shipping company to know the status and cost of a shipping order, and the shipping company requires the order information from the e-commerce company. Furthermore, the e-commerce company may have to exchange data with warehouses and suppliers to get the information about the products. In such an environment, information needs to be exchanged in a controlled way so that the desired business requirements can be met but other private information is never leaked. For example, a shipping company has all the information about its customers. However, only the information about the customers that deal with the e-commerce company in question should be visible to the e-commerce company. The information about the remaining customers should not be released to the e-commerce company. In addition, the data from shipping company may include other information such as which employee is delivering the order, and such information should not be released to the e-commerce company. Therefore, we need a mechanism to define the data access privileges in the cooperative data access environment.

We assume that each enterprise manages its own data and all data is stored in a standard relational form such as BCNF, but it is possible to extend the model to work with other data forms. The data access privileges of the enterprises are regulated by a set of authorization rules. Each authorization rule is defined either on the original tables belonging to an enterprise or over the lossless joins of the data from several different parties. Using join operations, an

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 282–288, 2013.

authorization rule only releases the matched information from the parties. For instance, if the e-commerce company can only access the join result of its data and the shipping company's data, then only the tuples about the shipping orders from the e-commerce company can be visible to the e-commerce company. In addition, the attributes such as "delivery_person" are never released to the e-commerce company, so suitable projection operations are applied on the join results in authorization rules to further restrict the access privileges. Hence, the requirement of selective data sharing can be achieved. Selection operations are not considered in the authorization rules.

Under such a scenario, an enterprise may be given an authorization rule on the join result of several relational tables. To obtain the join result, it is required to have one party that has the privileges to access all the basic relations and perform the required join operations. However, due to the access restrictions laid down by the authorization rules, it is possible that no party is capable of receiving all required data. Therefore, we may have to introduce a trusted third party to perform join operations.

Third parties may be expensive to use and the data given to them could be at greater risk of exposure than the data maintained by original parties. In this paper, we focus on the problem of using third parties minimally in order to deliver the information regulated by the given authorization rules. We model the cost of using third party as the amount of data being transferred to the third party, and prove that finding the minimal amount of data to implement a given rule is NP-hard. Therefore, we propose efficient greedy algorithm and evaluate its performance against brute force algorithm. The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 defines the problem and introduces some concepts. Section 4 discusses minimizing the cost of using third parties. Finally, Section 5 concludes the discussion.

2 Related Work

In previous works, researchers proposed models [5] for controlling the cooperative data release. There is also a mechanism [8] to check if an authorization rule can be enforced among cooperative parties. In addition, many classical works discuss query processing in centralized and distributed systems [2,7,3]. However, these works do not deal with constraints from the data owners, which make our problem quite different. There are works such as Sovereign joins [1] to provide third party join services, we can think this as one possible third party service model in our work. Such a service receives encrypted relations from the participating data providers, and sends the encrypted results to the recipients.

Because of the risks associated with third parties, secure multiparty computation (SMC) mechanisms have been developed that ensures no party needs to know about the information of other parties [6,9,4]. However, the generic solution of a SMC problem can be very complicated depending on the input data and does not scale in practice. Therefore, we consider using the third party to implement the rules.

3 Problem Definition and Concepts

We assume the possible join schema is given and all joins are lossless so that a join attribute is always a key attribute of some relations, and only select-project-join queries are considered. An authorization rule denoted as r_t is a triple $[A_t, J_t, P_t]$, where J_t is called the **join path** of the rule which indicates the join over the relational tables, A_t is the authorized attribute set which is the authorized attributes on the join path, and P_t is the party authorized to access the data. For instance, an example rule could be $(R.K, R.X, S.Y), (R \bowtie_{R.K} S) \rightarrow P_t$, where $R.K$ is the key attribute of both R and S, and join path is $R \bowtie S$.

We assume that a trusted third party (TP) is not among the existing cooperative parties and can receive information from any cooperative party. We assume that the TP always performs required operations honestly, and does not leak information to any other party. In our model, we assume the trusted third party works as a service. That is, each time we want to enforce a rule, we need to send all relevant information to the third party, and the third party is only responsible for returning the join results. After that, the third party does not retain any information about the completed join requests. We say an authorization rule can be *enforced* only if there is a way to obtain all the information regulated by the rule. With the existence of a third party, we can always enforce a rule by sending relevant information from cooperative parties to TP. We aim to minimize the amount of data to be sent to the third party.

To find the minimal amount of data to be sent, we can just select rules from the given authorization rules. It is because each rule defines a relational table and we can quantify the amount of information using the data in the tables. We say that a rule is *Relevant* to another if the join path of a rule contains a proper subset of relations of the join path of the other rule. All the rules being selected must be relevant to the target rule denoted as r_t, which is the rule to be enforced. If a relevant rule of r_t is not relevant to any other relevant rules of r_t with longer join paths on the same party, we call it a **Candidate Rule**. We only choose from candidate rules to decide the data that needs to be sent to the TP.

4 Minimizing Cost

In this section, we consider the problem of choosing the proper candidate rules to minimize the amount of information sent to the third party. In our cost model, the amount of information is quantified by sum of the number of attributes picked from each rule multiplied by the number of tuples in that selected rule. Thus, we want to minimize $Cost = \sum_{i=1}^{k} \pi(r_i) * w(r_i)$, where r_i is a selected rule, k is the number of selected rules, and $\pi(r_i)$ is the number of attributes selected to be sent, and $w(r_i)$ is the number of tuples in r_i.

4.1 Rules with Same Number of Tuples

We first assume the candidate rules have the same $w(r_i)$ value. To find the candidate rules that can provide enough information to enforce r_t, we map each attribute in r_t to only one candidate rule so that all of these attributes can be covered. Once we get such a mapping, we have one solution including the selected rules and projections on desired attributes. Among these solutions, we want the minimal cost solution according to our model. Since we assume all the candidate rules have the same number of tuples, it seems that the total cost of each candidate solution should always be the same. However, it is not true because the join attributes appearing in different relations are merged into one attribute in the join results. We can consider the example in Figure 1. The boxes in the figure show the attribute set of the rules, and the join paths and rule numbers are indicated above the boxes. There are four cooperating parties indicated by P_i and one TP, and the three basic relations are joining over the same key attributes $R.K$. Among the 4 candidate rules, if we select r_2, r_3 to retrieve the attributes $R.X$ and $S.Y$ (non-key attributes), we need to send $R.K$ and $S.K$ which are their join attributes to the third party as well. Whereas, if we choose r_1, then we only need to send 3 attributes as $R.K$ and $S.K$ are merged into one attribute in r_1. Thus, choosing a candidate rule with longer join path may reduce the number of attributes actually being transferred. Fewer rules means fewer overlapped join attributes to be sent (e.g., $R.K$ in r_1 and $T.K$ in r_4 are overlapped join attributes). In addition, selecting fewer rules can result in fewer join operations performed at the third party. Since we assume the numbers of tuples in candidate rules are the same, the problem is converted to identify minimal number of candidate rules that can be composed to cover the target attribute set. The problem can be reduced to unweighted set covering problem which is NP-hard.

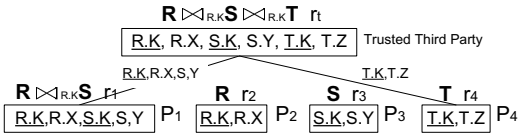


Fig. 1. An example of choosing candidate rules

Theorem 1. *Finding the minimal number of rules sent to the third party to enforce a target rule is NP-hard.*

Proof. Consider a set of elements $U = \{A_1, A_2, ..., A_n\}$ (called the universe), and a set of subsets $S = \{S_1, S_2, ...S_m\}$ where S_i is a set of elements from U. The unweighted set covering problem is to find the minimal number of S_i so that all the elements in U are covered. We can turn it into our rule selection problem. For this we start with the attribute set $\{\underline{A_0}, A_1, A_2, ..., A_n\}$, where A_0

is the key attribute of some relation R and A_i's are non-key attributes of R. For each $S_i \in S$, we construct a candidate rule r_i on R with the attribute set $S_i \bigcup \{A_0\}$ and assign it to a separate cooperative party. Therefore, if we can find the minimal set of rules to enforce some target rule r_t in polynomial time, the set covering problem can also be solved in polynomial time.

4.2 Rules with Different Number of Tuples

In general, the numbers of tuples in the relations/join paths are different, and they depend on the length of the join paths and the join selectivity among the different relations. *Join selectivity* [7] is the ratio of tuples that agree on the join attributes between different relations, and it can be estimated using the historical and statistical data of these relations. In classical query optimization, a large number of works assume such values are known when generating the query plans. We also assume that this is the case. Therefore, we can assign each candidate rule r_i with a cost $cst_i = w(J_i) * \pi(r_i)$, where $\pi(r_i)$ is the per tuple cost of choosing rule r_i, and $w(J_i)$ is the number of tuples in join path J_i. The problem is similar to (but not identical to) the weighted set covering problem. In our problem, once some attributes are covered by previously chosen rules, the following chosen rules should project out these attributes so as to reduce cost. Therefore, our cost function should be as follows where S_i is the attribute set of rule r_i and U is the target attribute set. Basically, the equation says if the key attribute of a rule has already been covered, then one more attribute is added to the cost of choosing this rule.

$$cost(C) = \sum_{i=1}^{k} w(S_i)\pi(S_i), \pi(S_i) = \begin{cases} |S_j \bigcap (U \setminus \bigcup_{j=1}^{i-1} S_j)|, & \text{if } (key(S_i) \notin \bigcup_{j=1}^{i-1} S_j) \\ |S_j \bigcap (U \setminus \bigcup_{j=1}^{i-1} S_j)| + 1, & \text{if } (key(S_i) \in \bigcup_{j=1}^{i-1} S_j) \end{cases} \quad (1)$$

Corollary 1. *Finding the minimal amount of information sent to the third party to enforce a target rule is NP-hard.*

Proof. Based on Theorem 1, if we have a polynomial algorithm to find the minimal amount of information with rules of different costs, we can assign the same cost to each candidate rule so as to solve the unweighted version of the problem.

In the weighted set covering problem, the best known greedy algorithm finds the most effective subset by calculating the number of missing attributes it contributes divided by the cost of the subset. In our case, we also want to select the attributes with least costs from the available subsets. Similar to the weighted set covering algorithm which selects the subset S_i using the one with minimal $\frac{w(S_i)}{|S_i \setminus U|}$, we select the rule with the minimal value of $\frac{w(S_i)*\pi(S_i)}{|S_j \bigcap (U \setminus \bigcup_{j=1}^{i-1} S_j)|}$, where $\pi(S_i)$ is defined in equation (1).

In our problem, with one more rule selected, the third party need to perform one more join operation, and possibly one more join attribute need to be transferred to the third party. Therefore, when selecting a candidate rule, we examine

Algorithm 1. Selecting Minimal Relevant Data For Third Party

Require: The set R of candidate rules of r_t on cooperative parties
Ensure: Find minimal amount of data being sent to TP to enforce r_t

for Each candidate rule $r_i \in R$ **do**
  Do projection on r_i according to the attributes in r_t
  Assign r_i with its estimated number of tuples t_i
The set of selected rules $C \leftarrow \emptyset$
Target attribute set $U \leftarrow$ merged attribute set of r_t
while $U \neq \emptyset$ **do**
  Find a rule $r_i \in R$ that minimize $\alpha = \frac{w(S_i) * \pi(S_i)}{|S_j \bigcap (U \setminus \bigcup_{j=1}^{i-1} S_j)|}$
  $R \leftarrow R \setminus r_i$
  for Each attribute $A_i \in (r_i \bigcap U)$ **do**
    $cost(A_i) \leftarrow w(S_i)$
  $r_i \leftarrow \pi_U(r_i) * w(S_i)$
  $U \leftarrow U \setminus r_i$
  $C \leftarrow C \bigcup r_i$
Return C

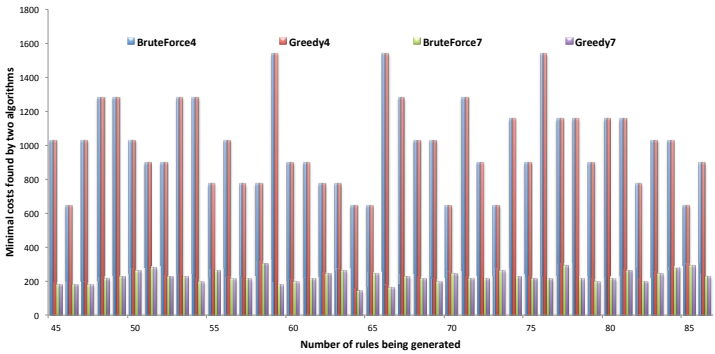


Fig. 2. Minimal communication costs found by two algorithms

the number of attributes this rule can provide and the costs of retrieving these attributes. In the second case of $\pi(S_i)$ in equation (1), the cost of one extra attribute is added. However, if this selected rule can provide many attributes to the uncovered set, the cost of this additional attribute can be amortized. This makes the algorithm prefer rules providing more attributes and results in less number of selected rules which is consistent with our goal. We present our greedy algorithm in Algorithm 1.

We evaluated the effectiveness of our greedy algorithm against brute force algorithm via preliminary simulations. In this simulation evaluation, we use a join schema with 8 parties. The number of tuples in a rule is defined as a function of the join path length, basically $w(J_i) = 1024/2^{length(J_i)-1}$. In other words, we assume as the join path length increases by one, the number of tuples in the results decreases by half. We tested with randomly generated target rules with join path length of 4 and 7. Figure 2 shows the comparison between two algorithms. In fact, the two algorithms generate almost the same results. In Figure 2, the legend of "BruteForce4" indicates the target rule has the join path

length of 4, and brute force algorithm is used. Among these solutions, in less than 2% of the cases the two algorithms produce different answers. In addition, the maximal difference between them is just 5%. The results also indicate the join path length of the target rule affects the costs, but two algorithms give similar solutions independent of the join path length.

5 Conclusions and Future Work

In this paper, we considered a set of authorization rules for cooperative data access among different parties. A trusted third party may be required to do the expected join operations so as to enforce a given rule. We discussed what is the minimal amount of data to be sent to the third party. As the problem is NP-hard, we proposed greedy algorithms to generate solutions which were close to the optimal ones. In the future, we will look into the problem of how to combine the third parties with the existing parties to generate optimal safe query plans.

References

1. Agrawal, R., Asonov, D., Kantarcioglu, M., Li, Y.: Sovereign joins. In: Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, Atlanta, GA, USA, April 3-8, IEEE Computer Society (2006)
2. Bernstein, P.A., Goodman, N., Wong, E., Reeve, C.L., Rothnie Jr., J.B.: Query processing in a system for distributed databases (SDD-1). ACM Transactions on Database Systems 6(4), 602–625 (1981)
3. Chaudhuri, S.: An overview of query optimization in relational systems. In: Proceedings of the 7th ACM Symposium on Principles of Database Systems (1998)
4. Chow, S.S., Lee, J.-H., Subramanian, L.: Two-party computation model for privacy-preserving queries over distributed databases. In: Proceedings of the 16th Network and Distributed System Security Symposium (2009)
5. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Controlled information sharing in collaborative distributed query processing. In: ICDCS 2008, Beijing, China (June 2008)
6. Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
7. Kossmann, D.: The state of the art in distributed query processing. ACM Computer Survey 32(4), 422–469 (2000)
8. Le, M., Kant, K., Jajodia, S.: Rule configuration checking in secure cooperative data access. In: SafeConfig 2012 (October 2012)
9. Mishra, D.K., Trivedi, P., Shukla, S.: A glance at secure multiparty computation for privacy preserving data mining. International Journal 1 (2009)

Unlinkable Content Playbacks in a Multiparty DRM System*

Ronald Petrlic and Stephan Sekula

University of Paderborn, Germany
ronald.petrlic@upb.de, sekula@live.upb.de

Abstract. We present a solution to the problem of privacy invasion in a multiparty digital rights management scheme. (Roaming) users buy content licenses from a content provider and execute it at any nearby content distributor. Our approach, which does not need any trusted third party—in contrast to most related work on privacy-preserving DRM—is based on a re-encryption scheme that runs on any mobile Android device. Only a minor security-critical part needs to be performed on the device's smartcard which could, for instance, be a SIM card.

1 Introduction

Mobile users access digital content provided in the cloud from anywhere in the world. Music streaming services like *Spotify* enjoy popularity among users. The lack of bulky storage on mobile devices is compensated for by such services by streaming the content to the users's devices. Content is downloaded on demand and can be used only during playback. Thus, paying users are able to access huge amounts of content. There exist certain price models that allow the playback for a certain number of times, until a specific day (e.g., movie rentals), etc.

In such a scenario, we have *content providers* (CPs) that sell licenses to users and there are *content distributors* (CDs) that provide the content. Users can access content from CDs that are closest or provide best service at the moment. This bears advantages for roaming users as they can choose local distributors. Such scenarios are called *multiparty* DRM systems in the literature.

A drawback of today's DRM systems is that CPs/CDs can build content usage profiles of their users as they learn which user plays back content at a certain time, etc. Here we contribute with this paper. We suggest a privacy-preserving multiparty DRM system. In such a system, users anonymously buy content and anonymously playback the content. Moreover, neither CPs nor CDs can link content playbacks to each other and thus cannot build usage profiles under a pseudonym—as the past has shown that profiles under a pseudonym, assumed to be unrelatable to users, can be related to users given external information and thus, inverting user privacy again [1]. One major advantage of our approach compared to related work on privacy-preserving DRM is that we do not need a trusted third party (TTP) that checks licenses.

* This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre "On-The-Fly Computing" (SFB 901). The extended version of this paper can be found at arXiv:1304.8109.

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 289–296, 2013.

2 Related Work

In [2] a scenario where a content owner provides its content to users via local distributors is presented—similar to our scenario. Users buy licenses for content from a license server (trusted third party). Once a license is bought, the user gets in possession of the decryption key which allows him to access the content as often as desired. Differentiated license models are not intended—however, if license enforcement additionally took place on the client-side, such models could be implemented. As content download and license buying are done anonymously, none of the parties can build user profiles. [3] presents a privacy-preserving DRM scheme for multiparty scenarios without a TTP. A user anonymously requests a token set from the content owner that allows anonymous purchase of content licenses from content providers (CPs). A drawback is that CPs are able to build usage profiles of content executions under a pseudonym. [4] presents a DRM scenario that allows users to anonymously buy content from any CP and execute it at any computing center within the cloud. The users' permission to execute the content is checked before every single execution. Their solution is resistant against profile building. The authors suggest employing a re-encryption scheme based on secret sharing and homomorphic encryption to achieve unlinkability of executions. The approach is extended in [5] by employing an adapted version of proxy re-encryption [6]. The scheme makes explicit use of a service provider as TTP. The approach towards privacy-preserving DRM in [7] also requires a TTP for license checking before execution. It makes use of a number of cryptographic primitives such as proxy re-encryption, ring signatures and an anonymous recipient scheme to provide unlinkability of executions.

3 System Model

Our multiparty DRM scenario involves CPs, CDs, and users. The focus is on mobile users with different *content access devices* (CADs) accessing content. As devices have different hardware trust anchors—e.g., smartphones are equipped with SIM cards, tablet computers have *trusted platform modules* (TPMs), etc.—we subsume those trust anchors under the term *smartcards* in the following.[1]

The CP takes the role of, e.g., a film studio or music label that produces content. Users interact with the CP by buying a license that allows playback of the content—under certain terms that are mediated. The user's smartcard is used to check whether the user is still allowed to access the content. Then, a nearby CD is contacted and the CD streams the content to the user. The CD can have contracts with different CPs, which allows the user to access content by different CPs from a single source—as it is the case with state-of-the-art streaming servers as well. The CD might get paid for providing its services by the CPs (or even the users). We do not cover this aspect in the paper at hand.

We assume that CPs and CDs are *honest-but-curious*, i.e., they follow the protocol but try to find out as much as possible to track users. Users are assumed

[1] SIM cards are smartcards and TPMs are a special form of smartcards as well.

as *active adversaries*, i.e. trying to break the protocol to execute content without a license. Our protocol is not based on any TTP checking licenses.

DRM Requirements:

We identify the CP, CD, and the user as stakeholders. The requirements are:

Content Provider: *Req. I: Support for different license models*, *Req. II: Protection of the content (confidentiality)*, and *Req. III: Enforcement of licenses.*

User: *Req. IV: Profile building (under a pseudonym) must not be possible for any involved party.* To achieve Req. IV, the the following aspects must be met: *Anonymous content (license) buying towards content provider, and anonymous content execution towards content distributor*, *Unlinkability of content (license) purchases towards the content provider*, and *Unlinkability of content executions towards the content distributor.*

4 Privacy-Preserving Multiparty DRM System

System Initialization: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be cyclic groups with the same prime order q, the security parameter $n = ||q||$, $<g> = \mathbb{G}_1$, and $Z = e(g, g) \in \mathbb{G}_2$. Users are equipped with smartcards (SCs) that are programmed and shipped by trustworthy SC providers that install a private key sk_{sc} and the corresponding digital certificate $cert_{sc}$ on every smartcard. The private key and certificate are shared by all SCs since they are used for anonymous authentication towards the CP during the process of purchasing content. Authentication of SCs is required so that only legitimate SCs can be used to purchase content, however, CPs must not be able to recognize SCs. Moreover, the current time of production of the SC is set as the SC's timestamp ts. Content offered by the CP is encrypted using a symmetric encryption algorithm such as *AES* [8] and a separate content key ck_i for each content i. The user employs an anonymous payment scheme with his/her bank to get supplied with payment tokens pt.

Content Purchase: We assume that the connection between user and CP is anonymized (e.g., by using an anonymization network such as Tor [9]). The user initiates the content purchase via his/her content access device (CAD) by authenticating towards the SC with his/her PIN and initiating the TLS [10] handshake with the CP. The SC executes the KG algorithm as in [6] to generate a temporary key pair[2] ($pk\text{-}tmp_{sc} = (Z^{a_1}, g^{a_2}), sk\text{-}tmp_{sc} = (a_1, a_2)$), where $a_1, a_2 \in \mathbb{Z}_q$ are chosen randomly. During the TLS handshake, CP challenges CAD's SC with a nonce r and asks for SC's certificate. CAD forwards r to SC which signs r and $pk\text{-}tmp_{sc}$ with SC's private key sk_{sc}. The *signature* and SC's certificate $cert_{sc}$, as well as $pk\text{-}tmp_{sc}$ are forwarded to CAD and CAD forwards them, together with the *content-id*$_i$ of the content i to be bought, as well as the payment token pt to pay for the license. From this moment on, the communication between CAD and CP is authenticated and encrypted via TLS. CP verifies the response by checking the *signature*. This way, CAD's SC has anonymously authenticated towards CP, meaning CP knows that $pk\text{-}tmp_{sc}$ is from an

[2] A new temporary key pair is used for each content purchase.

authentic SC and the corresponding $sk\text{-}tmp_{sc}$ does not leave the SC. CP creates the license for content i. This license includes a license identifier id, a timestamp ts, the $content\text{-}id_i$, the license $terms$, and CP's certificate $cert_{cp}$. Note that the license terms depend on the license model. The license is encrypted under SC's $pk\text{-}tmp_{sc}$. Moreover, the content key ck_i for content i is encrypted under $pk\text{-}tmp_{sc}$ as well. The *license*, the *signature* of the license, the $content\text{-}id_i$ and the encrypted content key $(ck_i)_{pk-tmp_{sc}}$ are forwarded to CAD. CAD stores $(ck_i)_{pk-tmp_{sc}}$ and forwards the *license* and the *signature* to SC. The SC verifies the license's signature and decrypts the license with $sk\text{-}tmp_{sc}$. Then it checks whether the id was not used before and whether ts is newer than the current ts on the SC—both to prevent replay attacks. The SC's ts is then set to the newer ts of the license.[3] Finally, the license is stored under the $content\text{-}id_i$ on the SC.

Content Execution: To playback the purchased content, the user first selects a CD of his choice (this choice could be automated as well, e.g., dependent of the region the user currently is in). We assume that the connection between user and CD is anonymized (e.g., by using Tor [9]). The CAD establishes a TLS connection [10] with CD—CD authenticates towards CAD with its certificate. CAD afterwards requests a new certificate from CD. CD creates a new key-pair using KG as in [6]: $(pk\text{-}j_{cd} = (Z^{a_1}, g^{a_2}), sk\text{-}j_{cd} = (a_1, a_2))$, where $a_1, a_2 \in \mathbb{Z}_q$ are chosen randomly and j denotes the j^{th} request to the CD. The $pk\text{-}j_{cd}$ is included in the newly generated certificate $cert\text{-}j_{cd}$, as well as a unique certificate id and the current timestamp ts. CD self-signs the certificate[4]. The certificate is forwarded to the CAD. The user authenticates towards the SC with his PIN entered on the CAD and the SC then forwards the list of available *content-ids* to CAD. The user chooses the $content\text{-}id_i$ to be executed and forwards it, together with $cert\text{-}j_{cd}$ to SC. SC checks whether the signature of $cert\text{-}j_{cd}$ is valid, whether CD was certified by a known CA, whether the certificate id was not used before and whether the ts is newer than the current ts on the SC. If these tests pass, the new ts from the certificate is set on the SC. It is important to note, that SC checks whether the certificate really belongs to a CD. If this was not the case, the user might be able to launch an attack by including a self-signed certificate that he has generated himself. Hence, if SC would not verify that the certificate belonged to a CD, the user might acquire a re-encryption key from SC that allowed him to decrypt the content key, granting him unlimited access to the content. Furthermore, SC checks whether the license $terms$ still allow the content to be played back. If this is the case, the $terms$ are updated. Then, SC generates the re-encryption key $rk_{pk-tmp_{sc} \to pk-j_{cd}}$ by using the RG algorithm as in [6], taking as input CD's public key $g^{a_2} \in pk\text{-}j_{cd}$, and its own private key $a_1 \in sk\text{-}tmp_{sc}$ (as created during the content purchase). The re-encryption key is then forwarded to CAD. CAD re-encrypts the encrypted content key $(ck_i)_{pk-tmp_{sc}}$ by employing the R algorithm as in [6] with $rk_{pk-tmp_{sc} \to pk-j_{cd}}$ as input to retrieve $(ck_i)_{pk-j_{cd}}$—i.e., the encrypted content key under CD's public

[3] Note that the SC does not have an internal clock and thus cannot keep track of (authenticated) time. The time can only be set via new and verified licenses.

[4] The signing certificate was issued by a valid certificate authority, though.

key. The re-encrypted content key is then forwarded to CD and CD decrypts the ciphertext using the D algorithm as in [6] with its private key $a_2 \in sk\text{-}j_{cd}$ as input to retrieve ck_i. The content—retrieved from CP—can now be decrypted by CD using ck_i and the symmetric scheme as employed during system initialization. Eventually, the content is provided, for example, streamed, to the user's CAD.

Authorization Categories [11]: There might be content that should not be accessible to everybody, such as X-rated content. Before initially obtaining a SC, the user provides certain information to the SC provider (e.g., his passport). The SC provider will then securely[5] store the required information on the user's SC. If we assume that the user's SC now contains information like the user's date of birth or home country, it can check whether or not the user is allowed to access content. This means that if the user requests access to, for instance, X-rated content, the SC checks the user's date of birth and according to this information either allows or denies access to the queried content.

5 Evaluation and Discussion

Performance Analysis: The user's CAD performs the re-encryption of the content key. CP and SC are involved in a challenge-response protocol for authentication of SC which is not too expensive. Further, CP has to encrypt the content key using SC's public key and the content using the content key. The latter is a symmetric encryption executed only once per content. Additionally, CD decrypts the re-encrypted content key as well as the content obtained from CP. The required generation of keys is not expensive. We show that current smartphones are easily capable of executing the required tasks by implementing a demo application on an Android smartphone. We have implemented the re-encryption using the jPBC (*Java Pairing Based Cryptography*) library[6]. The app that has been developed re-encrypts 128 Bytes of data—the length of a symmetric key to be encrypted—in $302\,ms$ on a Samsung Galaxy Nexus ($2 \times 1.5\,GHz$) running Android 4.2. Due to a lack of a proper SC[7], we could not implement the re-encryption key generation algorithm RG as in [6]. Thus, to show the practicability of the implementation, we must refer to [12]. The authors have implemented elliptic curve scalar point multiplications and additions for a smartcard in C and Assembler—which are needed in our approach as well. As the authors conclude, the standard Javacard API (version 2.2.2) cannot be used as the available EC Diffie-Hellman key exchange only provides the hashed version of the key derivation function. [12] However, we need the immediate result of the key derivation function, i.e., the result of the EC point multiplication. Our own implementation of the EC point multiplication on the smartcard's CPU did not yield practicable results—as the efficient cryptographic co-processors could not be utilized due to proprietary code.

[5] Secure storage in this context especially means integrity-protection.

[6] `http://gas.dia.unisa.it/projects/jpbc/`

[7] According to the specifications, the NXP JCOP card 4.1, V2.2.1 can be used to implement the needed functionality.

Evaluation of Requirements: CP is able to provide different kinds of rights to users for content playback. Our system allows for the most popular models like *flatrate*, *execute at most n-times*, *execute until a certain date*, etc., and thus we meet *Req. I*. CP distributes its content only in encrypted form. Thus, none of the parties not in possession of the content decryption key is able to access the content and our protocol meets *Req. II*. Smartcards, as trusted devices, are used in our protocol to enforce licenses. Thus, if the SC's check of a license fails, the re-encryption key is not generated and the user is not able to execute the content. A replay attack with an "old" CD certificate fails as the SC does not accept the ts—since it is older than the current one stored on the SC. The SC's property of tamper-resistance is required since we assumed users to be active adversaries. Thus, we meet *Req. III*. Concernng *Req. IV* we have:

(1) Users anonymously pay for content (licenses), i.e., they do not need to register with CP/CD and need not provide their payment details, which is why they stay anonymous during their transactions with CP and CD.

(2) All SCs use the same certificate for anonymous authentication towards CP, thus CP cannot link different purchases made with the same SC. SC's public key $pk\text{-}tmp_{sc}$ is newly generated for each content (license) purchase—preventing CP from linking purchases to each other. Moreover, the anonymous payment scheme provides unlinkability of individual payments. Furthermore, we assumed the connection between user and CP to be anonymized via Tor. Thus, unlinkability of content (license) purchases is achieved.

(3) The user only provides the re-encrypted content key to CD. Content i is only encrypted once during initialization with ck_i and thus, ck_i does not contain any information connected to the user or the user's CAD. As a new re-encryption key is generated for each content execution, the encrypted content key "looks" different for CD each time and hence, CD cannot link any pair $(ck_i)_{pk-j_{cd}}$, $(ck_i)_{pk-k_{cd}}$, for $j \neq k$ to each other. Further, we assumed the connection between user and CD to be anonymized via Tor. Therefore, multiple transactions executed by the user are unlinkable for the CD.

Moreover, even if an attacker gets access to the user's CAD, he does not learn which content has been bought and executed. The list of available content is only revealed by the SC after authentication with the proper PIN and the CAD application does not keep track of executed content. Thus, to sum it up, profile building (even under a pseudonym) is neither possible for CP nor CD.

Comparison to Related Work: In Tab. 1 we compare our proposed scheme to related work in the field of privacy-preserving digital rights management.

Need for TTP: One of the main advantages of our scheme compared to related work is that it does not need a trusted third party which is involved in the license checking process as in [5, 7] during each content execution. In [2], the license server constitutes the TTP. However, it is not involved in the protocol for each single content execution but only once, when retrieving the license.

Table 1. Comparison of our scheme to related work in terms of properties

Properties	Paper at hand	[7]	[5]	[2]	[3]
Need for TTP	no	yes	yes	yes	no
Need for trusted hardware	yes	no	no	no	yes
Support for differentiated license models	yes	yes	yes	no	yes
Unlinkability of content executions	yes	yes	yes	yes	no
Computational efficiency	good	medium	bad	good	good
Flexibility in choosing content distributor	yes	yes	yes	yes	yes

Need for trusted hardware: In our protocol a smartcard performs the license checking. Trusted hardware is not needed by other protocols that rely on some TTP. A trusted platform module (TPM) is needed in the protocol presented in [3] to securely store tokens at the user's computing platform.

Support for differentiated license models: The protocol presented here and in [5, 7] allow for differentiated license models. The protocol presented in [2] does not allow such flexibility—once a license is bought for some content, it may be executed by the user as often as desired. The authors of [3] do not clearly state whether differentiated license models are intended. From the protocol's point of view, it should be possible to implement, e.g., *execute at most n times*-models as a token set provided by the content owner. Such token sets could include n tokens. Further, licenses that allow only a single content execution could be mapped to each token by the content provider[8] later on.

Unlinkability of content executions: All of the approaches covered here, except for [3], provide unlinkability of content executions and thus, prevent any party from building a content usage profile (under a pseudonym).

Computational Efficiency: In terms of computational overhead, our proposed scheme is very efficient, as discussed above. The scheme presented in [7] makes use of a number of different cryptographic primitives and thus performs less well. In [5], the entire content is re-encrypted for each content execution. Efficient standard cryptographic primitives are used in [2, 3].

Flexibility in choosing content distributor: All the schemes presented in this overview provide users with the possibility to freely choose the CDs. In other two-party DRM scenarios, such a flexibility is typically not provided.

6 Conclusion

We have come up with a privacy-preserving multiparty DRM concept. Users anonymously buy content licenses from a CP and anonymously execute the content at any CD by, for example, streaming the content from CDs nearby.

[8] Content distributor in our scenario.

Anonymity in this context means that none of the involved parties is able to build a content usage profile—not even under a pseudonym. In contrast to related work on privacy-preserving DRM, our approach does not require a trusted third party. We implemented our concept on a state-of-the-art smartphone and proved its practicability for a multiparty DRM scenario in a mobile environment in which a user buys a license allowing the playback of, e.g., some TV show—roaming in different regions, the user is free to choose the nearest streaming server (content distributor) and hence, getting the best throughput.

References

1. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: Proceedings of the 2008 IEEE Symposium on Security and Privacy, SP 2008, pp. 111–125. IEEE Computer Society, Washington, DC (2008)
2. Mishra, D., Mukhopadhyay, S.: Privacy rights management in multiparty multilevel DRM system. In: Proceedings of the International Conference on Advances in Computing, Communications and Informatics, ICACCI 2012, pp. 625–631. ACM (2012)
3. Win, L.L., Thomas, T., Emmanuel, S.: A privacy preserving content distribution mechanism for DRM without trusted third parties. In: IEEE International Conference on Multimedia and Expo (ICME), pp. 1–6 (July 2011)
4. Petrlic, R., Sorge, C.: Privacy-preserving DRM for cloud computing. In: Proceedings of the 26th International Conference on Advanced Information Networking and Applications Workshops, pp. 1286–1291. IEEE Computer Society (2012)
5. Petrlic, R.: Proxy re-encryption in a privacy-preserving cloud computing DRM scheme. In: Xiang, Y., Lopez, J., Kuo, C.-C.J., Zhou, W. (eds.) CSS 2012. LNCS, vol. 7672, pp. 194–211. Springer, Heidelberg (2012)
6. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. 9, 1–30 (2006)
7. Joshi, N., Petrlic, R.: Towards practical privacy-preserving digital rights management for cloud computing. In: Proceedings of The 10th Annual IEEE Consumer Communications & Networking Conference (CCNC 2013), pp. 259–264 (2013)
8. National Institute of Standards and Technology (NIST): Advanced Encryption Standard (AES) (FIPS PUB 197) (November 2001)
9. Dingledine, R., Mathewson, N., Syverson, P.: TOR: the second-generation onion router. In: Proceedings of the 13th Conference on USENIX Security Symposium, SSYM 2004, vol. 13, p. 21. USENIX Association, Berkeley (2004)
10. Internet Engineering Task Force (IETF): The Transport Layer Security (TLS) Protocol, Version 1.2, RFC 5246 (August 2008)
11. Perlman, R., Kaufman, C., Perlner, R.: Privacy-preserving DRM. In: Proceedings of the 9th Symposium on Identity and Trust on the Internetm, IDTRUST 2010, pp. 69–83. ACM, New York (2010)
12. Ullmann, M., Kügler, D., Neumann, H., Stappert, S., Vögeler, M.: Password authenticated key agreement for contactless smart cards. In: Proceedings of Workshop on RFID Security (RFIDsec 2008) (2008)

Analysis of TRBAC with Dynamic Temporal Role Hierarchies

Emre Uzun[1], Vijayalakshmi Atluri[2], Jaideep Vaidya[1], and Shamik Sural[3]

[1] MSIS Department, Rutgers Business School, USA
{emreu,jsvaidya}@cimic.rutgers.edu
[2] National Science Foundation and MSIS Department, Rutgers Business School, USA
atluri@cimic.rutgers.edu
[3] School of Information Technology, IIT Kharagpur, India
shamik@cse.iitkgp.ernet.in

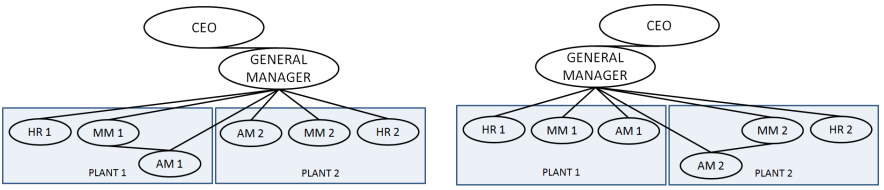
Abstract. The temporal role based access control (TRBAC) models support the notion of temporal roles, user-to-role and permission-to-role assignment, as well as allow role enabling. In this paper, we argue that role hierarchies can be temporal in nature with a dynamism that allows it to have a different structure in different time intervals; and safety analysis of such extensions is crucial. Towards this end, we propose the temporal role based access control model extended with dynamic temporal role hierarchies, denoted as TRBAC_{RH}, and offer an approach to perform its safety analysis. We also present an administrative model to govern changes to the proposed role hierarchy.

1 Introduction

The temporal extension of the role based access control (TRBAC) model assumes one or more of the following features: temporal User to Role Assignments, temporal Permission to Role Assignments, role enabling, and role hierarchies [2,7]. In this paper, we introduce dynamic temporal role hierarchies for TRBAC. Role Hierarchies (RH), or sometimes called Role to Role Assignments (RRA), are one of the three basic relations that are defined in RBAC along with URA and PRA [9]. Whether the basis for RH in an enterprise is either functional or administrative, it simply allows higher level (senior) roles inherit the permissions assigned to the lower level (junior) roles.

In this paper, we argue that the role hierarchies can be temporal in nature, i.e., they may change with time. Although role hierarchies in prior temporal extensions of RBAC have been specified, they do not allow temporal constraints to be specified on RH that not only *restrict the time* during which the hierarchy is valid, but also *change its structure* by shifting the position of the roles in the hierarchy. Essentially this means that a senior level role cannot always inherit the permissions of a junior level role. Also, a role may change its level in the hierarchy, for example, a junior level role may be elevated to a higher level role during certain time periods. To capture this dynamic structure, we enhance the traditional definition of TRBAC with *Dynamic Temporal Role Hierarchy* (DTRH), and we denote the resulting model, TRBAC_{RH}, a temporal role based access control model with dynamic temporal role hierarchies. Although enterprises usually specify a static hierarchy, DTRH comes into play in some temporary or periodical

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 297–304, 2013.



(a) Role Hierarchy when General Manager is at Plant 2

(b) Role Hierarchy when General Manager is at Plant 1

Fig. 1. Role Hierarchies on Different Days of the Week

exceptional situations that are required for operational purposes. In the following, we provide such a motivating example.

Consider a manufacturing company with two different production plants, one having the headquarters of the company. The company has a CEO and a General Manager (GM) who works at both the plants; an Accounting Manager (AM), a Manufacturing Manager (MM), and a Human Resources Manager (HR) for each plant. Although CEO works at the headquarters, GM works in both of the plants in different days of the week. As in Figures 1(a) and 1(b), when he is present at a plant, he manages the operations and audits the actions of the AM of that plant. However, when he is at the other plant, MM has the responsibility to audit the operations of AM without completely assuming the GM role, which is considered to have many additional permissions. Since the hierarchical relationships among the roles change, this situation can be specified by DTRH, by simply having a policy which makes MM move to the second level, on top of AM only on the days when GM is away. Nevertheless, it is still possible to represent the scenario in this example using a static role hierarchy. However, lack of temporal role hierarchies will force the system administrators to create a dummy role, like "Manager and Auditor" (MA), that does not essentially represent a regular job function. Also, this role should have the required permission and hierarchy assignments that MM needs. Moreover, MM should be assigned to two separate roles (MM and MA) which are enabled and disabled in regular time intervals. Clearly, creation of such redundant dummy roles increases the administrative burden [4].

Role delegation, is another way of handling such scenarios [3,12,1,11,5]. Users are delegated to the necessary roles of the users that are away. Although this process seems more practical than dealing with dummy roles, some complications are possible. The *delegatees* might not be allowed to assume all of the permissions of the role that they are delegated. At this point, we have to note that [12] and [5] provides a scheme for partial delegation by either temporary dummy roles or blocking some permissions in the delegated role. Even though our example scenario can be modeled using role delegation without imposing significant overhead, employing temporal role hierarchies has still an advantage as it lends itself for performing *safety analysis* since none of the role delegation studies propose it.

The main contribution of this paper is to perform safety analysis of $\mathrm{TRBAC}_{\mathrm{RH}}$. Whether handling the temporal role hierarchies is done using the specification of DTRH, using dummy roles or delegation, none of the prior work on safety analysis considers

RBAC models with temporal constraints on role hierarchies. The safety analysis of TRBAC_{RH} leads us to expand the set of possible safety questions. As discussed above, having DTRH can reduce redundancy and facilitate the administration in various dynamic work environments. Since we have a dynamic hierarchy, which is controlled by an administrative model (Section 3.2), the implicit role assignments require much more attention than before. There is no problem of this sort in the case of static role hierarchies, however a simple manipulation in the hierarchy could create a security breach, and should be detected in advance to prevent any such occurrence. Therefore, we need to examine new security questions in the analysis of systems with dynamic temporal role hierarchies. A possible safety question can be: "Will a user u ever get *implicitly* assigned to role r in the future?" A liveness question can be: "Will a user u ever lose any role that he is *implicitly* assigned in the future?" Finally, a mutual exclusion question can be: "Will users u_1 and u_2 ever get *implicitly* assigned to role r at the same time slot in the future?"

We define the TRBAC_{RH} by extending the definitions of TRBAC with the dynamic temporal role hierarchies, as well as its administrative model. We also propose an approach to perform safety analysis on this model to answer potential safety questions discussed above. For our analysis, we adopt the TRBAC safety analysis approach recently proposed by Uzun et al. [10]. Specifically, we decompose the TRBAC_{RH} analysis problem into multiple RBAC analysis problems and simply employ existing RBAC analysis techniques to solve the TRBAC_{RH} analysis.

2 Preliminaries

Temporal Role Based Access Control Model: Temporal RBAC was first proposed by Bertino et al. [2] to be an RBAC model with the capability of role enabling and disabling via periodical and duration constraints. Joshi et al. [7] extended this model to have temporal capabilities on user to role and role to permission assignments along with some other components like constraints, role triggers and role hierarchies. In both of these models, the time notion is embedded using *Calendar* expression which is composed of *periodicity* and *duration* expressions. Uzun et al. [10] provide a simplified version of the temporal models of [2] and [7] in order to provide strategies to perform safety analysis on TRBAC. The main difference between this model and the models by [2] and [7] is the simplified calendar expression, which only has periodicity constraints. Since we base our temporal role hierarchies on the TRBAC model by [10], we now give some of its components and notation.

Let U, R, $PRMS$ be finite sets of users, roles and permissions, respectively, of a traditional RBAC system. Although the PA relation, $PA \subseteq PRMS \times R$ is defined the same way as in RBAC [9], UA relation is defined in a different way, considering the temporal nature of the model. The unit time is represented by discrete *time slots*. Let T_{MAX} be a positive integer. A *time slot* of $Times$ is a pair $(a, a+1)$, where a is an integer, and $0 \leq a < a+1 \leq T_{MAX}$. We use the term *time interval*, for a consecutive series of time slots. A *schedule* s over T_{MAX} is a set of time slots. The model has the periodicity property (just like the preceding TRBAC models) which is provided by having schedules that repeat themselves in every T_{MAX} time slots. This

temporal notion is embedded into two different components of the model: $TUA \subseteq (U \times R \times S)$ is the *temporal user to role assignment* relation and $RS \subseteq (R \times S)$ is the *role-status* relation which controls the role enabling and disabling. A tuple $(u, r, s) \in TUA$ represents that user u is a member of the role r only during the time intervals of schedule s. A tuple $(r, s) \in RS$ imposes that role r is *enabled* only during the time intervals of s and therefore it can only be assumed at these times. Thus, a user u can assume role r at time $t \in [0, T_{MAX}]$ provided that $(u, r, s_1) \in TUA$, $(r, s_2) \in RS$, and $t \in (s_1 \cap s_2)$, for some schedules s_1 and s_2. The administrative model for TRBAC is used to change these two temporal components. More specifically, the administrative rules `t_can_assign`, `t_can_revoke`, `can_enable` and `can_disable` is used to assign / revoke roles to users, and enable / disable roles, respectively. Applying these rules change the assignments along with their schedules.

Static and Temporal Role Hierarchies: A Role Hierarchy relationship $(r_1 \geq r_2)$ between roles r_1 and r_2 means that r_1 is superior to r_2, so that any user who has r_1 assigned, can inherit the permissions assigned to r_2. In traditional RBAC, this assignment is, naturally, static [9]. However, presence of a temporal dimension brings some additional flexibility on how these hierarchies work. Previously proposed models for temporal role hierarchies [6,7] focus on the permission and activation inheritance through the role hierarchies in the presence of role enabling and disabling. Particularly, the role hierarchy is still static, but the temporal constraints on the role enabling determines whether the role hierarchy will provide inheritance for a role at a given time. Three types of hierarchy relations for temporal domain are proposed: *Inheritance Only Hierarchy* $(\geq)$, *Activation Only Hierarchy* $(\succeq)$ and *General Inheritance Hierarchy* $(\gg)$. Lastly, a *Hybrid Hierarchy* exists when the pairwise relations among different roles are of different types. Interested readers may consult [6,7] for details.

3 Dynamic Temporal Role Hierarchies in TRBAC

The flexibility to have a different hierarchy structure at different time intervals makes Dynamic Temporal Role Hierarchy different than the Temporal Role Hierarchy in [7]. In order to represent this additional capability, we provide a new Role to Role Relation called *dynamic temporal role hierarchy policy*, and an administrative model to make modifications on it, like the RRA97 relation of ARBAC97 [8].

3.1 Dynamic Temporal Role Hierarchy Policies

A TRBAC policy with the presence of dynamic temporal role hierarchies, denoted as TRBAC_{RH}, and is defined as follows: Let S be the set of all possible schedules over T_{MAX}. A TRBAC_{RH} policy over T_{MAX} is a tuple $M = \langle U, R, PRMS, TUA, PA, RS, DTRH \rangle$ where $DTRH \subseteq (R \times R \times S \times \{weak, strong\})$ is the *temporal role hierarchy relation*. In our model, $DTRH$ is represented as a collection of dynamic temporal role hierarchy policies, which are tuples consisted of a pair of roles associated with a schedule that denotes the time slots that the policy is valid. In our model, we have dynamic temporal role hierarchy for inheritance only relation $DTRH_I$, for activation only relation $DTRH_A$ and for general inheritance relation $DTRH_{IA}$. For notational simplicity, we use $DTRH$, when we refer to any one of them.

Definition 1. *A dynamic temporal role hierarchy policy* $(r_1 \geq_{s,weak} r_2) \in DTRH_I$ *between roles* r_1 *and* r_2 *is an inheritance-only weak temporal relation, that is valid in the time slots specified by a schedule* s*. Under this policy, a user* u *who can activate* r_1 *can inherit permissions of* r_2 *at time* t *if (1)* $(u, r_1, s_1) \in TUA$ *(2)* $(r_1, s_2) \in RS$ *and (3)* $t \in (s_1 \cap s_2 \cap s)$*, provided that there exists schedules* s_1 *and* s_2 *that determine the time slots that* u *is assigned to* r_1 *and* r_1 *is enabled, respectively.*

Definition 2. *A dynamic temporal role hierarchy policy* $(r_1 \succeq_{s,weak} r_2) \in DTRH_A$ *between roles* r_1 *and* r_2 *is an activation-only weak temporal relationship, that is valid in the time slots specified by a schedule* s*. Under this policy, a user* u *can activate* r_2 *at time* t *if (1)* $(u, r_1, s_1) \in TUA$ *(2)* $(r_2, s_2) \in RS$ *and (3)* $t \in (s_1 \cap s_2 \cap s)$*, provided that there exists schedules* s_1 *and* s_2 *that determine the time slots that* u *is assigned to* r_1*, and* r_2 *is enabled, respectively.*

Definition 3. *A dynamic temporal role hierarchy policy* $(r_1 \gg_{s,weak} r_2) \in DTRH_{IA}$ *between roles* r_1 *and* r_2 *is a general weak temporal relationship, that is valid in the time slots specified by a schedule* s*. Under this policy, a user* u *can activate* r_2 *at time* t*, or inherit permissions of* r_2 *if (1)* $(u, r_1, s_1) \in TUA$ *(2)* $(r_2, s_2) \in RS$ *and (3)* $t \in (s_1 \cap s_2 \cap s)$*, provided that there exists schedules* s_1*, and* s_2 *that determine the time slots that* u *is assigned to* r_1 *and* r_2 *is enabled, respectively.*

In the above three definitions, the relations become strong, (i.e: $r_1 \geq_{s,strong} r_2) \in DTRH_I$, $(r_1 \succeq_{s,strong} r_2) \in DTRH_A$ and $(r_1 \gg_{s,strong} r_2) \in DTRH_{IA}$), when (2) is replaced with $(r_1, s_2), (r_2, s_3) \in RS$ and (3) is replaced with $t \in (s_1 \cap s_2 \cap s_3 \cap s)$ where s_3 is the schedule that determine the time slots that r_2 is enabled. Now, let us give an example about how these policies work.

Consider that we have temporal access control system with three roles, r_1, r_2 and r_3 and $T_{MAX} = 3$. Suppose that we have the following $DTRH$ and RS policies defined: (1) $(r_1, (0, 2)) \in RS$ (2) $(r_2, (0, 1)) \in RS$ (3) $(r_3, (1, 3)) \in RS$ (4) $(r_1 \geq_{(0,3),strong} r_2) \in DTRH_I$ (5) $(r_1 \geq_{(0,3),weak} r_3) \in DTRH_I$. According to these policies, a user who has r_1 assigned can inherit permissions of r_2 only in the time interval $(0, 1)$, because r_2 is not enabled in $(1, 3)$ and the role hierarchy relation is strong. However, u can inherit permissions of r_3 in $(0, 2)$, even if r_3 is not enabled in $(0, 1)$, since the relation is weak.

A *hybrid* relation in a dynamic temporal role hierarchy, $DTRH_H$, may contain all of the tuples defined in the above definitions, and each relation among different roles is determined using the type of that specific relation.

Dynamic temporal role hierarchy policies $(r_1 \geq_{s,weak} r_2) \in DTRH$ satisfy the following properties for a given schedule s: (1) Reflexive: $(r_1 \geq_{s,weak} r_1) \in DTRH$, (2) Transitive: If $(r_1 \geq_{s,weak} r_2), (r_2 \geq_{s,weak} r_3) \in DTRH$, then $(r_1 \geq_{s,weak} r_3) \in DTRH$. (3) Asymmetric: If $(r_1 \geq_{s,weak} r_2) \in DTRH$ then $(r_2 \geq_{s,weak} r_1) \notin DTRH$. These properties apply for both strong and the other types of relations ($\succeq, \gg$) as well.

3.2 Administrative Model for TRBAC with Dynamic Temporal Role Hierarchies

Administrative models are required for RBAC systems in order to govern the modifications on the access control policies [8]. Without these models, the access control

policies are considered static as $DTRH$, which is a static policy unless there is an administrative model to allow for modifications. Uzun et al.[10] present an administrative model for TRBAC. In this section, we propose an extension to that model, which makes it cover TRBAC_{RH}. This extension is composed of a rule called `t_can_modify`, similar in semantics to the `can_modify` in [8], but with additional capabilities for temporal dimension. This rule updates the valid time slots of the dynamic temporal role hierarchy policies. Also, in contrast to precondition structures that have been proposed in the literature for other administrative rules (like `can_assign`), it has two sets of preconditions, one for senior and one for junior role in order to protect the integrity of the hierarchy. The rule is composed of eight parameters that should be satisfied to execute the rule. Let t be the time slot that the rule is required to be executed. (1) $admin$ denotes the administrative role that a user must belong in order to execute the rule. (2) s_{rule} is a schedule that denotes the time slots in which the rule is executable. In order to satisfy, $t \subseteq s_{rule}$. (3) $s_{hierarchy}$ is a schedule that denotes the time slots of the hierarchy policy that the rule is authorized to modify. (4) $type \in \{strong, weak\}$ denotes the type of the hierarchy relation. (5) r_{sr} is the senior role of the hierarchy policy. (6) r_{jr} is the junior role of the hierarchy policy. (7) $SR(Pos, Neg)$ denotes the positive and negative preconditions of the senior role r_{sr}. The preconditions are satisfied in the following way: Let $\hat{s}$ denote the time slots that are intended to be modified by the rule ($\hat{s} \subseteq s_{hierarchy}$). For each $r \in Pos$, there must be a role hierarchy policy $(r \geq_{\hat{s},type} r_{sr}) \in DTRH$ and for each $r \in Neg$, there must not be a hierarchy policy $(r \geq_{\hat{s},type} r_{sr}) \in DTRH$. (8) $JR(Pos, Neg)$ denotes the positive and negative preconditions of the junior role r_{jr}. The preconditions are satisfied in the following way. Let $\hat{s}$ denote the time slots that are intended to be modified by the rule ($\hat{s} \subseteq s_{hierarchy}$). For each $r \in Pos$, there must be a role hierarchy policy $(r_{jr} \geq_{\hat{s},type} r) \in DTRH$ and for each $r \in Neg$, there must not be a hierarchy policy $(r_{jr} \geq_{\hat{s},type} r) \in DTRH$. Under these parameters, a tuple $(admin, s_{rule}, SR(Pos, Neg), JR(Pos, Neg), s_{hierarchy}, r_{sr}, r_{jr}, type) \in$ `t_can_modify` allows to update the role hierarchy relation $r_{sr} \geq_{s,type} r_{jr}$ as follows: Let $\hat{s}$ be a schedule over T_{MAX} with $\hat{s} \subseteq s_{hierarchy}$. Then, if this rule can be executed at time t, and the preconditions are satisfied w.r.t. schedule $\hat{s}$, then the tuple $r_{sr} \geq_{s,type} r_{jr}$ is updated to $r_{sr} \geq_{s \cup \hat{s},type} r_{jr}$ or $r_{sr} \geq_{s \setminus \hat{s},type} r_{jr}$, depending on the intended modification. This definition is for inheritance only hierarchies, but it also applies to activation only and general inheritance hierarchies, by replacing $\geq$ with $\succeq$ and $\gg$.

4 Toward Safety Analysis of TRBAC Systems with Dynamic Temporal Role Hierarchies

An important aspect of any access control model is its safety analysis. This is necessary to answer security questions such as those given in Section 1. In this section, we examine how safety analysis of the TRBAC model with DTRH can be carried out. The basic idea is to use the decomposition approach proposed in [10], which reduces the TRBAC safety problem into multiple RBAC safety sub-problems and handles each sub-problem separately using an RBAC safety analyzer that has been proposed in the literature. Here, we need to make two assumptions: (1) The administrative model for the dynamic temporal role hierarchy given in Section 3.2 cannot completely be decomposed into a traditional RBAC. The underlying reason is the precondition structure of

dynamic temporal role hierarchies, that does not exist in ARBAC97 role hierarchy component RRA97. Decomposing TRBAC into multiple RBAC safety sub-problems relax the schedule components, but the precondition requirements remain in effect. Since, there is no known RBAC safety analyzer that can handle preconditions in role hierarchies, we assume that the administrative model for dynamic temporal role hierarchy contains rules with no precondition requirement for safety analysis purposes. (2) The safety questions and the structure of the analysis in [10] are based on checking the presence of a particular role (or roles) being assigned to a particular user. There is no permission level control available in the model. Hence, we restrict our safety analysis on the Activation-Only hierarchies. So, we assume $DTRH = DTRH_A$. Moreover, we assume all relationships are strong.

The decomposition that we utilize is the Role Schedule Approach of [10]. In this approach the sub-problems are constructed using the role schedules of the administrative rules. In TRBAC, the administrative rules for role assignment and role enabling have two separate schedules: Rule Schedule and Role Schedule. Rule schedule is similar to the s_{rule} of `t_can_modify` and determines the periods in which the rule is valid. Similarly, the role schedule is similar to the $s_{hierarchy}$ of the `t_can_modify` and determines the time slots that the rule is authorized to modify TUA and RS policies. The key observation that makes this decomposition possible is the independency among different time slots, and the periodic behavior of the model. Particularly, if we are interested in the safety analysis of a time slot t, then we only need to consider the administrative rules, that are authorized to modify time slot t of TUA and RS relations. Furthermore, since the system is periodic, for any long run analysis, we can safely assume that the validity constraints of the rules (s_{rule}) will be enforced implicitly even if they are ignored. For detailed discussion, readers may refer to [10]. In TRBAC_{RH}, we perform similar operations to generate sub-problems. Our model has the same property of having independency among the time slots. First, we define the dynamic temporal role hierarchy policies for a single time slot t, denoted as $DTRH_t \subseteq DTRH$ to be a collection of role hierarchy policies in the system that satisfies:

$$DTRH_t = \{(r_i \succeq_{s_1,strong} r_j) | t \in s_1\} \forall r_i, r_j, \in R$$

where s_1 and s_2 are the schedules of role hierarchy rules. The dynamic role hierarchy policies in the time slot t of RH_t, then reduces to $(r_i \succeq_{strong} r_j)$. These policies become non-temporal role hierarchy policies for the time slot t. Similarly, administrative rules for a specific time slot t is defined as:

$$(admin, s_{rule}, SR(\emptyset, \emptyset), JR(\emptyset, \emptyset), s_{hierarchy}, r_{sr}, r_{jr}, type) \in \texttt{t\_can\_modify}_t, t \in s_{hierarchy}$$

Hence, the set $\texttt{t\_can\_modify}_t$ contains the administrative rules that are authorized to modify the t^{th} time slot of the dynamic temporal role hierarchy policies. In other words, if one is interested in the safety analysis for time slot t, then there is not any other administrative rule authorized to modify time slot t, but the rules in $\texttt{t\_can\_modify}_t$. The administrative rules for t are reduced to $(admin, r_{sr}, r_{jr}, type)$ which belong to the `can_modify` $\subseteq admin \times 2^R$ of the RRA97 of [8].

The above defined two sets, RH_t and $\texttt{t\_can\_modify}_t$ provide safety analysis using the Role Schedule Approach in [10]. When decomposed, there are k RBAC safety

sub-problems, where k is the number of time slots. Any RBAC safety analyzer that is capable of handling role hierarchies and the `can_modify` relation can be used to analyze these sub-problems. Repeating this operation for each of these k sub-problems will yield the safety analysis of the TRBAC_{RH} system. The computational complexity of this process depends linearly on the computational complexity of the RBAC safety analyzer and k.

5 Conclusion and Future Work

In this paper, we introduced the concept of dynamic temporal role hierarchy, which, can be viewed as a different role hierarchy at different times. We develop an administrative model for RBAC with dynamic temporal role hierarchies along with a road map for its safety analysis. Currently we are implementing safety analysis using the RBAC analysis tools available. Implementing administrative model of the dynamic temporal role hierarchies in the safety analysis will require new tools to fully capture the capabilities of the model. This will be our future work.

Acknowledgements. This work is partially supported by the National Science Foundation under grant numbers CNS-0746943 and CNS-1018414.

References

1. Barka, E., Sandhu, R., et al.: A role-based delegation model and some extensions. In: NISSC, vol. 4, pp. 49–58 (2000)
2. Bertino, E., Bonatti, P.A., Ferrari, E.: TRBAC: A temporal role based access control model. ACM Transactions on Information and System Security 4(3), 191–233 (2001)
3. Crampton, J., Khambhammettu, H.: Delegation in role-based access control. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 174–191. Springer, Heidelberg (2006)
4. Guo, Q., Vaidya, J., Atluri, V.: The role hierarchy mining problem: Discovery of optimal role hierarchies. In: ACSAC 2008, pp. 237–246. IEEE (2008)
5. Joshi, J.B.D., Bertino, E.: Fine-grained role-based delegation in presence of the hybrid role hierarchy. In: SACMAT, pp. 81–90 (2006)
6. Joshi, J.B.D., Bertino, E., Ghafoor, A.: Hybrid role hierarchy for generalized temporal role based access control model. In: COMPSAC 2002, pp. 951–956. IEEE (2002)
7. Joshi, J.B.D., Bertino, E., Latif, U., Ghafoor, A.: A generalized temporal role based access control model. IEEE Transactions on Knowledge and Data Engineering 17(1), 4–23 (2005)
8. Sandhu, R., Bhamidipati, V., Coyne, E., Ganta, S., Youman, C.: The ARBAC97 model for role-based administration of roles: preliminary description and outline. In: ACM Workshop on Role-Based Access Control, pp. 41–50 (1997)
9. Sandhu, R., Coyne, E., Feinstein, H., Youman, C.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)
10. Uzun, E., Atluri, V., Sural, S., Vaidya, J., Parlato, G., Ferrara, A., Parthasarathy, M.: Analyzing temporal role based access control models. In: SACMAT. ACM (2012)
11. Zhang, L., Ahn, G.J., Chu, B.T.: A rule-based framework for role-based delegation and revocation. TISSEC 6(3), 404–441 (2003)
12. Zhang, X., Oh, S., Sandhu, R.: PBDM: a flexible delegation model in RBAC. In: SACMAT, pp. 149–157. ACM (2003)

Author Index